Technical Report
CMU/SEI-94-TR-013
ESC-TR-94-013
August 1994

Benefits of CMM-Based Software Process Improvement: Initial Results

James Herbsleb
Anita Carleton
James Rozum
Jane Siegel
David Zubrow

Technical Report
CMU/SEI-94-TR-013
ESC-TR-94-013
August 1994

Benefits of CMM-Based Software Process Improvement: Initial Results

James Herbsleb
Anita Carleton
James Rozum
Jane Siegel
David Zubrow

Empirical Methods

Software Process Measurement

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
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.

FOR THE COMMANDER

(signature on file)

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

This work is sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218. Phone: (703) 767-8222 or 1-800 225-3842.]

Table of Contents

List of Figures

List of Tables

Acknowledgments

Many people were involved in the production of this technical report. First, we would like to thank all of those who provided us with data and patiently answered our questions about software process improvement in their organizations (see Table 1):

Edward Weller, Ron Radice	Bull HN
Constance Ahara, Debbie De Toma, Jim Perry	GTE Government Systems
Sue Stetak	Hewlett Packard
Bob Rova, Ken Shumate, Ron Willis, Thomas Winfield	Hughes Aircraft Co.
Ted Keller	Loral Federal Systems (formerly IBM Federal Systems Company)
Barbara Bankeroff	Lockheed Sanders
John Pellegrin, Richard Stenglein, Bob Yacobellis	Motorola
Leitha Purcell	Northrop
Peter Burrows, Harvey Wohlwend	Schlumberger
Henry Gueldner	Siemens Stromberg-Carlson
Marie Silverthorn, Mary Vorgert, Bob Stoddard	Texas Instruments
Kelley Butler	United States Air Force Oklahoma City Air Logistics Center
Brenda Zettervall	United States Navy Fleet Combat Direction Systems Support Activity

Table 1. Organizations and People Who Provided Data

We would also like to thank Jim Armitage, Jack Harding, and Bob Park for their helpful comments on earlier versions of the ideas contained in this report. We also appreciate the help of Julia Allen and Ron Radice who reviewed several drafts of this report and provided numerous helpful comments along the way. Finally, we would like to thank Suzanne Couturiaux for her skillful help with the technical editing.

To the Reader

This technical report is intended to provide the reader with some initial results of the effects of software process improvement efforts on organizations. It is intended primarily for software practitioners, members of software engineering process groups, and software managers interested in understanding the business case for investing in software process improvement. It assumes a familiarity with the capability maturity model for software (CMM) [Paulk 93a, 93b]. There is a forthcoming related special report, CMU/SEI-94-SR-13, which is an executive summary of the material contained in this technical report.

Readers interested only in gleaning the results as quickly as possible can go directly to Chapter 3 for an overview of the results and Chapter 4 which presents case studies of the results obtained from specific improvement efforts in five organizations. We urge anyone planning to make use of data contained in this report to read Section 3.1, which includes information essential to appropriate interpretation of the data. Those readers wanting more background on the motivation of the effort and on how the data were collected should also read Chapters 1 and 2. Chapter 5 is intended for those readers who are interested in gathering and analyzing their own data. This chapter provides an overview of the technical issues involved in measuring the results of software process improvement.

This paper is organized as follows. We begin with the background of this effort in Chapter 1, including our motivation, the selection of participating organizations, and the kinds of data we requested from them. Chapter 2 is a summary of the improvement activities of the participating organizations, while Chapter 3 contains the aggregated results. In 3.1, we describe the criteria we applied for inclusion of data in this study, and explicitly discuss what we see as the major limitations of this particular data set. We go on to present the results for cost of the process improvement effort (3.2.1), productivity (3.2.2), rework (3.2.3), calendar time (3.2.3), quality (3.2.4), and business value (3.2.5).

Next, in Chapter 4 we present brief case studies of process improvement in five organizations: Bull HN, Hughes Aircraft, Schlumberger, Texas Instruments, and Oklahoma City Air Logistics Center. These case studies give you an opportunity to read about how specific activities produced results, and what lessons each organization believes it learned from the experience.

In Chapter 5, we identify and discuss the major technical issues that we confronted as we gathered data for this report. Finally, in Chapter 6, we provide conclusions about SPI efforts.

Benefits of CMM-Based Software Process Improvement: Initial Results

Abstract. Data from 13 organizations were collected and analyzed to obtain information on the results of CMM-based software process improvement efforts. We report the cost and business value of improvement efforts, as well as the yearly improvement in productivity, early defect detection, time to market, and post-release defect reports. Improvement efforts and results in five organizations are reported in more depth in case studies. In addition, technical issues that we confronted as we tried to measure the results of software process improvement are discussed. We end with conclusions about the results of SPI efforts.

1. Background

1.1 Motivation for This Empirical Study

Process improvement within software organizations is gaining momentum. More organizations today are initiating software process improvement efforts. One indication of this trend is the number of assessments performed each year. For most CMM-based software process improvement (SPI) efforts, the first step is an assessment of the current capability of the organization to develop software. We maintain a database of assessment results, and the latest figures on first assessments are shown in Figure 1. While this database contains just the fraction of all assessments reported to the SEI, it serves as a reasonable indicator of increasing interest. As shown in the figure, more first assessments were reported to the SEI in the last two years than in all previous years combined.

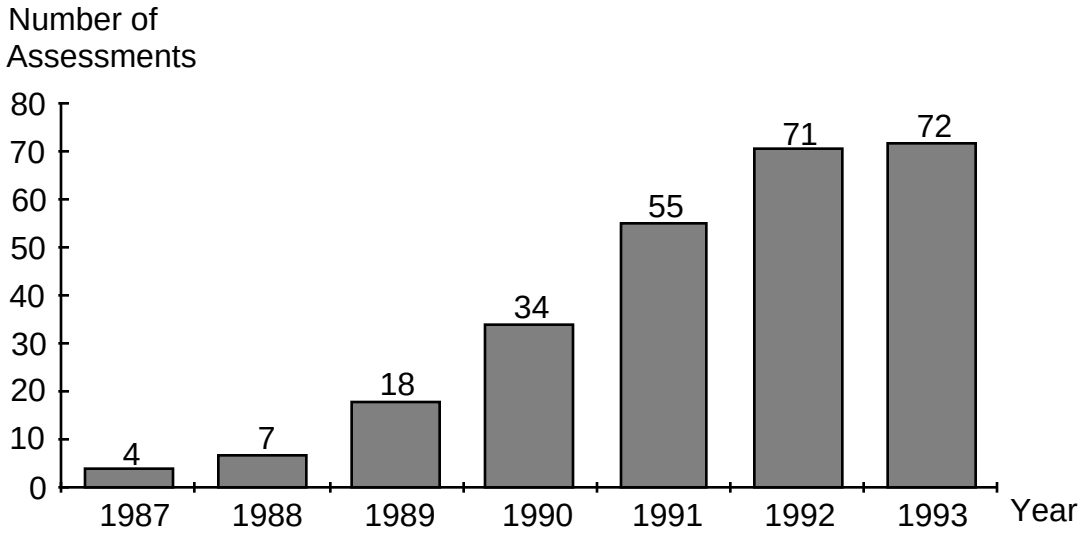

Figure 1. Number of First Assessments by Year Within Software Organizations

But for those organizations that have had active software process improvement efforts for several years, management is asking for quantitative evidence regarding the value returned for the investments made. Many other organizations are reluctant to make an initial commitment of resources without data they can use to build a business case for the expenditures. Indeed, investment in software process improvement is just one strategy that a company might take for improving its business performance.

The importance of reliable data on which these strategic decisions can be based was reinforced by a recent survey done by the SEI. In 1993, the SEI conducted a survey at the fifth Software Engineering Process Group (SEPG) National Meeting. This event is attended primarily by software engineers and managers responsible for process improvement in their organizations. Part of the survey asked the attendees about their most pressing needs. The top-ranked need reported by survey respondents was the need for quantitative information regarding the benefits from software process improvement efforts. This expressed need provided much of the impetus for the work reported here.

1.2 Responding to Community Needs

The SEI also surveyed SEPG members to determine the feasibility of this study as a means of responding quickly to the community's needs. In the fall of 1993, members of the Empirical Methods Project of the SEI contacted representatives of several organizations where CMM-based SPI occurred prior to 1990. Each representative was asked to characterize readily available data from a substantial list of information the SEI believed would be needed to produce this technical report on the results of SPI. For example, CMM-based appraisal results, data about changes in effort, cycle time, product quality, etc., were sought. The SEI asked 20 early adopting organizations engaged in CMM-based software process improvement to participate in this first national study of CMM-based SPI. Initially, all 20 organizations provided some information to the SEI. Seven of the 20 organizations were unable to participate due to constraints in their ability to release data, weak ties between their measurement activity and CMM-based software process improvement, etc. Thus, 13 organizations formed the basis for this initial report on CMM-based software process improvement.

Once this opportunistic sample of industry and government organizations agreed to participate in this SEI report on CMM-based software process improvement, an attempt to collect and analyze data about SPI efforts and effects began. Each organization received a brief description of the study and a request for a list of data items. For this study, the SEI solicited readily available data and information about the measures used and measurement methods applied by each organization. We expected that organizations would vary substantially in the measures they collected, and we anticipated the need to ascertain the sources and amount of variation of measures and results before aggregating data items across organizations.

Meeting the community’s need required the collection of actual organizational data on projects, products, and improvement efforts. These data are typically held as highly sensitive and confidential. Realizing this, the SEI offered confidentiality for all the data supplied to it by

the participating organizations. Each participating organization was asked to decide whether their data would be reported anonymously, identified as part of a group, or identified by organization name. Also, each organization was guaranteed the opportunity to verify and review material prior to release outside the SEI. Upon request, nondisclosure forms were signed by designated points of contact for participating organizations and the SEI.

1.3 Data Collection

1.3.1 Participating Organizations

The following list identifies the 13 organizations that participated in this study. They are a diverse group, including Department of Defense contractors, commercial organizations, and military organizations. They also represent a wide range of maturity levels. The range of application areas covered is also diverse, including such domains as telecommunications, embedded real-time systems, information systems, and operating systems.

- Bull HN
- GTE Government Systems
- Hewlett Packard
- Hughes Aircraft Co.
- Loral Federal Systems (formerly IBM Federal Systems Company)
- Lockheed Sanders
- Motorola
- Northrop
- Schlumberger
- Siemens Stromberg-Carlson
- Texas Instruments
- United States Air Force Oklahoma City Air Logistics Center
- United States Navy Fleet Combat Direction Systems Support Activity

1.3.2 Data Requested

We requested several kinds of data from these organizations, as is shown in the following list:

Organizational characteristics

- Organizational environment
- Business characteristics

Software process improvement efforts

- Descriptions of SPI efforts

- Process maturity information
- Measures and techniques currently used
- Description of data collection activities

Results

- Impact of SPI on business objectives
- Impact of SPI on social factors
- Actual performance versus projections

The requested data fall into three categories: descriptive information about the organizations, information about their process improvement and measurement programs, and data about the results of their improvement efforts. These requests were based on the view that the results obtained may depend on the organizational context as well as the SPI effort itself.

2. Improvement Activities

The organizations in this study were engaged in a wide range of software process improvement activities. All of the organizations providing data that we used had action plans and were working on the key process areas appropriate to their maturity levels. We will not try to enumerate all the activities here; rather, we will just summarize those that were most frequent:

- forming process groups,
- training,
- performing peer reviews,
- devising forums for exchange of ideas and intergroup coordination, and
- inserting new technology.

Virtually all of the organizations had formed a software engineering process group (SEPG) as the organizational focus of the process improvement efforts. Many had both corporate-level SEPGs and local or site SEPGs. In addition, many organizations created other special purpose groups such as a management steering team or software mentor team to oversee, coordinate, and prioritize improvement efforts. Several organizations also had software process and technology task forces to investigate technologies (e.g., technology working group, software technology center) or to address particular process issues. (e.g., software quality engineering).

Training was an important element of the improvement activities in nearly every organization. Although we do not have complete and systematic information on all of the training programs, the most frequently offered courses appear to be project management, peer reviews, and instruction in the local development process and methods. Some courses were offered by third parties, while other organizations handled all training internally.

Another very common activity for organizations at all maturity levels was some form of peer reviews. Most of the organizations conducted code inspections, many also conducted design inspections, and several were pioneering requirements inspections. All who commented about or collected data on peer reviews were firmly convinced of their value.

A few organizations had established forums for the exchange of ideas and for coordinating efforts among groups. The more expensive type of forum is an in-house conference or workshop where SEPGs across the corporation get together to learn from each others' experiences. A more modest, but apparently effective alternative is a newsletter which informs and publicizes accomplishments.

Several organizations were changing their processes not only for process improvement per se but also to enable the insertion of new technology. The most frequently mentioned technology innovation was reuse, based either on a library or on a generic architecture. In these cases, there were many comments on the extensive process changes required to incorporate these technologies.

3. Aggregated Results of SPI

3.1 Data

To ensure reasonable quality of data for this study, we established several criteria for inclusion of data. First, we focused on software process improvement efforts based on the capability maturity model. Since this model is closely associated with the SEI, and this is the data we are most frequently asked for, we decided to make this our first priority.[1] Second, we included only data that had an interpretation that was fairly clear to us. We scoured several thousand pages of documents of many kinds, including presentation slides, memos, action plans, reports, newsletters, and so on. We often followed up examination of documents with phone calls for clarification and requests for more documents. In spite of this, there remained a few cases where we did not believe we had sufficient understanding of some potential data points to include them in this study. Third, we looked for some indication that the data point was valid. Examples of things that would satisfy this criterion are: a description of the organization measurement program, clear data definitions, detail in the data itself that seemed to indicate care was taken in their collection and handling, or a description of how these particular data were collected. These do not guarantee accuracy and precision, of course, but we regard them as reasonable reassurances appropriate for the type of data available to us.

Before describing the results, we want to identify several caveats that shape the way these data should be interpreted. These caveats identify limitations in the generality of the current results. They also serve to identify challenges for future efforts, where the selection and definition of measures, the sampling of organizations, and the sampling of projects within organizations can all be addressed more systematically.

- There may be masked tradeoffs in the data we present. It is generally possible to improve one measure, e.g., productivity, by sacrificing another, e.g., quality. We generally did not receive a complete set of "core measures" (e.g., effort, size, calendar time, and quality, see [Carleton 92]) from each organization. So we are unable to determine the extent to which gains in one measure, reported to us, were offset by declines in others, which were not reported.
- All of the organizations are doing things other than CMM-based software process improvement, e.g., gaining experience in a domain, changing personnel, and so on. Although we asked for data indicative of the results of SPI, it is probable that other factors contributed to the improvements.
- We do not know if these results are representative of all organizations engaged in CMM-based SPI. The participating organizations all had good SPI experiences, and the data come from divisions and projects that have succeeded in their current

[1] We anticipate studies in the future that will take a somewhat broader look at software process improvement.

efforts. We do not know at this point if these experiences are typical. We think the best way of interpreting these results is to view them as indicators of what is possible, given a supportive environment. In Chapter 4, we present some of the lessons learned that describe some of the factors that these organizations believe were responsible for their success.

3.2 Results

We organized the results we received into several categories:

- cost of the process improvement effort,
- productivity,
- calendar time,
- quality, and
- business value (often loosely referred to as return on investment).

These quantities were, of course, measured in very different ways in different organizations. For this reason, the absolute levels are not terribly meaningful. For example, if you do not know what constitutes a defect, and you don't know what counts as a line of code, the absolute number of defects per thousand lines of code is not terribly meaningful. On the other hand, it is reasonable to assume that there is some consistency within an organization in how it defines and collects these data. For this reason, the most informative measure is probably change within an organization over time, and that is, in general, what we report here.

The number we use for productivity and early defect detection is *gain per year,*[2] which we compute much like a (nominal) compound interest rate.[3] For example, tripling productivity

---

[2]We chose gain per year because different organizations provided data for different numbers of years, and we needed some way to make these figures comparable. We rejected the simpler procedure of taking the total gain and dividing by the number of years, because this gives an exaggerated picture of the performance. For example, a threefold gain over three years would be a 100% gain per year. This implies a doubling each year, which is not accurate. Gain per year avoids this problem.

[3]The following formula was used to calculate *percentage gain per year*:

$$\left(\sqrt[i]{\text{overall gain ratio}} - 1\right) * 100$$

where **i** is the number of years, and **overall gain ratio** is the ratio of the baseline value to the final value. So, for example, if productivity increased from 1 unit to 1.8 units over a period of 5 years, **i** = 5, **overall gain ratio** = 1.8. The resulting *percentage gain per year* is 12%.

over a period of three years is approximately a 44% gain per year. For decreasing quantities, we use *reduction per year*, which is based on an analogous calculation.[4]

Please note, we do not claim that the actual values in the intermediate years closely approximate the values given by this formula. We use this calculation only as a summary of the available data, reflecting a process that occurred over some period of years. The only purpose is to permit comparisons of processes spanning different numbers of years. *This formula is not intended as a model of the process itself.*

In the figures that follow, organizations are labeled with alphabetical identifiers to protect confidential data. Organization labeled with different letters in various charts might or might not be the same organization. Again, this is to protect the confidentiality of the organizations, so that it is not possible to identify an organization from a published figure then find additional, confidential information about that organization.

As mentioned in the preceding paragraph, some of these data have been published before, while other data points have not, to our knowledge, been previously made public. Of the 24 data points reported here, 8 have been previously published while the remaining 16 have not.

3.2.1 Cost

Figure 2 shows the amount spent by each organization on software process improvement activities. The very large differences are not unexpected, since the organizations are very

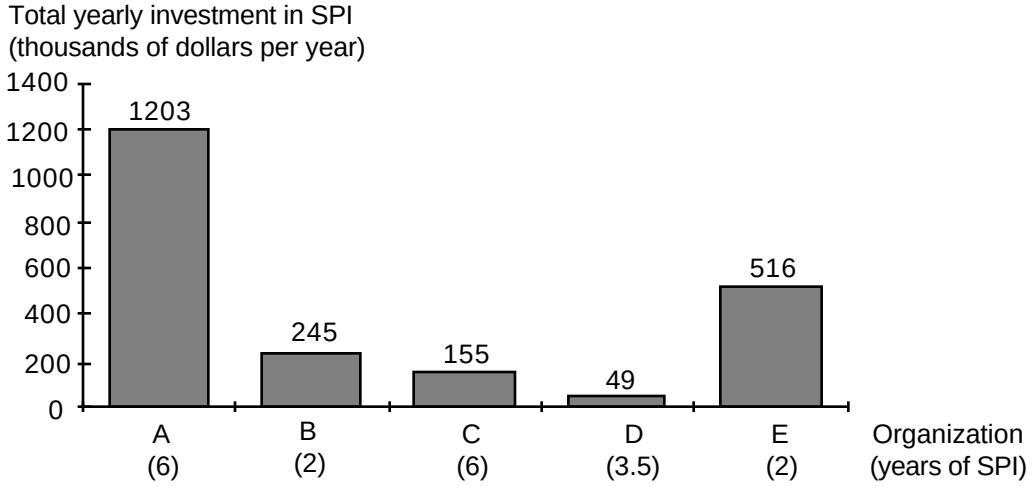

Figure 2. Thousands of Dollars per Year Spent on SPI

---

[4]The formula used to compute *percentage reduction per year* is similar to the formula for *percentage gain per year*, above:

$$\left(1 - \sqrt[i]{\text{overall reduction ratio}}\right) * 100$$

where **i** is the number of years, and **overall reduction ratio** is the ratio of the baseline value to the final value.

different in size, from division and sector level to small development organizations. In order to get a more meaningful view of the costs, we show figures for the same organizations normalized for size in Figure 3. We used the number of "software engineers" or "software professionals" as the denominator.

Interestingly, the largest organization and the smallest organization spent the two highest dollar amounts per software engineer. These two organizations also had the two highest business value figures (see the discussion of business value data, below).

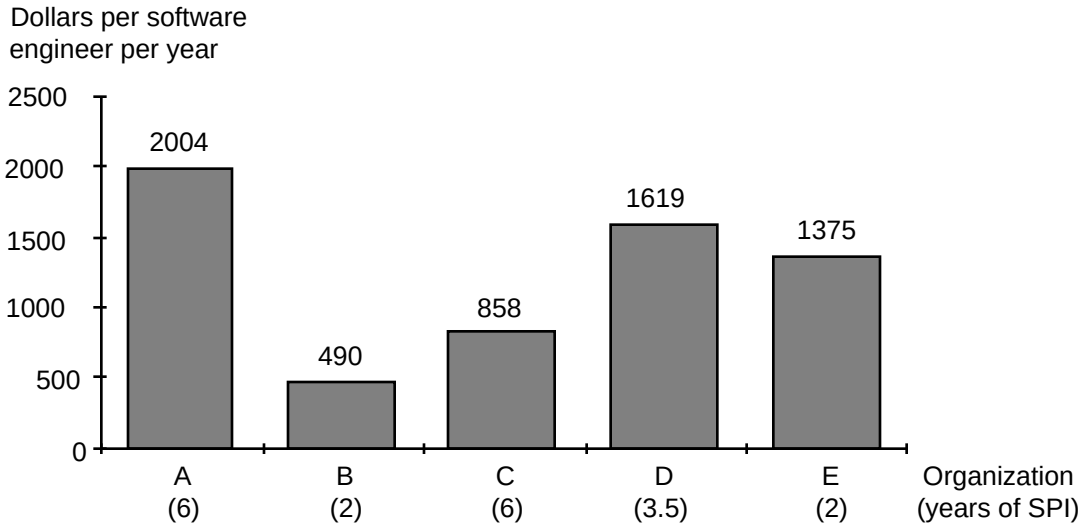

Figure 3. Dollars per Software Engineer per Year Spent on SPI.

3.2.2 Productivity

The productivity measures we report here are all variations of lines of code (LOC) per unit of time. These results, recorded as productivity gain per year, are shown in Figure 4.

The largest gain, organization G, is based mainly on a comparison of two projects, one that did not adopt SPI and one that did. They developed very similar applications, and had substantial overlap in staff. The most important factor in the superior performance of the second project was probably its requirements elicitation and management. The second largest gain, organization H, represents a process improvement effort that included a reuse program, with tools and environment to support development with reuse. The reused code is

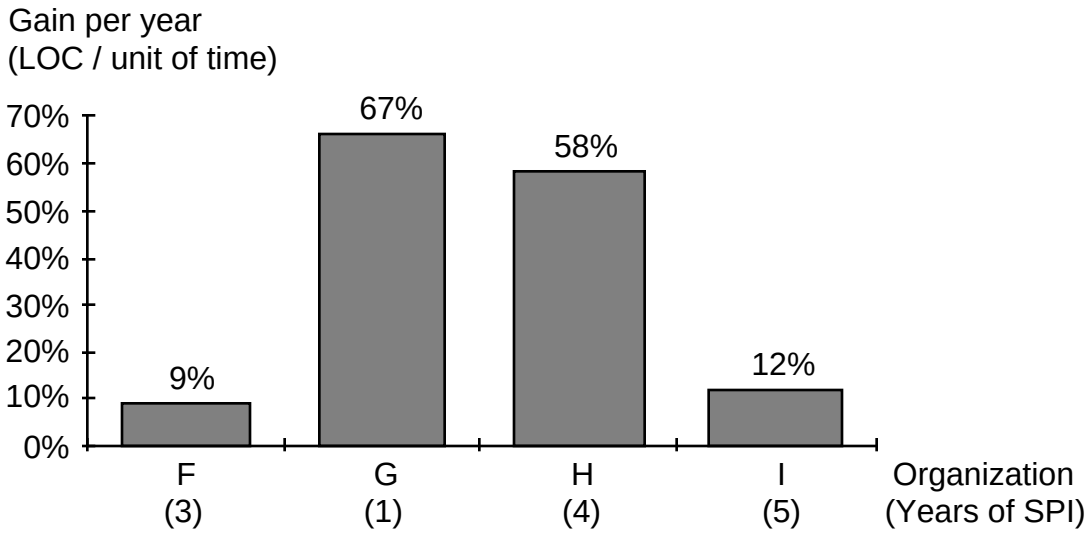

Figure 4. Gain per Year in Productivity

counted in the productivity figure; if it were excluded, the figure would still be an impressive 28% gain per year. It is impossible to separate process improvement gains from reuse gains, since the reuse would not have been possible without extensive improvements in the software process. The other two data points also represent substantial gains maintained over significant periods.

Another form of productivity for which we have data is the early detection of defects. Figure 5 shows the yearly increases in early detection of defects for three organizations. For organization J, all of the software systems on which this figure is based have gone through the entire life cycle and are now out of service. The gains represented on the chart for organization J are gains in the proportion of total life-cycle defects that were found pretest. These data are taken from a number of releases of a large application, each with substantial new and modified code. All are now out of service, so the total life-cycle defects are known. The figures for K and L, on the other hand, are based on software still in service. These figures simply represent increases in the number of defects found pretest. All of these gains represent improvement over time through the use of inspections.

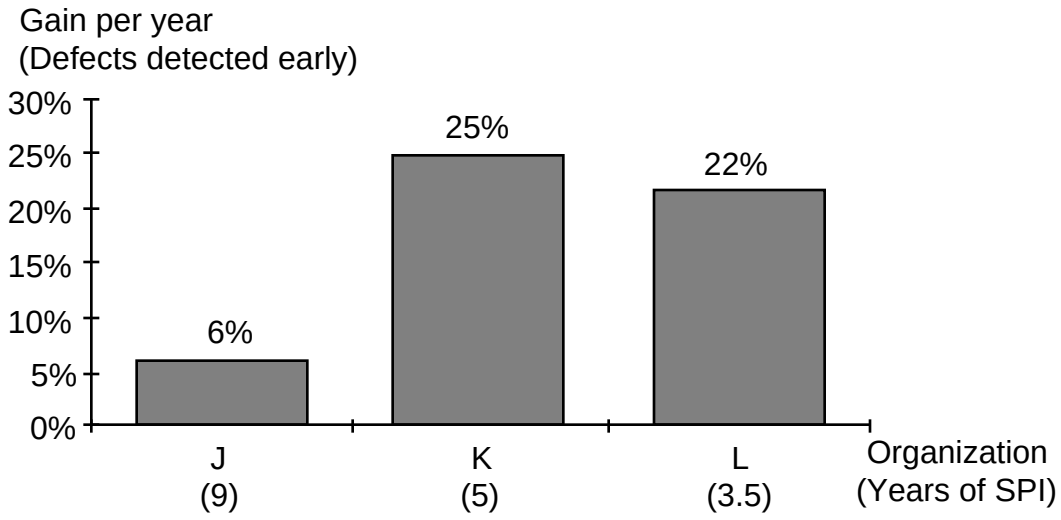

Figure 5. Gain per Year in Early Detection of Defects

These increases in early defect detection represent enormous savings in the cost of rework. By one recent estimate, for example, the cost of fixing a defect discovered pre-release is approximately $50 per line of code, while the cost of fixing a defect discovered post-release is about $4000 per line of code [Townsend 94]. In general, one would also expect substantial savings for finding defects in earlier stages of the life cycle [Humphrey 89 (p. 364)].

3.2.3 Calendar Time

Figure 6 shows the reduction in calendar time to develop software products experienced by two organizations. Obviously, these are substantial gains and represent the potential of very significant competitive advantage. Both figures are for development covering the entire life cycle. As one would expect, given the enormous gain for organization N, there are several factors contributing to reduction in time to market in this organization. The product was embedded software in a product line. The process improvement effort included reuse as an

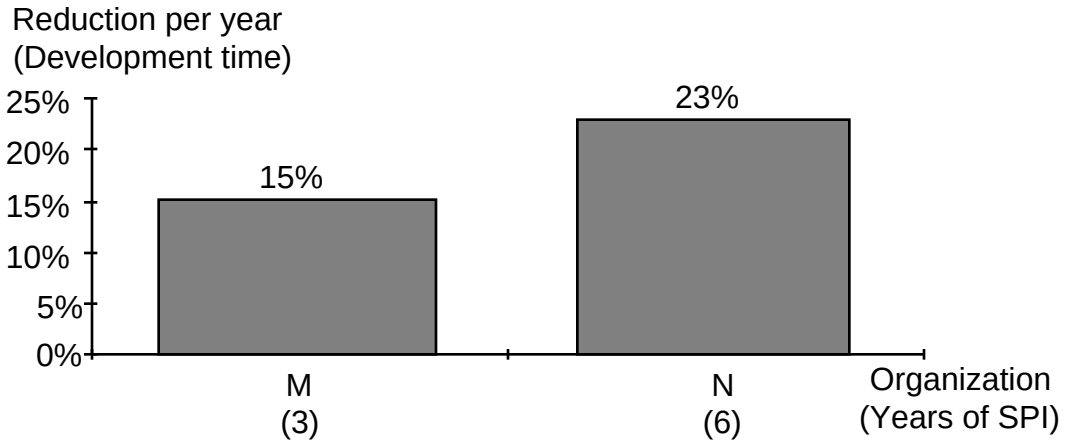

Figure 6. Reduction per Year in Calendar Time to Develop Software Systems

inseparable element. The time to market actually suffered for the first year or so as the reusable components were being developed. After two-three years, the time to market dropped very rapidly to generate these enormous gains relative to the pre-SPI (and pre-reuse) baseline.

3.2.4 Quality

Quality refers to the state of the software as it was released or delivered to customers. The most common measure of quality among the data submitted to us was the number of post-release defect reports. Figure 7 shows the yearly reduction in this measure. Most organizations were engaged in improvements aimed at lowering defect injection rates as well as improving detection. Two organizations, Q and R, had astonishing results within a very short time frame. The figure for Q is probably influenced by a major release near the beginning of the period, with much less new code over the remainder. The gain for R was very substantial as well, and was in part the result of a relatively high defect density baseline.

Organization P is remarkable for the period over which it sustained a very sizable reduction of 39% per year. The rates for P represent successive releases, with substantial amounts of new and modified code, all of which have gone through their entire life cycle. The last release had no defects reported in the new and modified code. The 39% rate of reduction, over a nine-year period, represents two orders of magnitude reduction in the post-release rate of error reports. The other two organizations, S and T, also sustained substantial reductions over a significant period.

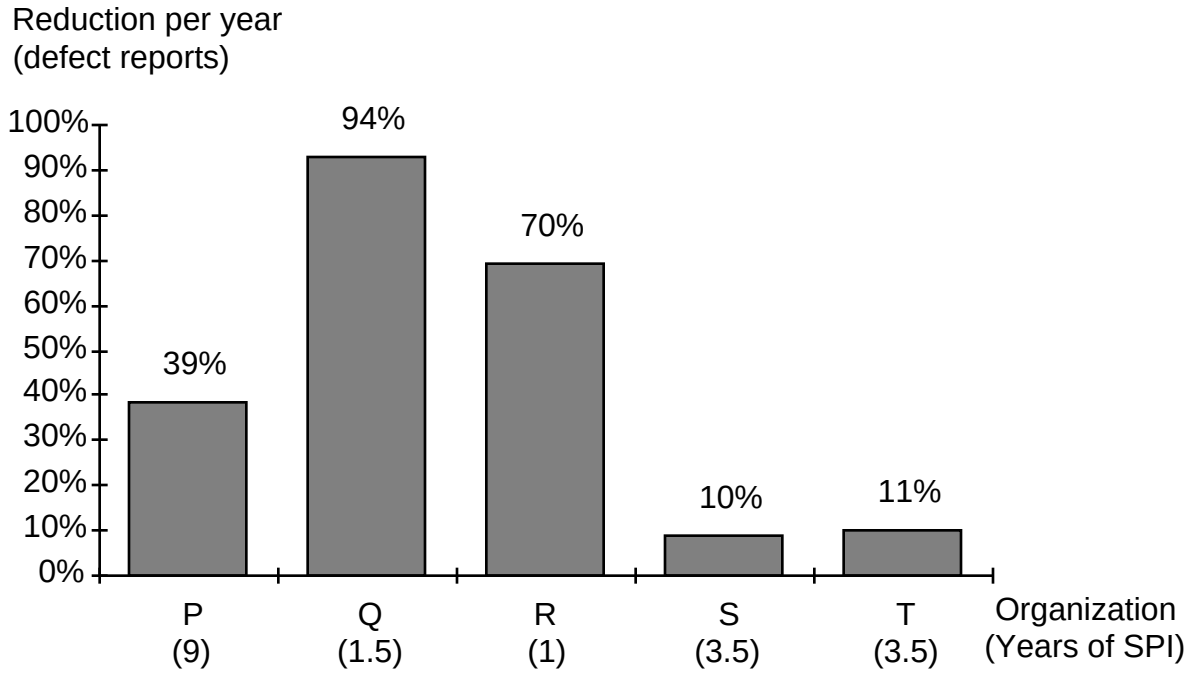

Figure 7. Reduction per Year in Post-release Defect Reports

3.2.5 Business Value

The bottom-line figure of most interest to many practitioners and managers is the value returned on each dollar invested. This is often referred to rather loosely as the "return on investment" (ROI).[5] The figures that we report are the ratio of measured benefits to measured costs.

This number, despite its importance to many in the community, is probably the most difficult of the numbers to interpret because of variations in how costs and benefits were estimated and allocated to the improvement effort. In general the business savings figures were calculated as follows. Benefits typically included savings from productivity gains and savings from fewer defects. Less frequently, savings from earlier detection were also included. The benefits generally did not include the value of enhanced competitive position, as a result, for example, of increased quality and shortened time to market. The costs of SPI generally included the cost of the SEPG, assessments, and training, but did not include indirect costs such as incidental staff time to put new procedures into place.

As Figure 8 shows, the results are very impressive. There are very few investments one can make that return a business value five or six times their cost. But we wish to stress once again that we do not know how typical these results are. We take them to be indicative of what is possible when improvement is planned and executed well and takes place in a favorable environment.

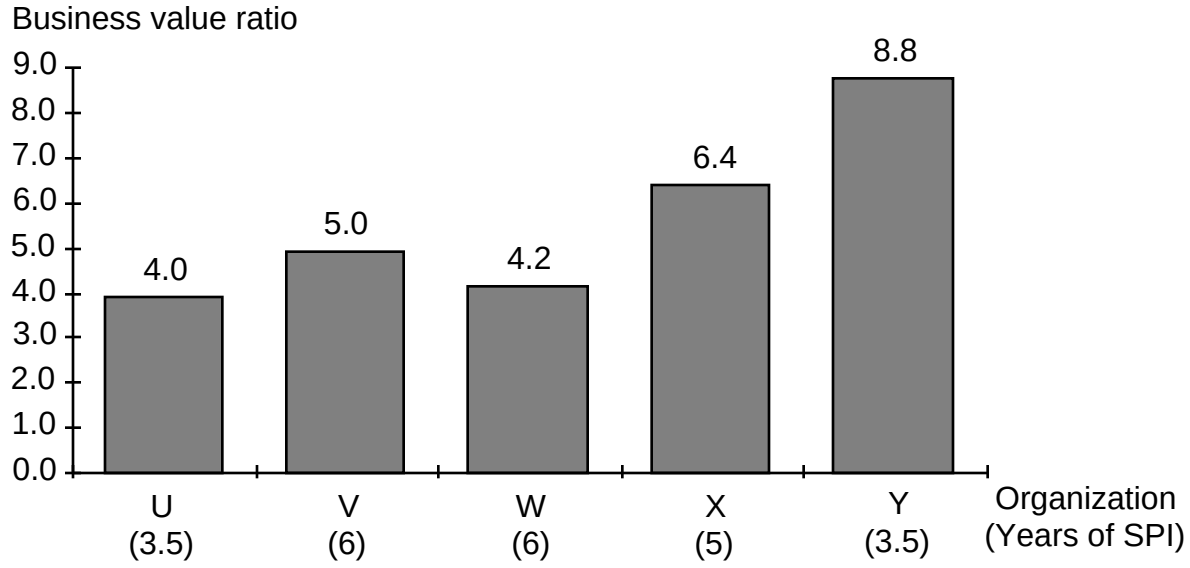

Figure 8. Business Value Ratio of SPI Efforts

[5]The definition of ROI in the world of finance involves much more complex calculations [Park 90].

3.3. Summary of Aggregated Results

There is an enormous demand for information about the results of software process improvement efforts. In response to this demand, we collected data from 13 organizations that represent a variety of maturity levels. By looking at how performance changed over time within each organization as improvement efforts were implemented, we identified substantial gains in productivity, early defect detection, time to market, and quality. These gains added up to impressive returns on the resources invested. These results, summarized in Table 2, show the kinds of gains that are possible in a favorable environment.

Table 2. Summary of the Overall Results

CATEGORY	RANGE	MEDIAN
Total yearly cost of SPI activities	$49,000 - $1,202,000	$245,000
Years engaged in SPI	1 - 9	3.5
Cost of SPI per software engineer	$490 - $2004	$1375
Productivity gain per year	9% - 67%	35%
Early detection gain per year (defects discovered pre-test)	6% - 25%	22%
Yearly reduction in time to market	15% - 23%	19%
Yearly reduction in post-release defect reports	10% - 94%	39%
Business value of investment in SPI (value returned on each dollar invested)	4.0 - 8.8	5.0

While it is important to aggregate results to see the broad view of how process improvement affects software development, it is also important to understand how particular activities, introduced into different kinds of organizations, lead to improvement. The case studies in the following section address this need. Each concludes with "lessons learned" which summarize many of the factors these organizations believe are important in getting these kinds of results.

4. Case Studies

In this section, we present case studies of software process improvement in five organizations: Bull HN, Hughes Aircraft, Schlumberger, Texas Instruments, and Oklahoma City Air Logistics Center at Tinker Air Force Base. These organizations were selected for case studies because they provided especially rich descriptions of their SPI efforts and results, and because they agreed to release this information publicly. Each case study consists of a description of software process improvement activities within the organization, the results of those activities, and the lessons the organizations extracted from their experiences.

4.1 Bull HN

4.1.1 Organization Description

BULL HN Information Systems Inc. is the US subsidiary of Groupe BULL, which is the fourth-largest European systems integrator and employs 40,000 people in over 100 countries. Groupe BULL's system integration focus is pinned heavily on helping companies migrate from proprietary to open systems. Its multi-vendor approach means BULL offers a solution service irrespective of whether the manufacturer of the required computer equipment is BULL itself or a competitor. The data in this case study is primarily from an operating system development organization in Phoenix. The operating system is a mainframe system, including database and transaction processing "executives."

4.1.2 History of SPI Efforts

Prior to the current software improvement effort at Bull, there had been two previous efforts within the last decade or so at the Phoenix site. The first was initiated in 1982 and resulted in a standard software development process based on templates for requirements and design documents and a framework for life-cycle reviews. These templates were used for about eight years, even though they were never completed.

The second effort began in 1986 and was spearheaded by a steering committee consisting of second-level managers along with some technical people. The people involved were not the same as those in the first initiative. Three working groups studied and produced reports on software maintenance, new design methods, and common standards for using the C programming language. Near the end of this process, the effort was preempted by a corporate-level initiative, which itself ended before significant results were achieved.

4.1.3 Current SPI effort

In 1989, an SEPG was created at the Phoenix site to coordinate site-wide process improvement efforts. The SEPG has fluctuated between 4 and 11 members. In addition to

these full-time members, some of the effort of the development organization, including one full-time person, was allocated to software process. A technical education board (TEB) was also created to facilitate continuing technical education and technology transfer.

The larger corporate environment into which these efforts at the Phoenix site were introduced could be described as hostile, since there was consistent and severe downsizing. Nevertheless, SPI was visible and provided a clear objective for everyone. Executive management was given several related goals, including achieving the repeatable maturity level and ISO 9001 certification. In order to move toward these goals, training time for personnel was increased to about 67 hours per person per year, and the SEPGs were funded at a level of about 2% of development budgets. Process definition and an active measurement program were initiated.

In April 1990, the Phoenix site began an inspection program as part of its standard development process. The initiative came initially from a member of the technical staff, with support from his manager and funding from the TEB. This program included inspection training for all technical personnel, as well as many managers, and extensive collection of defect data for a major project. Design documents as well as code were inspected. The success of this effort was very important in generating management buy-in for other kinds of process improvement. In fact, the savings from inspections can be viewed as funding the development staff contribution to further process improvement activities.

The Phoenix site had two SEI-style assessments. The first was in September 1990, and indicated that the organization was functioning at the initial level. About two years later, a CMM-based assessment was conducted and showed that the organization had progressed to the repeatable level, with significant progress toward satisfying key process areas for the defined level.

The most recent SPI efforts at Phoenix have two focal points. First is establishing the capability maturity model as the source of goals for the SEPG. The other focal point was a software engineering process architecture (SEPA), which is a set of architectural guidelines for software process, supported by templates. The results that follow were taken from the Phoenix site.

4.1.4 Results from Current SPI Activities

Productivity. One way in which the organization enhanced productivity as a result of SPI efforts was to base process decisions on a comparison of baseline data and data collected on the ongoing project. On one project, for example, some code originally written in one language had to be rewritten in another, for efficiency reasons. The team initially decided to skip inspections, but when they began unit testing they discovered the mean time to find and fix a defect was about four times higher than the mean time it had taken with inspections in past projects. So they reconsidered their decision, and inspected the code, with considerable savings in effort. Having relevant data readily available was sufficient to allow developers to take the initiative and make an informed decision.

Another example of basing an important management decision on an analysis of data from the current project was a case where unit test was skipped, at a savings of 1.5 person years. In this case, an inspection revealed a defect rate which was about half of what had been experienced in similar projects in the past. In order to determine if the inspection was simply failing to find existing defects or if the code really was nearly defect-free, unit test of a small sample of modules was performed. No defects were found. Additional investigation revealed a well-structured table-based design, which lent additional credibility to the possibility that the new code was in fact nearly defect-free. On this basis, they decided to forgo unit test of the remainder of the modules. This turned out to be a good decision. To date, the software has been through more than a year of integration and system test, and of 302 defects discovered, all but 8 were discovered prior to test.

The organization has estimated its savings from inspections. The estimate is based on the very conservative assumption that removing a defect costs twice as much to remove in any given stage as it does in the previous stage. So, for example, a high-level design defect costs twice as much to remove if it is not discovered until coding as it would have cost to remove had it been discovered in low-level design. If it is not discovered until unit test, the cost doubles again. The estimation procedure also makes the reasonable assumption that 50% of all defects that existed when a stage was entered are removed by inspections during that stage. This estimation procedure indicates that at least 1.2 million dollars are saved annually by inspections of requirements and design documents. Savings from code inspections are easier to quantify, and are estimated at 2.3 million dollars.

Schedule. Because of headcount changes, many cycle times were difficult to quantify. Of those that could be quantified, some test cycles were reduced by half. Design cycle times tend to be longer, but times for code and test are shorter.

Quality. There have been significant gains in quality as a result of the SPI program. There has been about a 7-10% reduction per year in defects reported by customers (see Figure 9).

Figure 9. Mean Monthly Number of Customer-Reported Defects for Six-Month Periods

Since inspections of fixes were instituted, there has also been a 50% reduction in the number of fixes that were themselves defective and had to be replaced. Interestingly, the effect is larger than one might expect, given that only about 13% of the inspections detected any defects. The additional improvement must be attributable either to the maintainers working

more carefully because they know their work will be inspected, or to lower defect injection rates caused by other facets of the process improvement effort.

There are also some current data from two recently released products. Project A, which had about 30 KLOC, has been in use about 5 months, and so far has had only one customer defect found. Data from previous products of similar size that have received comparable amounts of use by customers would lead one to expect about 25-30 defects to be discovered in this time period. Project B, which had 56 KLOC of which 25K was reused code, has been in use for a year at a number of sites, and only 17 defect reports have been received. Directly comparable data from previous systems are not available, but a previous similar product had defect report *backlogs* of 200 trouble reports.

Business Value. Based just on the savings from inspection, calculated as outlined above, and the cost of the SEPG, the organization calculates about a 4:1 return on investment. Any other benefits of the SPI efforts of the SEPG are as yet unquantified, and therefore are not reflected in the business value ratio.

4.1.5 Lessons learned

General SPI lessons

- Process improvement is a long-term effort which requires leadership and long-term commitment from executive management. Middle management must stay involved, and the technical community must buy in. Generating observable results is very important in motivating and sustaining interest.
- Substantial improvements can be made even in a hostile environment; these efforts took place amidst severe and consistent downsizing.
- When the development staff has ready access to their data, they make the right decision themselves without the SEPG/QA people looking over their shoulder.
- Define the measures very carefully. Take particular pains to avoid using terms that have been used differently in the past or are loaded with unintended meaning. An example is the word "severity" which was used to describe defects that are visible to the user. Engineers may be reluctant to own up to "high severity" defects, even though no particular onus is intended.
- Make sure the definitions of measures are actually followed.
- There is often a tradeoff between accuracy of data and the effort necessary to collect the data. In order to get enough data to be potentially useful, it may be necessary to sacrifice some accuracy. An example is having secretarial staff enter inspection data into the database. Since they do not have a high level of technical expertise, they are unlikely to catch as many data-entry errors as technical staff, but this may be an unavoidable cost to get data entered at all.
- There is likely to be a drop in the effectiveness of SPI efforts when moving from a pilot project to wider use. This can be combated with the use of data to demonstrate the effectiveness of the techniques and motivate others to give serious effort to implementing the programs.

Inspection lessons

- Inspections by themselves are not a silver bullet. They do not by themselves overcome serious flaws in the software engineering process.
- Inspect *before* unit test. The inspection team is more motivated because they expect to find defects, and smaller code segments can be inspected. In addition, it is occasionally possible to skip unit test altogether, and inspections may uncover design defects that could slip through unit testing and therefore be much more expensive to detect and fix.
- It is important to inspect requirements and design documents as well as code. Otherwise, design defects will often not be discovered until system integration test.

4.2 Hughes Aircraft

4.2.1 Organization Description

The Software Engineering Division (SED) of Hughes Aircraft is part of Hughes Ground Systems Group. It is the largest dedicated software organization in the Ground Systems Group that provides contract support for many other divisions. The SED primarily focuses on US Defense Department contracts. Roughly 500 professionals are employed at Hughes SED. Of these, 41 percent have 10 to 20 years experience in software development and 12 percent have 20 or more years experience.

4.2.2 History of SPI Efforts

Prior to 1987, the SED had established several policies and practices aimed at improving their software process. These included technical and project review processes. The technical review process specified criteria, data collection procedures, and review reporting. Also, each project was required to produce a plan for evaluating the quality of its output. The technical review process as implemented, however, was inconsistent. A policy for required training was also attempted, but since it was implemented as a promotion requirement, it ran afoul of equal employment opportunity considerations.

The project review process, on the other hand, was very well defined, institutionalized, and supported by rigorous data collection and reporting standards. As early as 1973, SED began using rate charts to track schedule adherence. By 1983, work unit completion was tracked against budget with calculations of earned value, a cost performance index, and a schedule performance index. By 1987, product measures such as software size trend, defect density, and trouble report resolution status, as well as financial data such as earnings and return on sales, were collected and used as well. Data were reported monthly to senior management and used in a quantitative process management sense to keep projects on track. All of this was done in accordance with clearly specified reporting standards.

As part of a total quality management effort that began in 1985, error data were collected, categorized into types, and used to identify areas in need of improvement. Each project used

its own format and collected data in its own way, so the data were of limited use above the project level. These efforts were rather haphazardly implemented and lacked a central focus. In 1987, as part of the continuous measurable improvement activity, SED analyzed the data from the disjointed project efforts and used the results of this analysis as lessons learned. This formed the basis for a clearly defined data collection and reporting process and for the creation of an automated tool, the quality indicator program, which has been used to collect and analyze data for over five years.

4.2.3 Current SPI Activities

In 1987, the Software Engineering Institute participated in the first process assessment of the Software Engineering Division (SED) of Hughes Aircraft in Fullerton, California. This assessment found Hughes' SED to be a level 2 organization. The first assessment yielded six findings:

- Formalize technology management.
- Standardize and centralize the collection and reporting of defect data.
- Fill gaps in the training program, including project management, internal reviews, requirements writing, testing, and quality assurance.
- Standardize the review process.
- Define the software engineering process.
- Strengthen software quality assurance.

The success with which these findings were addressed was apparent in 1990, when a second SEI-assisted assessment found the SED to be a strong level 3 organization [Humphrey 91] with many activities in place to take it to levels 4 and 5. The actual cost for SED to go from strong level 2 to strong level 3 in a two-year period was 75 person-months plus the cost of the SEI-assisted assessment, which was $45,000. Improvements included the formation of a process group, key training actions, and a comprehensive technical review process. SED also began to expand their use of data in a number of ways. Defect data were summarized across many projects, and control limits were established to highlight process quality problems as early as possible. Pareto analysis of defect types helped to identify the most costly defects, and root cause analysis was used to identify and solve the process problems that gave rise to them.

The 1990 assessment concluded that Hughes had achieved a strong position of software process leadership and had established the foundation for continuing process improvement. Although the findings and recommendations are pertinent only to the SED in Fullerton, Hughes has capitalized on this experience to launch a broader process improvement effort.

In 1990, six basic findings were identified during the second assessment:

- Integrate software into systems engineering.
- Fill gaps in computer-aided software engineering (CASE) technology.
- Expand scope of quantitative process management (QPM).
- Optimize QPM to business goals.

- Increase participation in software requirements.
- Ensure adequate software quality assurance (SQA) support.

These findings were addressed in an action plan which covered a period of 18 months and included 65 milestones. As an example of how specific findings were addressed, SED addressed its CASE technology needs by creating a technology transfer group, and assigning it the responsibility for evaluating commercially available tools and disseminating that knowledge throughout the organization. They were also asked to undertake project needs assessments for CASE tools and to oversee the transfer of tools to the projects. In addition to creating this entity with these general responsibilities, SED also generated very specific requirements for its case tool solution. Among these requirements was maintaining a file of promotional material from CASE vendors, publishing a CASE newsletter, and automating recurring manual tasks such as unit testing. Each of the findings was addressed in a similar manner.

SED underwent a self-assessment in May, 1992. Using CMU/SEI-87-TR-23, the assessment resulted in a level 4 maturity rating. The assessors noted, however, that using the newer version of the CMM, SED would be at maturity level 3. The findings from this assessment were as follows:

- Predict and track errors.
- Analyze error data for root causes.
- Continuously improve the efficiency of reviews.
- Institute a process for prototyping.
- Develop a technology assessment mechanism.

The action plan resulting from this assessment covered the following 12 months, and contained about 20 milestones. A good example of the kinds of actions taken is the plan to analyze error data. A goal of developing the ability to predict the quality (i.e., post-release defect density) of the final product was established, and specific projection and data collections activities to achieve this goal were designed. At the start of each major phase, error density is to be projected out to the end of the project. As actual defect detection data are gathered, a profile of the projections and actuals over time will be made for each project. These data will be used to help determine the "health" of future projects, and whether extraordinary measures are warranted. Finally, this high-level design is further specified in terms of what specific actions will be taken, and when and by whom. The total projected effort for this action plan was 714 hours, or approximately 4.5 person-months.

4.2.4 Results from Current SPI Activities

The benefits Hughes was able to quantify are reported below. Many of the most important benefits were extremely difficult to measure with precision. Among these hard-to-quantify benefits that SED has experienced are:

- Predictability based on facts and data has a multiplier effect on overall project quality.
- Team spirit, pride, and morale have increased.

- Technology is easier to insert.
- Language, design, and models are simpler to change.
- Working conditions have improved, e.g., fewer overtime hours, lower stress, and fewer “gut-wrenching” problems to deal with.

Productivity. There are several types of data which show that the level 3 process has substantially improved productivity as compared to the previous level 2 process. Early detection of defects is an important contributor to productivity, since it can substantially reduce rework. Figure 10 shows the percentage of coding defects found in each phase for the old process versus the new process. The "new process" figures are from an actual project in 1992-1993 that implemented new coding activity guidelines designed to prevent defects and to detect them earlier in the software development life cycle. These guidelines applied to coding, peer reviews, rework, and formal review processes. The "old process" figures came from similar projects in the central organization-wide defects database. While we do not have a dollar estimate of the savings from early detection of defects, it is safe to say that the savings to the project from the enormous change shown in this figure are substantial.

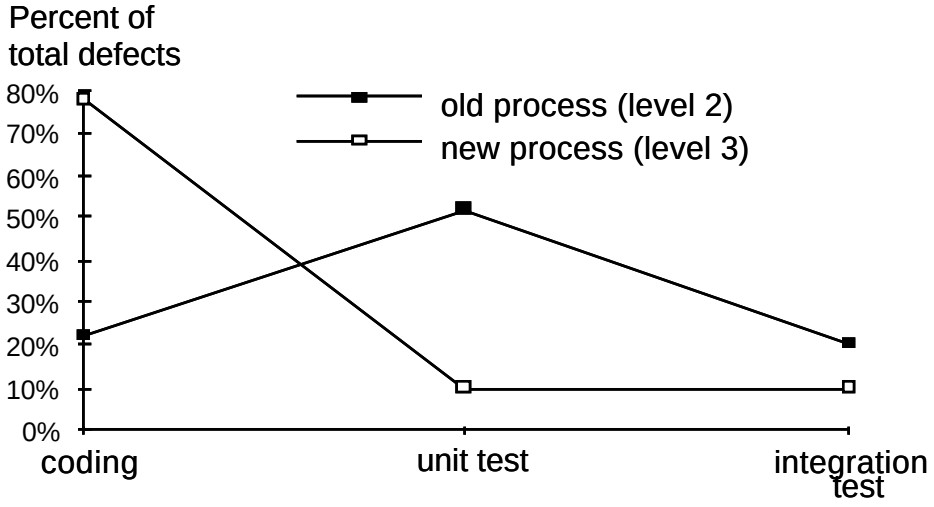

Figure 10. Percentage of Defects Found by Stage

Figure 11 shows a substantial reduction in the amount of rework performed. This is one of the effects one would expect from the earlier detection of code defects shown in the previous figure. But notice that there is substantially reduced rework in precoding stages as well, indicating that other facets of the SPI effort are also having an impact on rework. Obviously, the figure indicates a substantial savings.

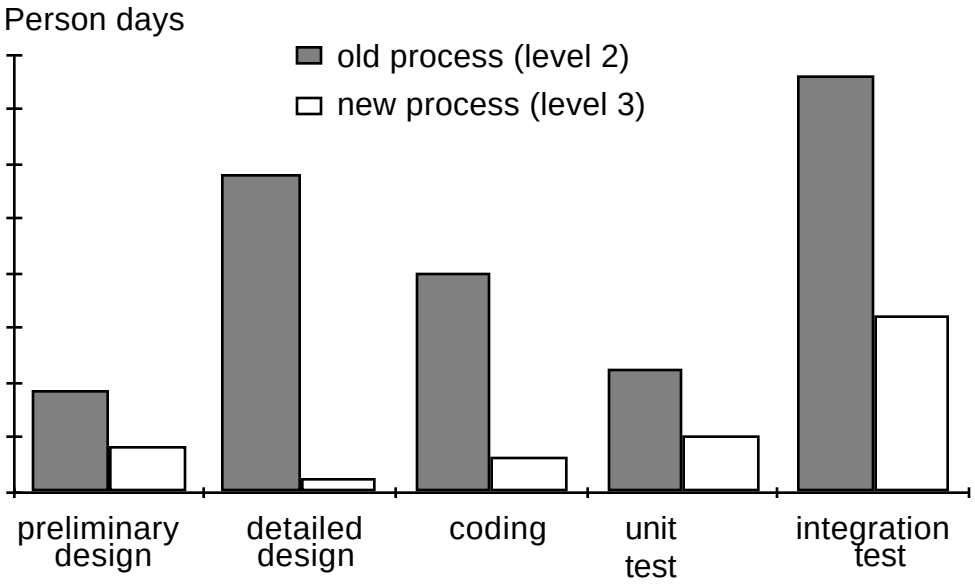

Figure 11. Rework for Each Stage

In addition to the actual reductions in cost of rework, there was a substantial improvement in the accuracy of overall predicted costs. Figure 12 shows how the cost performance index (ratio of budgeted cost of the work performed to the actual cost of the work performed) has improved over time. Notice that in the last two years, the average actual cost has been lower than budgeted cost.

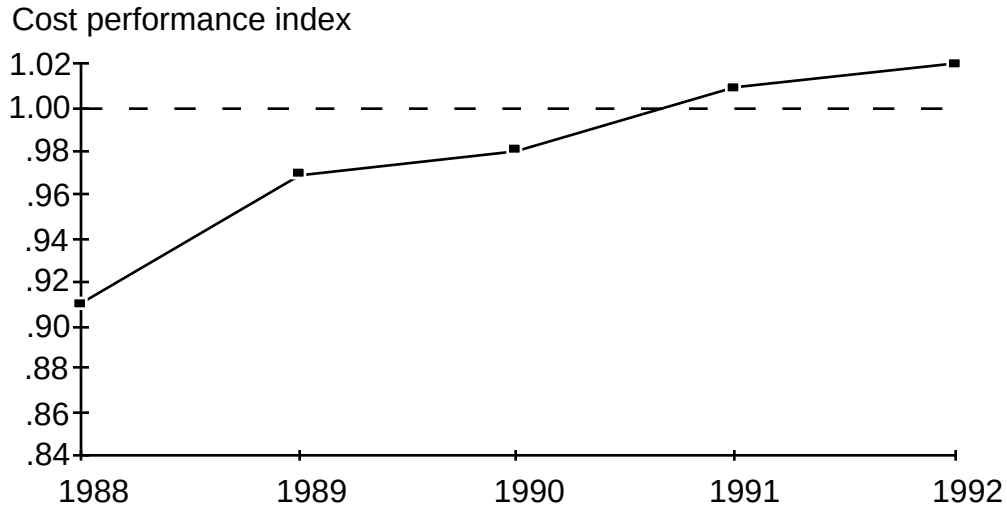

Figure 12. Cost Performance Index

Schedule. Improvement in meeting schedules is shown in Figure 13. Here, the schedule performance index is plotted over time. Notice that the first year of process improvement produced a modest dip in this index, while it has improved steadily in each subsequent year.

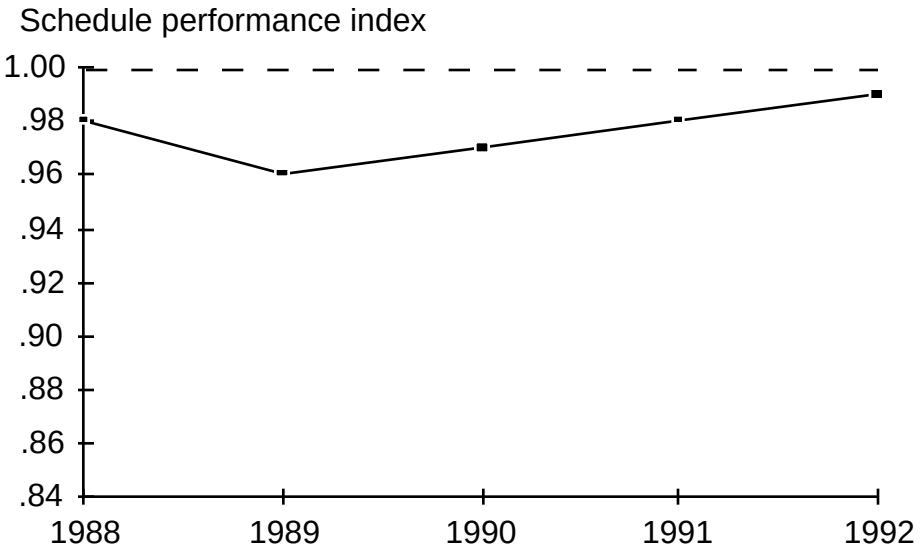

Figure 13. Schedule Performance Index

Business Value. First year benefits of about $2 million, versus cost of about $400,000, plus the assessment cost of $45,000 gives a business value of about 4.5 to 1.[6] Of course, as the benefits continue to accumulate and the up-front investment costs are spread over additional years, the ratio should continue to increase.

4.2.5 Lessons Learned

- Management commitment to software process improvement is critical to its success.
- Developing an action plan is essential.
- An organizational focal point for improvement efforts and a focal point for technology transfer are keys to success.
- Many of the most important benefits of SPI are intangible such as pride, esprit de corps, quality of work life, company image, and the emergence of a coherent culture. Given their importance, it would be very useful to develop measures of these things.
- As compared to the past, there are very few crises at SED since process improvement efforts have been implemented.

[6]This figure differs somewhat from the published figure [Humphrey 91]. In order to make this figure comparable to others in this paper, we also included the cost of an assessment. The result is a figure somewhat lower than the published 5:1 number.

4.3 Schlumberger

4.3.1 Organization Description

Schlumberger is an international company employing over 53,000 people in 100 countries. Included among Schlumberger's business are

- wellsite exploration and production services,
- testing and electronic transaction products, and
- metering products.

Over the last decade, software has increased dramatically in importance at Schlumberger. The company now calculates software is responsible for the major share, i.e., 50% to 100%, of the engineering investment in many of its products. Most of our data come from one particular Schlumberger organization, we will call it organization A, which is located in the United Kingdom.

4.3.2 History of SPI Efforts

Various process improvement and total quality management (TQM) initiatives had been undertaken across several Schlumberger organizations in recent years. A very broad TQM effort was initiated in organization A near the end of 1988. As part of these activities, a process group was set up to improve software process. The initial goal was to reduce defects in shipped code to less than 0.1 per thousand lines of source and to improve compliance with the organization's product development procedures. The procedures that existed at that time tended to be oriented toward hardware development. This initial goal was soon redefined and broadened; the new mandate was to "improve the quality (i.e., conformance to requirements) of software and simultaneously improve productivity."

While the TQM efforts were regarded as successful in some areas, e.g., manufacturing, they had relatively little impact on software development. The generic goals of improving quality and productivity did not seem to provide a sufficiently clear focus for planning activities to improve software development and maintenance.

4.3.3 Current SPI Effort

In 1989, Schlumberger established the Schlumberger Laboratory for Computer Science (SLCS), in part, to help boost quality and productivity. Techniques based on the SEI capability maturity model for software were adopted as the framework for improving software development and maintenance.

In order to accommodate resource constraints, and because of the small size of their development organizations, Schlumberger modified the standard SEI assessment procedure. Two to five representative projects were selected, and the maturity questionnaire was administered. The follow-up interviews were performed by one member of the SLCS staff over a period of one to two days. The results of these *studies* (as Schlumberger calls them

to distinguish them from the standard SEI-style *assessment*) were sets of 8-10 findings that identified the most critical areas in which improvement was needed at each site. These findings were based on common observations across the sampled projects.

The initial round of assessment activity involved 20 organizations, with about 10% of the 2000 developers in these organizations participating in interviews. About 70% of Schlumberger's software developers reside in one of these 20 organizations. In terms of maturity level, the results of these assessments were typical of the industry in general at that time. According to these studies, the pattern of results was fairly consistent across sites. The areas they determined to be most in need of improvement were

- project management,
- process definition and control, and
- project planning and control.

In addition to immediately initiating improvement efforts in these areas, it was also recommended that development organizations

- track deliverable size and effort,
- gather error data,
- invest in resources for improvement activities, and
- review how project commitments are made.

Action on these findings often took the form of establishing improvement teams in the organizations. Some were populated with full-time budgeted personnel, while others were put together on a part-time volunteer basis. These teams were supported by the corporate-level group in SLCS both by frequent visits and by electronic communication. In particular, the SLCS group helped to develop a set of process documents with each organization. A major training effort was also initiated, focusing primarily on in-house classes on software project management and peer technical reviews. SLCS also undertook a "technology watch" function to evaluate and publish findings on tool innovations such as defect tracking and CASE tools. A bulletin board was established, and postings are made from Schlumberger sites around the world. Finally, there was a major effort to encourage collaboration, across areas of expertise and among business units.

Schlumberger has now followed up the first round of studies with about 10 follow-up studies. These were done in a way that much more closely approximates an SEI-style assessment. However, they use a four-person team and a three-day schedule. The major changes seen in this second round were

- Most large and many small centers have and follow a written process document.
- Most sites have improvement teams.
- Geographically separated groups with similar product life cycles are sharing the same software process.
- All organizations that set improvement goals have moved up the SEI maturity scale.
- Communication, both between departments within an engineering center and between centers, has improved.

4.3.4 Results from Current SPI Activities

As noted earlier, our data from Schlumberger come almost entirely from one organization, which we are calling organization A. It was one of the original 20 for which an assessment was performed. It had already been ISO 9000 certified, and as mentioned above, a TQM initiative was already underway. While this organization provides the bulk of the data, we provide examples of data from other organizations where they are available.

Productivity. Although there is considerable variability in the data, based on a linear regression of the data points, organization A experienced approximately a 30% increase in productivity as a result of software improvement efforts, as shown in Figure 14. The cost is calculated as 1988 dollars per effective lines of code (ELOC). ELOC is an estimate of physical lines of noncomment source code, counted as if there were no reuse. It was estimated from cost data and from estimates of the proportion of software content in the project. Other, more subjective, criteria were also used in some cases. In Figure 14, the costs for each 6-month period are averaged across projects defined (i.e., a specification was agreed upon and approved) in 6-month periods.[7] Caution is needed in the interpretation of these data, since the numbers can be heavily influenced by the extent of reuse.

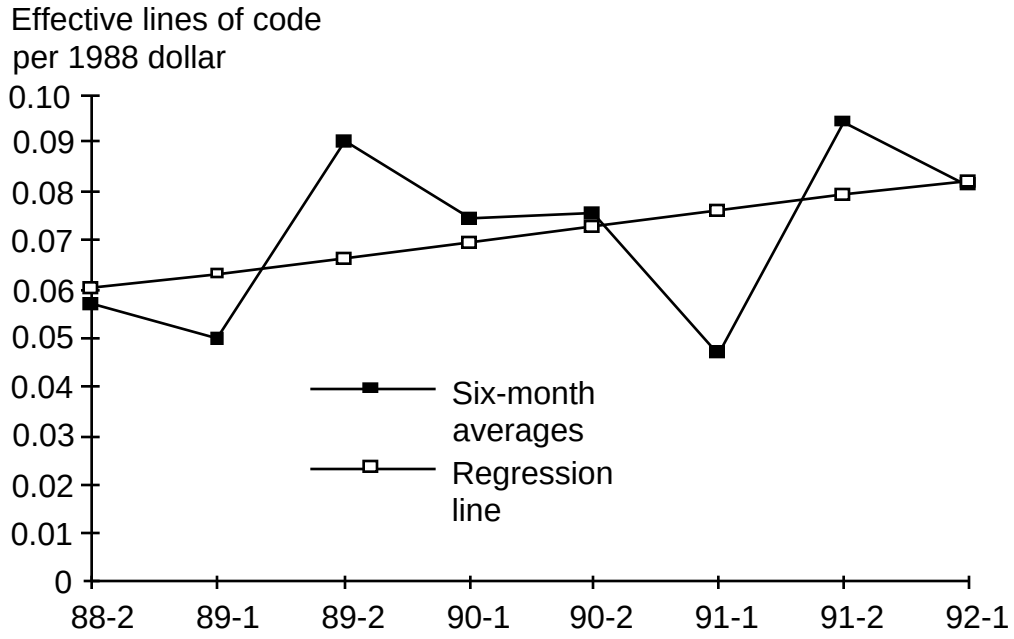

Figure 14. Effective Lines of Code per 1988 Dollar

There are also examples of improvements in productivity from other organizations in the company. Prior to the introduction of requirements management techniques, one group that works on complex embedded real-time systems took 34 validation cycles (code, test, recode) in order to satisfy customer requirements. After introduction of requirements management procedures, the next product, which was a similar system, took only 15 validation cycles.

[7] The regression line was fit using the least squares method.

Productivity, as measured by noncommented source statements (NCSS) per person-month, nearly doubled, due to the reduced effort for validation.

Schedule. Organization A experienced substantial improvement in adherence to schedule. Figure 15 shows the average progress rate (APR) for projects that were defined (i.e., a specification to do the work was agreed upon and approved) for each 6-month period. APR is the average, for all projects underway, of the ratio of planned project duration, based on the originally planned completion date, to the currently predicted project duration, based on the current estimated completion data. As APR approaches 100%, the schedules are being met. As the figure shows, organization A went from approximately a 55% APR to rates in excess of 90% for the last year.

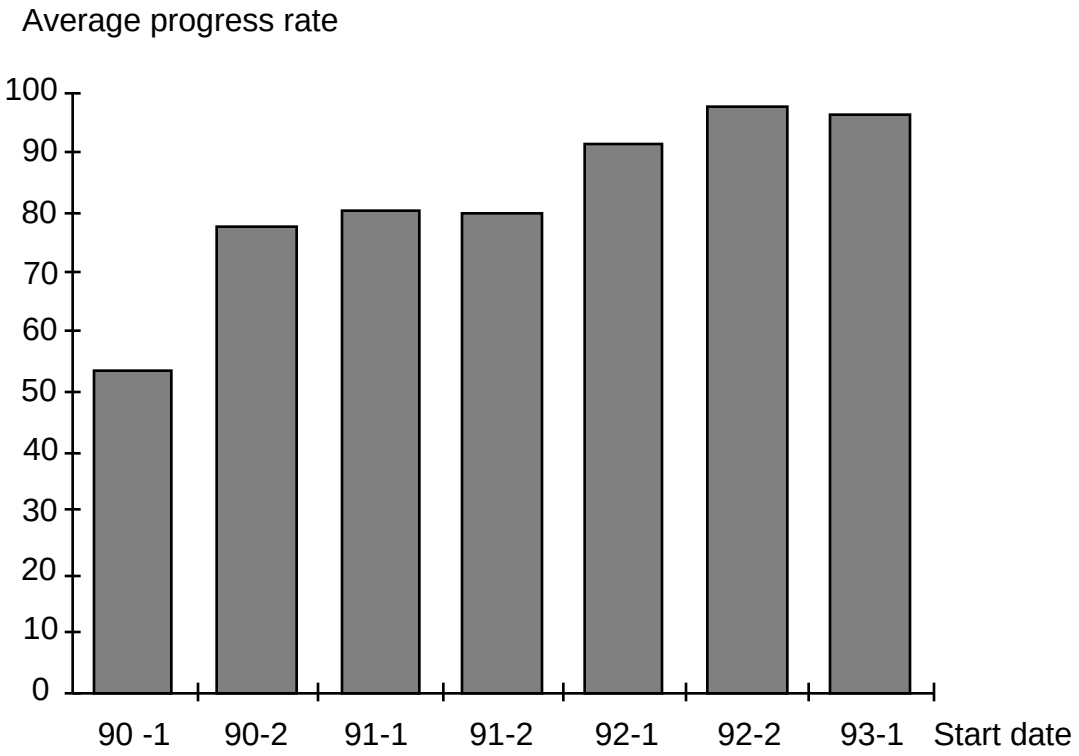

Figure 15. Average Progress Rate for Projects by Definition Date

The difference in actual versus scheduled completion dates for finished projects is shown in Figure 16. This figure shows the average difference between estimated and actual completion times (in weeks) for projects defined (i.e., specification agreed upon and approved) in each 6-month period.[8] The improvement shown in this figure is startling. The difference goes from nearly 35 weeks to -1 week. In other words, the average project signed

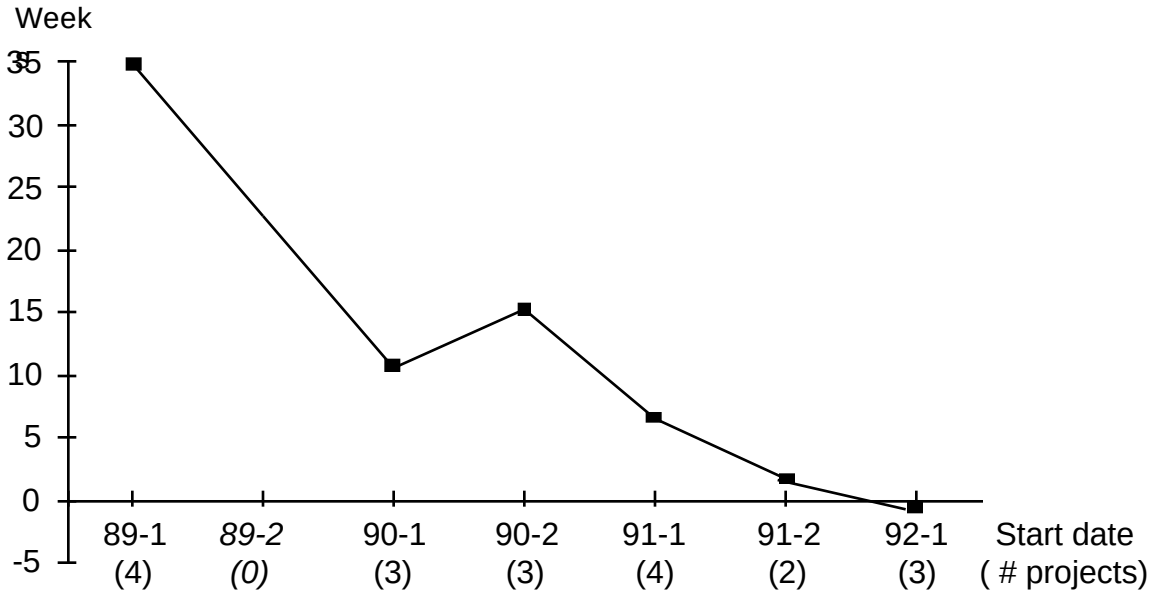

Figure 16. Average Difference Between Estimated and Actual Completion Times

off in the first half of 1992 (the last period for which project completion data are available) finished a week early. An examination of project parameters for this period indicates that this change was brought about primarily by work being completed faster than before. Schedules and project size throughout the period remained fairly constant, so the change does not seem to be the result of padding the schedules or taking on simpler projects.

Quality. Detailed data are not available, but organization A reports a decrease in defects from 0.21 per KLOC to 0.14 defects per KLOC during the period of 1989 to 1992. This is close to the original goal of 0.1 defects per KLOC established in 1988.

Business Value. A rough estimate of business value for organization A was obtained as follows. Over the 3.5 years of process improvement efforts, the organization estimates that total resources of about $170,000 were spent on software process improvement. A conservative estimate of benefits was computed as follows. The software engineering budget was approximately $2.5 million in 1988. We can calculate a savings for each six-month period which is equal to the productivity gain for that period times one-half the 1988 annual budget. From this, we can estimate the dollar savings for each six-month period. The accumulated savings divided by the cost given above generates an estimated business value

[8] No projects were started in the second half of 1989.

of 8.8:1. The figure is rough and very conservative, since it leaves out other potentially large benefits such as market advantage from shortening development schedules.

4.3.5 Lessons Learned

- Software process improvement requires a cultural change that should spread throughout the business. It is very important to coordinate the software process improvement activities with other parts of the business. Software people interact heavily with marketing, hardware development, sales, manufacturing, and others.
- Attention must be given to getting the support of middle management. They are often caught in a bind in which they are supposed to implement significant changes without missing pre-existing deadlines. This makes them less supportive. One possible solution that SLCS will try in the future is to work with middle managers first, so that senior management can be given an accurate picture of the time and resources required.
- Subsequent assessments were more difficult than initial assessments. The engineers were well coached with the "right" answers, and it was very difficult and time-consuming to find out the “true” answers and to generate useful findings.

4.4 Texas Instruments

4.4.1 Organization Description

Texas Instruments Incorporated (TI), headquartered in Dallas, Texas is a high-technology company with sales or manufacturing operations in more than 30 countries. TI products and services include semiconductors, defense electronic systems, software productivity tools, computers and peripheral products, custom engineering and manufacturing services, electrical controls, metallurgical materials, and consumer electronics products. This particular case study originated within a computer products and services group located at TI in Austin, Texas.

4.4.2 History of SPI Efforts

In 1981, a division of the Defense Systems & Electronics Group first developed a “software methodology” for voluntary use among projects. The local SPI/Software Quality Assurance Group maintained this document and provided consulting on demand from initial users of the document. As such, the “software methodology” and other such efforts grew during the 1980s from different grass-roots efforts. In 1988, the computer products and services group in Austin obtained a copy of the “software methodology” to use on their internal development efforts with assistance from their local SPI group. By this time, the widespread voluntary use of the “software methodology” was so great, that up to five releases of the document had been released with about 500 copies in circulation.

4.4.3 Current SPI Effort

Recognizing the significance of communication and leverage of the various SPI products and services, TI created the STEP corporate software process group in 1989. With thousands of software engineers and various local SPI groups within TI, the first task of the STEP group was to develop a company-wide preferred software process that supported tailoring to meet local business/process needs. This preferred process had its roots in the "software methodology" which formed the basis for this case study. Since then, the joint efforts of STEP and local SPI groups have accelerated TI achievements. TI's Defense Systems & Electronics Group (DSEG) has recently been rated at the defined maturity level using the SEI's new internal process improvement (IPI) appraisal method. [IPI is the evolved form of the older software process assessment (SPA) methods.]

4.4.4 Results from Current SPI Activities

This case study compares two projects which were very similar in all respects except for the software process used. Project A developed software for customer support. It was developed in an ad hoc manner, with little time spent with users to nail down the requirements. In fact, most of the requirements were not written down. Project B produced a system for customer representatives who were taking change requests, providing customer support, and screening service requests. There was much overlap in the functionality of the two systems being developed, in that both were oriented around databases and had extensive reporting functions. Both were fairly small, i.e., in the 25-50 KSLOC range. The teams developing the systems were identical, i.e., the same people but using different methods. Project B called in the local SPI group and followed their advice, including definition of the requirements, testing the requirements with users, and inspecting the design and code. The inspections were modified Fagan inspections and represented very structured walkthroughs. This gives us an opportunity to compare two projects which were very similar in all ways except the process used.

Productivity. Figure 17 shows the enormous reduction in effort experienced by Project B.

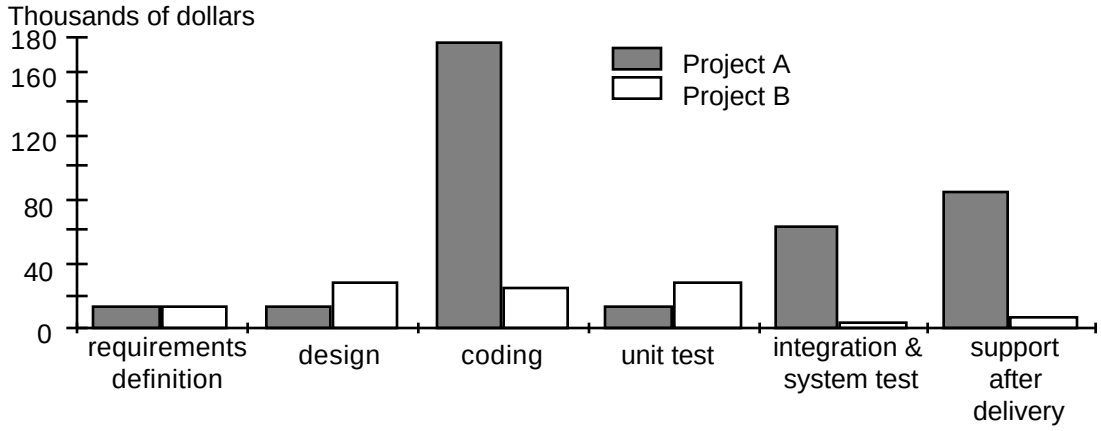

Figure 17. Resources Spent in Various Stages of Two Projects

Figure 18 permits normalized comparisons of where the costs were incurred, by plotting expenses in each stage as a percentage of total expenditures for each project. The savings show up primarily in coding, integration testing and support after delivery. In requirements

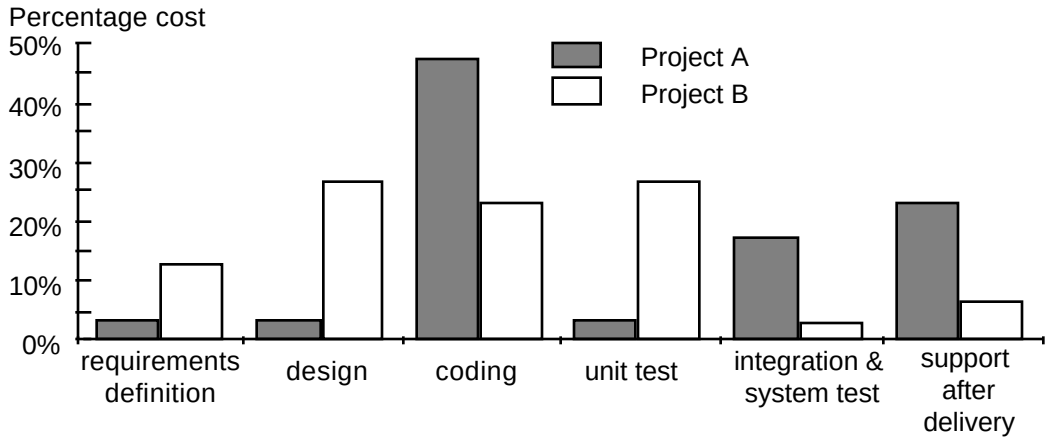

Figure 18. Percentage of Cost in Each Life-Cycle Stage for Two Projects

definition, design, and unit test, Project B was as costly or more costly that Project A. The overall savings, however, was significant. The cost per SLOC for Project B was only $3.60 as compared to $10.20 for Project A. Many factors may have contributed to this outcome. In particular, it appears that the emphasis on managing the requirements and reviewing them with the customer seems to have been a worthwhile investment. In Project A, two-thirds of the modules needed rework, mainly because the requirements were not managed effectively. It also seems very likely that the peer reviews helped reduce the costs of integration testing and support. We must acknowledge, of course, that this is only a comparison of two projects, so other effects, such as learning from the first project, also figure into the results. However, the distribution of effort across phases clearly indicates the focus on upstream activities and the areas of process improvement. (Also see the Hughes case study for a similar result.) The enormous savings suggest very positive results for these software process improvement efforts. The huge difference in the number of requests for design changes (Figure 19) shows the value of working with customers to ensure that the product addresses their needs.

Quality. Project B also experienced substantially fewer defects than Project A, as can be seen in Figure 19. Standardizing these numbers on size, Project A had 6.9 defects per KSLOC, while Project B had only 2.0 defects per KLOC. The defects in Project B were also less costly to fix. On average, a fix in Project A cost $706, while a fix for Project B cost $206.

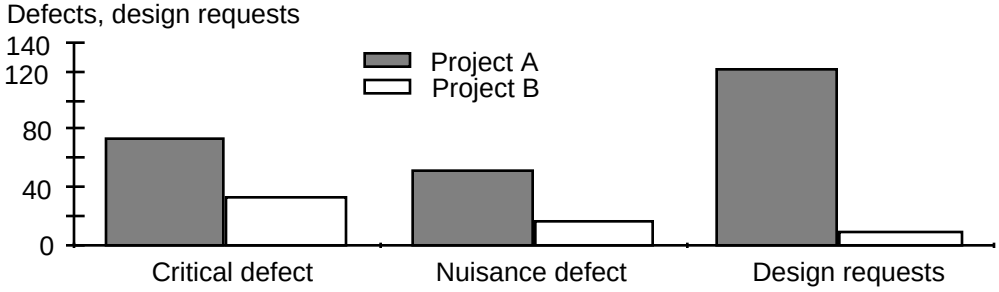

Figure 19. Defects and Design Requests for Two Projects

4.4.5 Lessons Learned

- Software process improvement depends on the critical role played by local SPI groups as supported by the project leader. This arrangement is the most sensitive to organizational and business needs.
- Local SPI groups can accelerate results by sharing and helping each other.
- The grass roots effort across TI related to the “software methodology” underscores the power of voluntary, cooperative process improvement. The nature of this document allowed it to survive many generations of organizational changes and changes of document owners.
- With multiple organizations contributing funding to maintain a living document, the “software methodology” has maintained its audience. It has also saved substantial resources compared to the case where each business group constructs a method on their own.

4.5 Oklahoma City Air Logistics Center, Tinker Air Force Base

4.5.1 Organization Description

The United States Air Force's Oklahoma City Air Logistics Center's (OC-ALC) Directorate of Aircraft Software Division is located at Tinker Air Force Base which is near Oklahoma City, Oklahoma. The software division is divided into 9 branches and employs approximately 400 people engaged in developing and maintaining software. Due to different types of workloads, the software division has divided their process improvement efforts into 2 parts. This case study will discuss the efforts in the Test Software and Industrial Plant Equipment Branches which are composed of over 180 personnel divided into 4 branches. Two of these branches develop and maintain test program sets (TPSs) which are used with automatic test equipment (ATE) to provide repair capability for complex avionics. The Industrial Plant Equipment Branch provides software support to jet engine, constant speed drive, and eddy current testing along with several automated processes associated with jet engine overhaul. The fourth branch, the Management and Technical Support Branch, provides the personnel, training, funding, drafting, computer, and other support functions. The weapon systems affected by the software products produced and maintained by the Test Software and Industrial Plant Equipment Branches include 10 different aircraft and 6 jet engines.

4.5.2 History of SPI Efforts

The OC-ALC began its software process improvement efforts in 1987 when it began defining its software processes. In 1988, the process and product enhancement team (PPET) was formed to play the role of caretaker of the process definition documents. The PPET failed to have much influence in process improvement, primarily because it lacked a defined and visible interface with management.

4.5.3 Current SPI Effort

In 1989, OC-ALC began a relationship with the SEI and had its first assessment in March 1990. At that time, the center was rated at the initial level of process maturity.

The findings from that first assessment were addressed by over 50 different process improvement efforts. They vary widely in scale, from very small, such as establishing a newsletter, to very large, such as developing an on-line maintenance tracking system. A small sampling of these improvement efforts follows:

Software maintainability evaluation (SME) process. This was originally developed to check software developed by contractors. It allowed OC-ALC to find defects quickly so that the software could be returned to contractors within the warranty period. It is now also used on software developed by the OC-ALC as a final quality check prior to delivery to its customers.

Cost/schedule control system criteria (C/SCSC). Implemented very recently, this method allows project leaders greater insight into the software development and maintenance process. They can determine, for example, how long it takes to complete a specific form. One very important consequence of using this method is the ability to tell external organizations the precise impact that they have on OC-ALC processes. So, for example, suppliers of needed assets, and interfaces with other groups such as the one controlling configuration management, can be informed in some detail of how their schedules will affect the overall development and maintenance process.

Maintenance tracking system (MTS). This system is used to track maintenance information across multiple weapon systems. Originally developed for tracking the maintenance status of the B-1B test program sets, it was modified for general use. Despite some initial resistance, it is now used throughout the organization.

OC-ALC/Aircraft Software Division library update. This library is used for storing engineering documentation, software, process assets, and so on. Various improvements to the library were implemented, based on a survey of internal and external customers.

Internal training courses. Courses in a number of areas such as test program set process, circuit simulation, and diagnostic tools were developed in-house when they were not available or difficult to obtain from external sources.

Technical order (TO) editor. This tool permits engineers to check all messages that can appear on the test operator's screen. It was provided to several contractors so that they could check and correct their TPSs prior to delivery to LAS.

In 1993 OC-ALC was selected as an alpha site by the SEI to pilot test its updated assessment method based on version 1.1 of the capability maturity model for software. OC-ALC was again assessed with the help of the SEI. This time, however, the SEI-assisted assessment resulted in OC-ALC being assessed at the repeatable level of process maturity (level 2). Today, OC-ALC has a goal of becoming an organization at the defined level of process maturity (level 3) by 1995—three years ahead of a United Stated Air Force policy requiring all Air Force software organizations to be at level 3 by the year 1998. OC-ALC is planning its next assessment for late 1995.

To fund its process improvement efforts, OC-ALC budgeted an amount equal to five % of the total labor costs for the organization. Two senior engineers are assigned full time to process improvement, and the remaining budget (seven staff-years per year) is allocated to specific improvement activities. All improvement activities are scheduled, funded, and tracked similarly to all other projects within the organization. OC-ALC personnel have commented that having a budget of five % of the total labor costs is significant and puts the process improvement efforts at the same level of management attention as other key projects within the organization.

Personnel chosen to work on process improvement activities at OC-ALC are chosen for their knowledge, skills, and enthusiasm. By having many of its personnel, including key, senior level engineers, work on improvement activities, ownership and responsibility is spread across the organization.

The software process improvement infrastructure at OC-ALC consists of a management steering team (MST), a software engineering process group (SEPG), and technical working groups (TWG). The MST is composed of division and branch management and the SEPG chairman. It has management and oversight responsibilities, including establishing priorities for improvement activities, assigning personnel, providing resources, and monitoring status. The MST has monthly meetings open to all personnel.

The objectives of the SEPG are to identify areas needing improvement, make recommendations, formulate action plans, and track progress and effectiveness of the improvement efforts. The SEPG, composed of both full and part-time personnel, meets biweekly. The SEPG meetings serve as an informal forum for OC-ALC personnel to share their ideas and raise issues that they see affecting the organization.

Technical working groups (TWGs) are established as needed to work process improvements in specific areas. TWGs are composed of technical experts for specific areas and membership may cross organizational boundaries. Each TWG is required to develop a charter defining the overall and specific objectives of the TWG, membership, voting methods, meeting frequency, and the length of time it will exist.

It is important to note that TWGs are not established for every improvement effort, just those that need to be worked on by a group. Individuals may be assigned to work some of the smaller improvements. Many improvements have originated within sections and been spread throughout the organization with the aid of the MST and the SEPG.

Current activities regarding software process improvement at OC-ALC are focused toward addressing the findings from the March 1993 assessment. The findings concerned level 3 issues along with some organization-specific issues. In addition, the physical facilities and non-process external dependencies were noted as tough problems.

4.5.4 Results of Software Process Improvement Activities

Business Value. Tinker OC-ALC has implemented over 50 different process improvement activities since 1989. Since that time OC-ALC has invested more than $1,070,000 in its process improvement activities. It estimates that it has saved over $4,792,527 as a direct

result of its process improvement activities. These savings are primarily cost avoidance, such as rework and duplication of function. Overall, the ratio of savings to cost shows a return of more than 4.48 dollars on every dollar invested in software process improvement. Since there is inevitably a delay between spending for an improvement and realizing the payoff, OC-ALC has also calculated a business value figure that excludes the amount spent for work in progress that has not yet had an opportunity to generate a benefit. Of the $1,070,000 spent, $305,254 fits in this "in progress" category. Calculating a business value ratio that excludes this amount gives a figure of 6.27 to 1. Table 3 shows a sample of costs and savings for some of the SPI activities at OC-ALC. Notice that the software maintainability evaluation process and the maintenance tracking system together account for about 25% of the total savings.

Improvement Effort	**Cost to Develop**	**Savings (3 years)**	**Business Value**
Software maintainability evaluation (SME) process	$9,280	$900,000	97 / 1
Maintenance tracking system (MTS)	$90,000	$1.2M	13 / 1
OC-ALC library update	$800	$12,000	15 / 1
TO Editor	$1,300	$109,000	84 / 1

Table 3. Financial Data for a Sample of Software Process Improvement Activities

OC-ALC collects cost data on all of its improvement activities but has not been able to measure or estimate savings on some of its specific improvement activities. For example, one of OC-ALC's first improvement activities that began in 1987 and continues today is its process definition efforts. To date, $338,500 has been spent on this activity but the savings are very difficult to quantify. As in any process improvement initiative, process definition is the foundation for improvement. OC-ALC understands this and continues to regard process definition as an important activity. Other activities are not measured in terms of their benefits or savings. One example is the organization's newsletter, produced at minimal cost of a few hundred dollars. It has been credited with boosting morale and keeping all employees informed of ongoing improvement activities.

OC-ALC also has realized that, even though it has been assessed as an organization at the repeatable level of process maturity, it needs to maintain its current organizational strengths. Recognizing that the goal is continuous improvement, OC-ALC also has ongoing improvement activities that help to strengthen those assets such as its implementation of a cost/schedule control system criteria (C/SCSC). Implemented in April 1993 (after the March

1993 assessment), the C/SCSC has taken the project tracking and planning process to a new level of expertise at OC-ALC. Project managers are now able to spot bottlenecks and tune their software processes. They are also able to provide detailed information to outside organizations concerning their affect on the organization's internal processes. Although the organization is still gathering data on savings, the cost to implement the system was only $4,222.

4.5.5 Lessons Learned

- Process improvement and the internal groups that implement it must have active and visible support from management in order to succeed. The management steering team has been the biggest single key to the success of improvement efforts. They send a visible signal that process improvement is a management concern and priority.
- Resistance, often in the form of statements such as "We're different," or "I've got real work to do," or "I'm busy serving my customer" can be overcome if it is widely perceived that process improvement is a priority for upper management. Goal setting by upper management, such as the Air Force goal of SEI CMM level 3 by 1998 for its software organizations, is a powerful way of focusing improvement efforts, motivating people, and demonstrating that the efforts are valued. Getting everyone involved in the improvement effort is another key.
- Having SEPG membership of both full-time personnel and part-time personnel drawn from various projects is very important. The full-time members provide continuity for the process improvement efforts, while the part-time members act as advisors, advocates, change agents, and communications liaisons.
- The SEPG leader must keep records and treat process improvement just like any other project. This person is key to process improvement success.
- Specific, well-defined improvement efforts, often started by one or two people, have proven easier to develop and implement. Experience has shown that addressing each finding with a number of small improvements can be a very effective strategy.
- Not every improvement effort requires a TWG. Using a TWG when one is not needed is a bit like “watering your flowers with a fire hose.”
- People must be kept aware of the improvement efforts, whether by newsletter, meetings, e-mail, or otherwise. Visibility is important to building acceptance and buy-in.
- Don't be afraid to fail. It is not possible to perfect a plan before implementation. Trying things out, modifying as necessary, and learning from failures as well as success are important tactics.
- Separate funding, rather than asking projects to support improvement efforts in their "spare time" has been critical in having process improvement perceived as "real work."
- The CMM should be used as a tool, not as an end in itself. One should not do something just because the "CMM says;" rather one should use the CMM as a way of

addressing process issues with a goal of increasing quality and productivity. This involves tailoring improvements to the organization.

4.6 Summary of Case Studies

In this chapter, we have seen five organizations that differ markedly in type, application domain, and approach to SPI. It is worth reflecting on that diversity for a moment. The organizations are commercial, DoD contractor, and military. Application domains include operating systems, embedded real-time, information systems, and test program sets. Each has its own approach to SPI, including

- An early emphasis on analysis and use of inspection data, within a wider-reaching SPI effort (Bull HN).
- A long history of process definition, data collection, and quality management (Hughes).
- Corporate-level assessment and consulting resources made available to many software organizations (Schlumberger and Texas Instruments).
- Many small, grass-roots improvement efforts coming together under an organization-wide structure to meet organizational goals (OC/ALC).

There was a similar diversity in the ways that organizations measured their progress. Some looked only at a few key measures, while others are collecting many. Some have a baseline going back a number of years, while others have begun to collect process measures only recently. Differing business strategies, of course, dictate a focus on different measures.

What we find remarkable about this collection of case studies is that, despite all these differences, each of these organizations was able to use the CMM as the basis for substantial measurable improvement. They are not merely reaching higher maturity levels, but more importantly, they are making progress toward their business objectives.

5. Technical Issues in Measuring Results of SPI

In a recent article in the Journal of Systems and Software, Fenton [Fenton 93] wrote about the current crisis in software and the myriad of proposed solutions:

> [A]necdotal evidence of significantly improved quality and productivity are not backed up by hard empirical evidence . . . and where real empirical evidence does exist, the results are counter to the view of the so-called experts.

While we have not always encountered results so unexpected, we agree wholeheartedly that intuition is no substitute for evidence. Fenton continues to note that software engineering is a difficult discipline within which to conduct experiments. Furthermore, he faults the lack of hard knowledge regarding the effectiveness and utility of software engineering innovations on "poor experimental designs and lack of adherence to proper measurement principles."

Hetzel [Hetzel 93, p 202] has commented on the difficulty of measuring software processes and comparing their results. To guide the use of measurement within an organization, he recommends a model of software engineering that focuses upon inputs, outputs, and results. This model captures the creation of the software work products and the impacts that they have on software development or the business performance of the company. He notes the problems associated with defining and measuring a process and its results. One must also be aware, however, of the influence of other organizational factors or configurations of processes that contribute to process outcomes.

Quantifying the results from investing in software process improvement is challenging. The following section addresses issues that must be confronted and addressed when attempting to quantify the impact of a particular software process improvement. It is intended for those readers who wish to pursue a deeper understanding of the technical issues involved in quantifying the results of software process improvement. For the purposes of presentation, we have divided the issues into three groups.

First, it is necessary to identify, and have the capability to measure, outcomes of interest. This involves addressing the issue of identifying measures that will indicate the success of one's business strategy, as well as carefully defining the measures and implementing procedures for actual data collection. These issues are discussed in Section 5.1.

Second, with any set of measures, the hope is that they will change in a positive direction over time. A technique for plotting some of the most important kinds of changes is introduced in Section 5.2.

Third, a causal link between the changes in software process and the measures of success must be established. This is important because development of software systems is a dynamic process in which many things besides the software process are changing. Personnel, levels of experience, the technical environment, customers, business environment, complexity of applications, and so on, all exhibit some degree of volatility, and all may influence the success (and measures of success) of software development projects.

In order to measure the benefits of SPI, one must be able to isolate the effects of process improvement from the effects of other factors. These issues are discussed in section 5.3.

5.1 Selecting Measures

Are we improving? If so, by how much? These can be surprisingly difficult questions to answer. In this section we give a brief overview of some of the issues involved in choosing a set of measures to judge the performance of a software process. We discuss the balanced scorecard, Goal-Question-Metric (GQM), CMM/SEI core measures, business value, and intangible benefits.

Balanced scorecard. Measuring an organization's performance is an important issue for all types of business units. Approaches that have worked well elsewhere may also be good candidates for software organizations. An example is the balanced scorecard [Kaplan 92, Kaplan 93] which proposes categories of measures for an organization to use to track its performance. These authors suggest that organizations should track four types of measures:

- financial,
- customer satisfaction,
- internal processes, and
- innovation and improvement activities.

By specifying only *categories* of measures, the balanced scorecard proposal leaves sufficient flexibility for organizations to define measures within each category that are most relevant to their strategic goals. There are many ways of measuring, e.g., customer satisfaction, and each organization should choose measures appropriate for its own market, customers, and products.

On the other hand, the balanced scorecard *requires* that measures from each of the four categories be chosen and tracked. This prevents masked tradeoffs. That is, it gives a relatively complete picture of the business performance of a company. By monitoring customer, internal, innovation, and financial perspectives, the company can be sure that gains made in one aspect of its operation do not come at the expense of its performance in another aspect. These may or may not be an appropriate set of categories for software organizations, but the idea of breadth of a measurement program is still an important one.

GQM. The GQM paradigm [Basili 82, Basili 91] is a widely-cited method of identifying a useful set of measures. Similar in spirit to the balanced scorecard, GQM starts by analyzing project goals and refining them into a set of questions that can be answered quantitatively. The questions then specify the measures that should be collected in order to track progress toward the goals. Guidelines and templates have been developed to assist the process. However, the measures must be tailored for the environment in which they will be used. This approach contrasts with Hetzel's [Hetzel 93], which is more bottom-up.

CMM/SEI core measures. As pointed out above, for a measurement program to be successful, the set of measures to be tracked must be chosen very carefully so that the

measures accurately reflect the business and engineering objectives of the organization. The following list provides examples of some of the measures likely to be important to many software development organizations:

- Resources expended on software process improvements.
- Resources expended to execute the software processes.
- Amount of time (calendar time) it takes to execute the process.
- Size of the products that result from the software process.
- Quality of the products produced.

For more information on these measures and how they can be defined to help an organization measure its benefits, refer to [Goethert 92] for measuring resources expended and the amount of calendar time to execute the process, [Park 92] and [IFPUG 90] for measuring the size of software products, and [Florac 92] for measuring the quality of products produced. Information about measures related to the CMM is provided in [Baumert 92]. A set of core measures is recommended in [Carleton 92].

Business value. Often the result of most interest to managers is the business value, or ratio of benefits to costs, of the software process improvement effort. Costs of the improvement activities are considered as either nonrecurring costs or recurring costs. Nonrecurring costs are those 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). The organization must decide how it wishes to amortize these costs over time. Recurring costs, on the other hand, are the costs of the activities that have been implemented to monitor products, prevent errors, or continually guide or manage the effort. These costs will be incurred during each time period.

After determining the cost incurred from the SPI effort, the organization determines the amount of money saved. Some of the simpler methods used to determine the amount of money saved include 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.

Suggestions on how to use these methods to measure the costs and benefits from a process improvement effort are discussed in detail in [Rozum 93]. Other sources on how to measure a process improvement effort include [Moller 93] and [Grady 92].

There are a number of reasons why it may be difficult to measure the benefit received from a SPI effort in dollars. For example, the primary benefit received may not be dollars saved, but rather an expansion of gross income. 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 or faster time to market. Sometimes, an organization can only hope that, if it produces a better quality product, customers will buy the product at an increased cost.

Effective measurement of quality, of course, depends on what is important to the customer. Some aspects of quality that could be measured to determine a benefit to the organization from its process improvement efforts include

- Mean time between failures.
- Mean time to repair.
- Availability (as determined by a combination of the mean time between failures and the mean time to repair).
- Customer satisfaction (probably the most overlooked measure of quality).

Intangible benefits. Process improvement efforts can also result in benefits that are either not quantifiable or difficult to convert into a common measure (such as cost, schedule, or quality) for decision-making purposes. For example, better morale, improved understanding of the corporate mission and vision, fewer crises, less stress, less turnover, and better communication within the organization are some of the intangible benefits that have been reported to accompany software process improvement. These particular examples arise from the impact of the SPI effort on the organization's employees. Other potential benefits such as improved reputation, good will, and brand name recognition arise from the impact of SPI on customers. It may be possible to include intangible benefits in cost-benefit studies by measuring or estimating them, but, even when this is not possible, it is important that the analysis acknowledge them.

Awareness of these benefits is important for two reasons. First, when considering alternative improvement programs that offer similar tangible returns, preference should be given to the program that offers the "larger" intangible return. It is not necessary to measure these precisely; simply ranking the preferences in order of estimated magnitude is generally sufficient. Of course, the situation becomes more complex when many such factors need to be considered and they have mixed impacts, i.e., some are negative and some are positive. Creating a list of such impacts often allows one to generate a rough summary estimate, e.g., on balance the intangibles are likely to produce a small positive impact.

Second, these types of benefits are often good indicators of the organizational climate or culture which is typically recognized as a significant part of implementing continuous process improvement. For instance, the SEI approach to software process improvement emphasizes the need for management commitment and sponsorship and the need for widespread buy-in from software practitioners [Humphrey 89, Paulk 93a, Paulk 93b]. Other case studies of product and service innovation and process improvement have also noted the importance of changing the organizational culture to one that supports "empowered employees" as a critical success factor [Deming 86, Kaplan 93]. To measure these types of changes, organizations now frequently conduct employee satisfaction surveys, track the volume of employee suggestions for improvement, and ask employees whether they understand the corporate strategy and whether their work assignments are consistent with the strategy [Kaplan 93].

A few of the organizations that have written about their software process improvement efforts have been impressed by the intangible benefits accruing from their effort. Hughes [Humphrey 91] found fewer crises, a better quality of work life, and a shared sense of mission and pride. Although the software requirements continued to be volatile, the process improvements cushioned the impact of fluctuating requirements on people. As observable indicators of the improvement in work life and decrease in crises, overtime hours were reduced and turnover among the professional software personnel has remained relatively low. Similar benefits were also noted at Raytheon [Dion 92] along with improved communications and support for training. At Tinker AFB the process improvement created a culture of continuous process improvement with wide-spread participation [Lipke 92]. At Tinker AFB, employees actively seek and propose improvement projects to the SEPG and steering team. In addition, increased customer satisfaction has also been noted by an increase in return business. This has been directly attributed to the process improvement effort which focused on quick response to customer problems.

These intangibles were recognized as valuable and important benefits of the process improvement efforts undertaken by the organizations in this study. Indeed, many organizational change strategies explicitly address the need to effect this type of change. What has been less well addressed is how to measure and place a value on these changes. This remains a need for the software community.

After a set of measures has been chosen and defined, and procedures for collecting the data are in place, there is still the question of how to use the data to identify and quantify improvement. This question is addressed in the next section.

5.2 Change over Time

Figure 20 illustrates how an organization might measure the performance of its software processes over time. We assume that the organization is interested in both its ability to predict performance and its actual performance on measures like cost and schedule. In this figure, an organization is comparing its estimates to the actual outcomes of its processes. A project's actual outcome is first normalized to other, similar projects in the organization. For example, cost can be normalized by expressing it as a percentage of budgeted cost. A set of projects is then considered and the normalized outcomes plotted. Those discrete outcomes can also be translated into probabilities, and a probability density function plotted as in the curves in Figure 20. A simpler procedure is to use a histogram to plot the number of projects that fall within ranges such as 0-10% over, 10-20% over, etc. Either one will give a quick, visual way of understanding how well projects are staying within budgets. A similar procedure can be used for other measures of interest, such as time to market or number of defects.

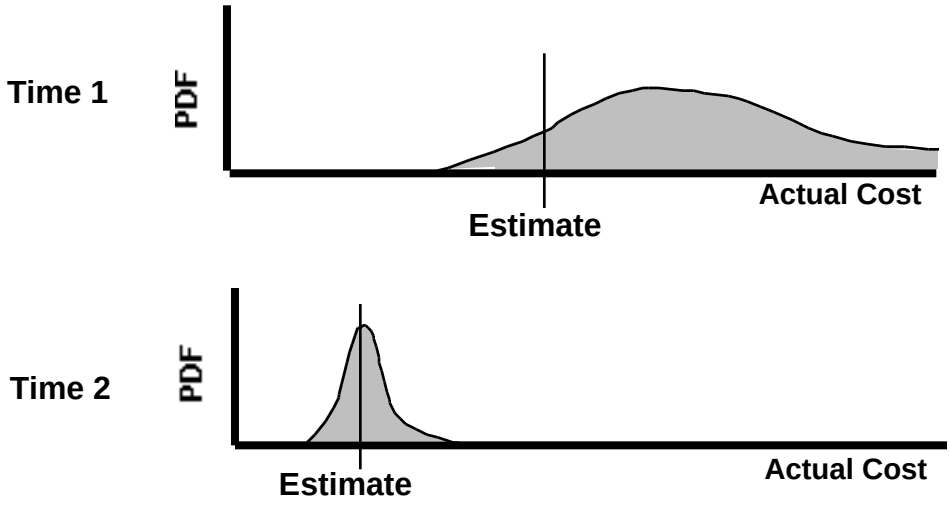

Figure 20. Theoretical Ability to Predict Process Outcomes of Organizations

There are at least three kinds of improvements that can be tracked with these plots [Paulk 93a]. As improvements are made, the actuals should move closer and closer to the estimates as the software process becomes more predictable. In the diagram, the distribution of actual figures becomes more nearly centered on the estimate. In addition, the distribution of actuals should be more tightly clustered around the center (i.e., have a smaller variance). Finally, the average of the actual values should move in the desired direction; average costs, for example, should go down.

While techniques of this sort (see [Rozum 93] for an overview) are very useful for tracking the changes over time in the performance of a software process, they do not specify what is causing the changes. It may be the process improvements, but it might also be increased experience, new people, new analysis or design methods, new tools, change in complexity of applications, faster hardware, and so on. Identifying the causes of changes in performance introduces a new set of issues requiring additional techniques. The following section gives a brief overview of some of the most important considerations when identifying courses of performance changes in the context of software process improvement.

5.3 Identifying Causes

There is an increasing interest in software engineering about analyses that go beyond tracking changes over time by trying to identify the causes of those changes (see, e.g., [Rombach 93, Fenton 93, Kitchenham 94]). Making inferences about causation adds new complexity to the analysis because there are usually many possible causes of changes in performance. Isolating the effects of a particular cause of interest, such as a process

improvement program, generally requires comparisons of results in two or more well-understood conditions. By comparing results where various causes are present or absent, or present in various measurable degrees, one can often reach conclusions about causation with a relatively high degree of confidence.

Unfortunately, practical considerations usually impose serious limitations on the ability of an organization to determine the causes of changes in the performance of its software processes. One of the most common and serious is simply the failure to collect any data at all in a reliable way. And where data collection is initiated, it is often in the context of a pilot project using a new process, so there is often no data from previous projects or other current projects for comparison purposes. It is then impossible, of course, to determine if any change has occurred as a result of the changed process. Where data from many projects are collected, interpretation is often made extremely difficult by failing to capture the important project characteristics that contributed to the results.

Beyond these practical issues lie considerations of research design and data analysis. To be effective, data collection must be very carefully planned and executed (see, e.g., [Fenton 91], [Basili 82]). Simply collecting available data, as we did in this effort, generally does not allow for very sophisticated analyses of causal relations.

6. Conclusions

It is clear, on the basis of the data reported here and elsewhere [Dion 92, Humphrey 91, Lipke 92, Wohlwend 93] that software process improvement can pay off. Diverse organizations with very different software products have approached software process improvement in a variety of ways within the context of the CMM, and each has realized substantial benefit. The rough calculations of business value ratios indicate that the returns can be very substantial. While we do not yet know how typical these results are, they are an indication of what is possible. We are continuing to solicit data of the sort reported here to further test this conclusion.

We are convinced, however, that it is time to move beyond the basic question of whether process improvement **can** pay off. We need to learn how, when, why, and for whom process improvement pays off. This includes both an empirical examination of the assumptions underlying models of process improvement such as the CMM, as well as an analysis of enablers and inhibitors not accounted for in models. We need to achieve an understanding of the critical factors that cause success and failure.

Achieving this understanding is not easy, however. No single organization is likely to have the range of experience necessary to identify these critical factors. We need to examine a broad sample of projects undergoing SPI in order to determine

- characteristics that successful SPI efforts have in common,
- characteristics that unsuccessful SPI efforts have in common, and
- the factors that distinguish successes and failures.

It seems likely that the answers will depend to some extent on the particular improvements attempted, the characteristics of the organization undergoing improvement, the application domain, the technical environment, and the people involved. Any effort to understand the results of SPI must take factors like these into account.

To address these questions, the SEI is investigating new mechanisms, such as a data-sharing consortium of organizations, to provide the essential data and some resources. The problems seem well-suited to a consortium, because a group of organizations working together can accomplish things impossible for any individual member. The benefits to an organization of membership in such a consortium would be considerable:

- Assistance with their software measurement program.
- Assistance with software process assessment.
- Benchmarking against similar organizations for goal-setting and motivation.
- Case studies summarizing experiences with particular improvement efforts.
- Reports of qualitative and quantitative analyses of critical factors drawn from experiences across the consortium.

We are actively working on this and other approaches to understanding the results of SPI as this paper goes to press.

For organizations undertaking SPI efforts, establishing an effective measurement program should be a very high priority. If the program is to gain and maintain the support of management, the leaders of the improvement effort must be able to make the business case. As we discussed above, in order to do this, the organization must develop measures that are closely tied to business strategy, then be able to show they improve over time. This implies, of course, that there is a baseline from pre-SPI projects to provide a background against which change can be detected. Often, of course, this will not be the case, but if data collection begins very early in the improvement effort, this will generally be adequate.

In addition to making the business case for management, collecting and analyzing data is important for guiding the process improvement effort. Some of the early initiatives will succeed and others will not. Effective measurement will allow the organization to make this distinction and propagate what works. Measurement and data analysis are essential components of any software process improvement program.

Bibliography

[Basili 82] Basili, Victor R. & Weiss, David M. *A Methodology for Collecting Valid Software Engineering Data* (TR-1235). College Park, Maryland: University of Maryland, 1982.

[Basili 91] Basili, Victor, R. & Selby, Richard W. "Paradigms for Experimentation and Empirical Studies in Software Engineering." *Reliability Engineering and System Safety 32* (1991): 171-191.

[Baumert 92] Baumert, John H. & McWhinney, Mark S. *Software Measures and the Capability Maturity Model* (CMU/SEI-92-TR-25, ADA257238). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Carleton 92] Carleton, Anita D.; Park, Robert E.; Goethert, Wolfhart B.; Florac, William A.; Bailey, Elizabeth K.; Pfleeger, Shari L. *Software Measurement for DoD Systems: Recommendations for Initial Core Measures* (CMU/SEI-92-TR-19, ADA258305). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Cook 79] Cook, Thomas D. & Campbell, Donald T. *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Chicago: Rand McNally, 1979.

[Deming 86] Deming, W. Edwards. *Out of the crisis.* Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1986.

[Dion 92] Dion, Raymond. "Process improvement and the corporate balance sheet." *IEEE Software 10*, 4 (1992): 28-35.

[Fenton 91] Fenton, Norman. *Software Metrics: A Rigorous Approach*. New York: Chapman and Hall, 1991.

[Fenton 93] Fenton, Norman. "How Effective Are Software Engineering Methods?" *Journal of Systems and Software 22,* (1993): 141-146.

[Florac 92] Florac, William A. *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22, ADA258556). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Goethert 92] Goethert, Wolfhart B.; Bailey, Elizabeth K.; & Busby, Mary B. *Software Effort And Schedule Measurement: A Framework for Counting Staff-Hours and Reporting Schedule Information* (CMU/SEI-92-TR-21, ADA258279). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Grady 93] Grady, Robert B. *Practical Software Metrics for Project Management and Process Improvement.* Englewood Cliffs, NJ: Prentice Hall, 1992.

[Hetzel 93] Hetzel, William C. *Making software measurement work*. Boston : QED Pub. Group, 1993.

[Humphrey 87] Humphrey, Watts S. & Sweet, W. *Method for Assessing the Software Engineering Capability of Contractors.* (CMU/SEI-87-TR-23, ADA187230). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1987.

[Humphrey 89] Humphrey, Watts S. *Managing the Software Process.* Reading, MA: Addison-Wesley, 1989.

[Humphrey 91] Humphrey, Watts S.; Snyder, Terry R.; & Willis, Ronald R. "Software Process Improvement at Hughes Aircraft." *IEEE Software* (July 1991): 11-23.

[IFPUG 90] *Function Points As Assets: Reporting to Management*. International Function Points Users Group, 1990.

[Kaplan 92] Kaplan, Robert S. & Norton, David P. "The Balanced Scorecard -- Measures that Drive Performance." *Harvard Business Review*, (January-February 1992): 71-79.

[Kaplan 93] Kaplan, Robert S. & Norton, David P. "Putting the Balanced Scorecard to Work." *Harvard Business Review*, (September-October 1993): 134-147.

[Kitchenham 94] Kitchenham, B. A.; Linkman, S. G.; & Law, D. T. "Critical Review of Quantitative Assessment." *Software Engineering Journal* (March 1994): 43-53.

[Lipke 92] Lipke, Walter H. & Butler, Kelley L. "Software Process Improvement: A Success Story." *Crosstalk*, November 1992, pp. 29-39.

[Moller 93] K.-H. Moller and D. J. Paulish. *Software Metrics*. New York: Chapman & Hall Computing, 1993.

[Park 90] Park, Chan S. and Gunter P. Sharp-Bette. *Advanced Engineering Economics.* New York: John Wiley & Sons, 1990.

[Park 92] Park, Robert E. *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20, ADA258304). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Paulk 93a] Paulk, Mark C.; Curtis, Bill; Chrissis, Mary B.; & Weber, Charles V. *Capability Maturity Model for Software (Version 1.1)* (SEI/CMU-93-TR-24, ADA263403). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

[Paulk 93b] Paulk, Mark C.; Weber, Charles V.; Garcia, Suzanne M.; Chrissis, Mary Beth; Bush, Marilyn. *Key Practices of the Capability Maturity Model, Version 1.1* (CMU/SEI-93-TR-25, ADA263432). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

[Rombach 93] Rombach, H. Dieter; Victor R. Basili, Richard W. Selby (Eds.) *Experimental Software Engineering Issues: Critical Assessment and Future Directions.* New York: Springer Verlag, 1993.

[Rozum 93] Rozum, James R. *Concepts on Measuring the Benefits of Software Process Improvements* (CMU/SEI-93-TR-09, ADA266994). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

[Townshend 94] Townshend, Patrick L.; & Gebhardt, Joan E. “The Right Question.” *Quality Data Processing*.

[Wohlwend 93] Wohlwend, Harvey ; Rosenbaum, Susan. "Software Improvements in an International Company." *Proceedings of ICSE '93*, pp. 212-220.