

Software Process Improvement (SPI): Modeling Return on Investment (ROI)¹

by David F. Rico

ABSTRACT

The purpose of this article is to exhibit metrics and models for estimating return on investment (ROI) of software process improvement (SPI). Additionally, this article is designed to show software managers and engineers: 1) how to estimate ROI early, quickly, and accurately, 2) how to maximize ROI using total life cycle costs, and 3) how to estimate ROI for Inspections, Personal Software Processsm (PSPsm), Team Software Processsm (TSPsm), Software Capability Maturity Model[®] (SW-CMM[®]), ISO 9001, and CMM Integrationsm (CMMIsm). While, this article draws upon authoritative sources of data for estimating ROI and exhibits relevant approximations of ROI, it is not intended to be an exhaustive analysis of ROI data in of itself. (This article is exemplified by the powerful combination of late-breaking research in cost and benefit analysis for SPI “and” scholarly methods in ROI analysis.)

INTRODUCTION

ROI, as its name implies, is the quantification of the benefits

¹ This article is based on Rico [1].
sm Personal Software Process, PSP, Team Software Process, TSP, Capability Maturity Model Integration, and CMMI are service marks of Carnegie Mellon University.

[®] Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

received or financial return of any given investment. Organizations make many investments in software engineering in order to grow their business base, satisfy customer requirements, or improve lagging productivity and quality. These investments often come in the form of hiring practices, tools and technologies, methods and processes, training and education, and adoption of government and industry standards.

However, software engineering investments often turn into expenses, sunk or irrecoverable costs, or simply failed attempts at satisfying short and long-term business goals and objectives. The ability, or rather inability, of software managers and engineers to accurately quantify the costs and benefits of such investments, or rather expenditures, before, during, and after their introduction is partly to blame.

Therefore, methods are needed to accurately estimate, calculate, and quantify investments in software engineering technologies before, during, and after their introduction. In particular, methods are needed to estimate the even narrower field of SPI tools, techniques, methods, and standards, because the goal of SPI is specifically to increase business value, satisfy customers, and improve market competitiveness in terms of productivity and quality.

This article is designed to show software managers and engineers how to estimate ROI early, quickly, and accurately by applying practical top-down methods for rapidly producing early and authoritative estimates of ROI. It is also designed to show software managers and engineers how to maximize ROI using total life cycle costs by applying practical, authoritative, and well established techniques for producing holistic, well-rounded, and convincing estimates of ROI for SPI. And, it is designed to show software managers and engineers how to apply simple, but powerful techniques for producing estimates of ROI for Inspections, PSP, TSP, SW-CMM, ISO 9001, and CMMI.

The goals and objectives of showing software managers and engineers how to estimate ROI early, maximize ROI using total life cycle costs, and apply simple but powerful techniques to estimate the ROI of popular SPI approaches, include:

- Provide authoritative guidance for beginners to estimate ROI for justifying SPI initiatives, CMMI, PSP, and TSP adoption, and other forms of SPI.
- Consolidate the myriad of research, books, methods, and collective knowledge into a single portable source that can easily be applied by anyone right out-of-the-box.

- Provide authoritative guidance on key strategies for identifying and quantifying all life cycle costs, which contribute to accurately, professionally, and convincingly modeling ROI for CMMI, SW-CMM, PSP, TSP, and other SPI approaches.
- Exhibit a highly simplified explanation of classical techniques, which cover the entire life cycle of a product from development through maintenance, which are highly regarded by the U.S. Government for mission critical system acquisitions.

are still struggling to identify and define their own relevant metrics and models, measure and model the relevant characteristics of their processes and products, and pinpoint the drivers of costs and benefits that contribute to accurately determining ROI.

While, there have been great strides or quantum leaps forward in the fields of software metrics and models by the likes of Kan [2], Pham [3], and Humphrey [4], the average software practitioner continues to refute the foundation established by these scholars and

subtracting out the costs. The benefits may or may not be greater than zero, and may or may not exceed the costs.

Before investing in any SPI method, software managers and engineers should estimate the costs and benefits of multiple alternatives. And, ultimately, of course, select a SPI method with a greater ROI than the alternatives.

A fundamental assumption is that there are benefits to SPI, those benefits are quantifiable, and the benefits not only exceed the costs,

but outweigh the costs convincingly enough to justify the difficulties associated with contemporary SPI methods (e.g., complexity, time, and labor). Practitioners, even from very mature organizations, continue to

- Provide simple examples and methods for producing authoritative estimates of the most relevant approaches to SPI.
- Provide simple, practical, and extremely useful ROI techniques based on total life cycle costs, which are rarely used in state-of-the-art ROI literature and practice (even by the most mature software organizations).

Definition	Source
Actual value developed by comparing program costs to benefits	Rachlin [6]
Measuring magnitude of benefits relative to costs	Lim [7]
Net benefit after expending some level of resources	Poulin [8]
Profit computed by dividing net income by assets used	Reifer [9]

Figure 1: Definition of ROI

practice their discipline without the light of software metrics and models. However, it is primarily McGibbon [5] and Rachlin [6] that have provided us with an oracle for unlocking the mystery and ultimately the definition of ROI. Proper identification and analysis of McGibbon's and Rachlin's works leads us to the authoritative definitions of ROI in Figure 1.

A proper interpretation of these four definitions is simply adding up all of the benefits and

believe there is no ROI for SPI [10], ROI is nominal [11], or the payback period for SPI lies far out into the distant future [12, 13].

Rico [14] stands alone in producing ground-breaking evidence that ROI is not only possible and substantial, but can be achieved in hours and days (even within the bounds of a single project). This is an important aspect of ROI, since the one percent of defense contractors that finally submit to the

DEFINITIONS

ROI, in spite of its relative simplicity and maturity as an outright discipline, is not well understood by the fields of software engineering and SPI. This is partly due to the fact that software managers and engineers

application of SPI, demand early results within the least possible constraints of time and cost. (Defense contractors are not in the least enamored with research projecting ROI into the far distant future.) “Early” ROI will be discussed in further detail in the section on Advanced Issues.

METHODS

There are quite literally a myriad or plethora of methods for determining ROI. The first major challenge for software managers and engineers is to identify one or more approaches for estimating the ROI of SPI. Therein lies the problem. There are few quantitative studies on the costs and benefits of SPI.

There are even fewer studies dedicated solely to the analysis of ROI for SPI.

Cost and benefit studies of SPI have trickled in over the last decade. However, they have been too few and far between, they sparingly report any metrics and models for costs, benefits, and ROI, and the few that could have been truly useful are somewhat esoteric and confusing.

Rico [14] is a broad survey of metrics, models, methods, and data for costs and benefits of SPI (as well as an in-depth analysis of ROI and breakeven points). Rico [1], upon which this article is based, focuses solely on simple methods for estimating the ROI of SPI.

However, even these studies and analyses tend to be somewhat lengthy and even esoteric. It’s

quite a challenge to summarize the cost and benefit factors and values that contribute to estimating ROI, without failing to provide a scholarly analysis of the assumptions surrounding each factor. (This article will attempt to summarize the factors and values surrounding ROI metrics and models, as well as their associated assumptions in order to help

software managers and engineers estimate ROI for SPI.)

Here is a summary of decision analysis methods which may be used for analyzing the costs and benefits of SPI, and even ROI itself, in Figure 2.

Except for Rico [14] and Reifer [17], few of these texts focus specifically on ROI for SPI.

While, Reifer provides a rare survey of techniques for constructing generalized business cases, Rico gets closer to the issue at hand by providing a methodology for analyzing the costs and benefits of SPI, as well as ROI and breakeven analysis of SPI.

Rachlin [6] zeros in on the most relevant approach for estimating ROI, both simplistic and advanced. However, since

Rachlin only provides generalized ROI models for any application, the harder part of quantifying the atomic-level costs and benefits for SPI cannot be found in his text. Once again, therein lies the problem.

It’s not just a matter of identifying the correct approach for determining ROI, as provided by

Methods	Source
Defect removal model, linear optimization, decision analysis model	Rico [14]
Mathematical programming, goal programming, transportation-assignment, branch and bound, decision tables, decision trees, forecasting, PERT/CPM, inventory, Markov chains, waiting lines, simulation, heuristic programming, game theory, dynamic programming	Turban [15]
Expected value, optimal decision policy, decision trees, value of information, Monte Carlo simulation, dynamic project modeling, parameter method, moments method, fuzzy logic, approximate integration, etc.	Schuyler [16]
Breakeven analysis, cause-and-effect analysis, cost/benefit analysis, value chain analysis, investment opportunity analysis, pareto analysis, payback analysis, sensitivity analysis, trend analysis	Reifer [17]
Benefit/cost ratio, ROI (%), ROI Process	Rachlin [6]

Figure 2: Methods for ROI

Rachlin [6]. But, software managers and engineers are faced with the fundamental inability to identify the cost and benefit factors of SPI, which drive an authoritative ROI approach as exhibited by Rachlin.

Therefore, when the cost and benefit factors of SPI as presented by Rico [14] are combined with the fundamental ROI model as presented by Rachlin [6], a model for estimating the ROI of SPI suddenly emerges. Rachlin's basic model for ROI will be presented in the next section, while Rico's cost and benefit factors for SPI will be presented and explained in the section entitled, Examples.

(It's important to note that Rachlin doesn't merely provide equations for ROI, but a comprehensive, field-proven methodology for estimating ROI. However, Rachlin's complete ROI methodology is beyond the scope of this article, and may even be considered overkill, and perhaps unnecessary, for everyday practical application and use.)

MODEL

While, one can spend literally months and years analyzing the sparse literature and searching for relevant approaches to defining and estimating ROI, Rachlin [6]

provides one-stop shopping on this seemingly futile journey. Rachlin defines the basic model for estimating ROI, as well as a comprehensive "process" for applying these simplistic equations in a scholarly and professional manner.

Rachlin's [6] fundamental ROI model consists of two basic equations (also depicted or shown in Figure 3):

- Benefit/Cost Ratio (B/CR): B/CR is a simple process of dividing the benefits of SPI by the costs of SPI.

Type	Model
Benefit/ Cost Ratio	$B/CR = \frac{\text{Benefits}}{\text{Costs}}$
Return on Investment	$ROI (\%) = \frac{\text{Benefits} - \text{Costs}}{\text{Costs}} \times 100$

Figure 3: Model for ROI

- Return on Investment (ROI%): The ROI% equation is similar to the B/CR equation, except that the costs of SPI are subtracted from the benefits of SPI before dividing by the costs (and then converting the result into a percentage).

(While, Rachlin [6 and 18] go on to provide a patented process for applying these basic equations, this labor and cost-intensive approach is considered beyond the scope of this article, not

particularly applicable to SPI, and unnecessary for rapid and authoritative development of ROI estimates for SPI.)

Let's stop and examine Rachlin's [6] basic ROI equations for just a moment. Notice that the ROI equations consist of only two terms:

- Benefits: For SPI, benefits generally consist of the quantitative value, payback, or interest that is returned for an investment in SPI.
- Costs: For SPI, costs refer to the expenses, expenditures, and capital outlay necessary to apply a SPI approach, which will result in some benefit.

We've essentially arrived at the crossroads, or impasse as some

may assert, in the estimation of ROI for SPI. As simple and innocuous as the terms benefit and cost may appear on the surface, they are terms surrounded in the darkness of the medieval SPI era that we live in, ambiguity and inconsistency of definition, application, and use, and even outright controversy, dismay, and disbelief.

One of the most amazing phenomenon that has arisen from the battle over the costs and

benefits of SPI, is the categorical rejection of cost and benefit metrics, models, data, and especially comparative studies that do not shine brightly upon prevailing methods, standards, approaches, and conventional wisdom. In other words, when a study begins to compare the costs and benefits of multiple alternatives, someone is bound to be offended that their favorite approach to SPI tends to have fewer benefits and more costs than a more efficient alternative.

This was certainly the case when McGibbon's [5] seminal classic emerged in 1996, it has certainly been the case with Rico's [14] comparison of the top eight approaches to SPI, and it was certainly the case when Rico [1] was presented in early 2002. It's not uncommon for U.S. military officers in key pentagon acquisition positions to explode in fury and anger when their favorite approach to SPI is reported to have few benefits and many costs. It's more often the case that studies comparing the costs and benefits of SPI are ignored, unreferenced, and swept under the rug because of their seemingly unflattering view of some approaches to SPI.

While, some are convinced that comparative studies of SPI approaches along with their costs and benefits are meant to denigrate and deconstruct the fledgling discipline of SPI, these studies are more often than not created to form a solid foundation for making quantitatively beneficial decisions, which will

ensure the success of SPI initiatives. In other words comparative studies are meant to help, not hurt has many would assert.

McGibbon [5], Rico [14], and Rico [1] are excellent resources for objectively analyzing the fundamental cost and benefit factors or drivers associated with SPI methods, approaches, techniques, and more importantly, decision-making that will ensure the success of not only SPI initiatives, but the success of the SPI field itself. McGibbon establishes a seminal framework for comparing the costs and benefits of SPI approaches, Rico [14] builds upon and expands the breadth and depth of McGibbon's basic framework, and Rico [1] along with this article begin attempting to bring ROI analysis, estimation, and quantification into the center stage of practical, simplistic, and everyday decision-making.

It's important to note here that ROI estimation is very sensitive to accurate quantification of both the costs and benefits of SPI. Approaches to SPI with high costs will have a lower ROI, benefits being equal. Approaches with low costs will tend have high ROI estimates.

For example, the same basic benefit model was used by Rico [1 and 14]. However, Rico [1] quantified costs in greater detail which caused the best two approaches to change precedence with respect to ROI, as well as cost and benefit efficiency. Rico

[14] really illuminated sensitivities to benefits. Rico [1] illuminated sensitivities to costs. (These subtle differences provide an early clue to success. That is, apply approaches to SPI with minimal costs and maximum benefits.)

The issues of how to accurately quantify the benefits of SPI will be enumerated in the Section on Examples. The issues on how to accurately quantify the costs of SPI will be enumerated in the section entitled, Costs/Benefits.

EXAMPLES

This section provides simple, but powerful, authoritative, and relatively accurate examples of how to apply Rachlin's [6] basic equations for estimating the ROI of six major approaches to SPI. Again, Rachlin's B/CR and ROI% equations will be applied to benefit data from Rico [14] as well as other authoritative sources of SPI data. The six approaches to SPI are:

- **Inspection:** The software inspection process is a highly-structured and facilitated group meeting to objectively identify the maximum number of software defects with the purpose of improving software quality [19].
- **PSP:** The PSP is a training curriculum to teach simple, but powerful techniques in software project management and software quality management [20]. It requires trainees to develop a series of mathematically intensive

computer programs using increasingly complex software management techniques. The purpose of PSP is to convince trainees to apply these techniques in everyday practice by experiencing their value and benefits first-hand.

- **TSP:** The TSP is an extension of PSP, which introduces group software project management techniques versus the individual focus taught by PSP [21].
- **SW-CMM:** The SW-CMM is a supplier selection model created by the U.S. DoD to evaluate, identify, and select software contractors that practice minimum software project management techniques [22].
- **ISO 9001:** ISO 9001, like the SW-CMM, is a supplier selection model created by the European Union to evaluate, identify, and select suppliers that practice minimum quality management techniques [23].
- **CMMI:** The CMMI, which is the newest version of SW-CMM, is also a supplier selection model created by the U.S. DoD to evaluate, identify, and select systems engineering contractors that practice minimum systems engineering

project management techniques [24].

The purpose of these examples is to show software managers and engineers how to estimate ROI for SPI using authoritative ROI metrics, models, and processes. The purpose is not to serve or act as an exhaustive scholarly analysis of the costs of SPI, the benefits of SPI, or ROI for SPI.

Any in-depth, scholarly study of ROI for SPI must contain an empirical analysis of the costs and benefits of SPI, perhaps consider

CMM, ISO 9001, and CMMI in no way assumes these are the best or only SPI methods. In fact, there are many SPI approaches. And, the best ones are yet to be identified, quantified, and exploited [25]. In fact, while these may be good short-term solutions to begin with, one would surely be succumbing to imminent failure if even lower cost, higher payback approaches to SPI weren't employed. However, it is important to note that these six approaches to SPI are considered best-in-class for this early era in SPI history, and other approaches to SPI have not even been mentioned because they are unquantifiable, subjective, and may do even more harm than good. Many digressive and deconstructive rapidly sweeping fads were omitted from this

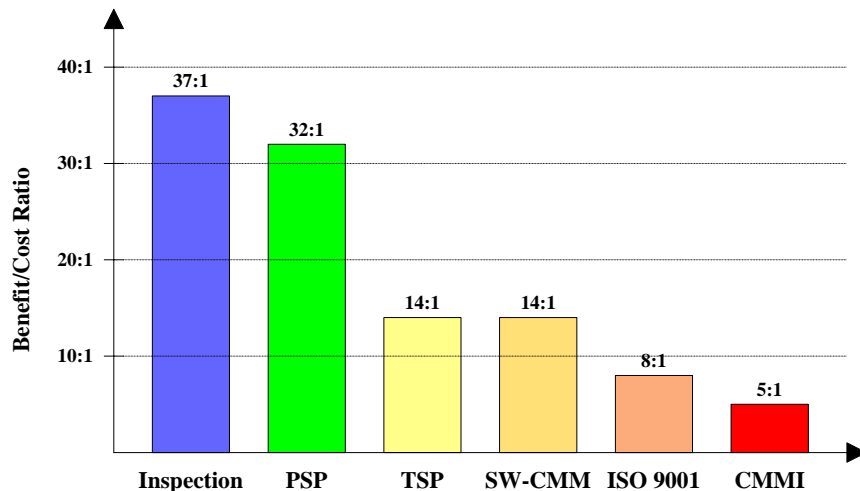


Figure 4: Examples for ROI

more than one ROI approach, and then make assertions about the ROI of SPI after considering valid cost and benefit data. However, this article certainly serves as a framework and highly structured proposal for such an in-depth study.

At a minimum, this article provides practical tools to estimate the ROI of SPI for immediate application and use. (And, of course, the selection of Inspections, PSP, TSP, SW-

analysis, because of their primitive notions.)

INSPECTION

Inspections are manually intensive meetings to perform static analysis of software products to objectively identify the maximum number of software defects possible. Many have challenged their costs, benefits, effectiveness, and even asserted the greater benefits of highly structured individual reviews

[26]. (This fails to even mention the cultural barriers and hopeless resistance to this non-programming activity.) However, what these studies completely fail to mention is that Inspections, when performed, offer substantial benefits to omitting any sort of pre-test software defect removal. In other words, as inefficient as they are, they offer many incontrovertible benefits.

Let's examine the dynamics of Inspection cost, benefit, and ROI analysis using Rachlin's [6] equations for B/CR and ROI%. Remember, there are only two basic terms, 1) costs and 2) benefits.

- **Training Cost:**

Let's begin by modeling the training costs for implementing Inspections on a four-person project. The average market price for Inspection training is about \$410 per person. The average length of time for Inspection training is three days or 24 business hours. At a minimum cost of \$100 per hour, training time comes to \$2,400. Add \$410 to \$2,400 for a total of \$2,810 per person for Inspection training. Multiply \$2,810 by four people and that comes to \$11,240 to train four people to perform Inspections.

- **Implementation Cost:** Now let's examine the cost of

implementing Inspections by our four trained inspectors. Let's assume the project will develop 10,000 software source lines of code (SLOC), which is not unlikely for a web project in modern times. (Inspections of requirements, designs, and tests drive the Inspection costs even higher, but are omitted for simplicity's sake.) At an Inspection rate of 240 SLOC per meeting, that comes to approximately 41.67 meetings. (The optimal Inspection rate is 120 SLOC per meeting, so we're lowering the cost and

- **Total Cost:** So, we add the training cost of \$11,240 to the implementation cost of \$70,833, and we arrive at a total cost of \$82,073 for four trained inspectors to Inspect 10,000 SLOC.

- **Total Life Cycle Benefits:** The estimated maintenance hours for 10,000 SLOC after our four trained inspectors perform their Inspections are 11,806. The estimated maintenance hours for 10,000 SLOC with no Inspections are 41,800. So, our four trained inspectors have saved 29,994 maintenance

hours on their very first implementation of Inspections. (Maintenance savings are underestimated by up to four times. The maintenance hours assume a world class testing capability, which few

organizations actually have.) Multiply 29,994 by \$100 and the estimated savings are an eye-popping \$2,999,400. (See Rico [14] for an in-depth analysis of software maintenance effort with and without Inspections.)

- **B/CR:** (The formula for B/CR is benefits divided by costs.) Therefore, divide \$2,999,400 by \$82,073 and the B/CR for Inspections is 37:1.

Model	Estimation
Training Cost	$(\$410 \text{ fee} + \$2,400 \text{ labor}) \times 4 \text{ people} = \$11,240$
Project Cost	$10 \text{ KSLOC} / 240 \text{ LOC per meeting} * 17 * \$100 \text{ per hour} = \$70,833$
Life Cycle Benefits	$(41,800 - 11,806 \text{ maintenance hours}) * \$100 \text{ per hour} = \$2,999,400$
Benefit/Cost Ratio	$\$2,999,400 \text{ benefits} / \$82,073 \text{ costs} = 37:1$
ROI%	$(\$2,999,400 \text{ benefits} - \$82,073 \text{ costs}) / \$82,073 \text{ costs} = 3,555\%$

Figure 5: ROI for Inspections

efficiency of Inspections a little.) Since each Inspection run requires about 17 hours for planning, overviews, preparation, meetings, rework, and follow-up, we then multiply 41.67 by 17 for a total of 708.33 hours. Once again, at \$100 per hour, that comes to \$70,833 for our four trained inspectors to perform Inspections on 10,000 SLOC. (See Rico [14] for an in-depth analysis of Inspection and Test metrics, models, effort, and costs.)

- **ROI%:** (The formula for ROI% is benefits less costs divided by costs times 100.) Therefore, first subtract the \$82,073 in Inspection costs from the \$2,999,400 in Inspection benefits and divide the results by the \$82,073 in Inspection costs and multiply by 100 for an impressive ROI% of 3,555%.

(Remember, the total payback period is only four staff months, so it is unnecessary to use complex discounting methods to determine the value of the investment, which are more applicable to capital investments in plants, buildings, and facilities. In fact, the \$82,073 in total Inspection cost was completely recovered during the first of 42 Inspections. The remaining 41 Inspections were all profit. See Rico [14] for an in-depth analysis of breakeven points for Inspections.)

PSP

As mentioned before, PSP is a highly effective training curriculum designed to teach software engineers the benefits of simple, but powerful techniques in software project management and software quality management. PSP is composed of seven simple software life cycles consisting of

increasingly complex methods in software project management and software quality management.

The goal is for software engineers to develop a series of complex mathematical computer programs using each of the seven software life cycles. (PSP-trained software engineers often complain that the mathematical exercises confound the process of learning PSP itself.) PSP is designed for software engineers to experience, firsthand, the increasing benefits in terms of precision and quality of using basic software project and quality management

(Issues of cost, obscurity, difficulty, scalability, and overzealous copyright protection have relegated PSP to the dusty shelves of academic libraries. It's really a darn shame; because PSP as a software project management training curriculum is orders of magnitude more effective than the courses of most consultants and authors. Rico [27] produced a 524-page software life cycle to help software engineers transition PSP from the classroom to the field, which is prohibited from distribution by Carnegie Mellon University for business-competitiveness reasons.)

Now, let's examine the dynamics of PSP cost, benefit, and ROI analysis using Rachlin's [6] equations for B/CR and ROI%.

- **Training Cost:** Let's begin by modeling the training costs

for implementing PSP on a four-person project. The Software Engineering Institute's (SEI's) price for PSP training is \$5,000 per person. The costs of the airline, hotels, meals, and parking are about \$5,400 for two weeks. The length of time for PSP training is 10 days or 80 business hours. Each hour of classroom time requires approximately one hour of non-classroom time for a total of 80 more hours. At a minimum cost of \$100 per hour, training time comes to

Model	Estimation
Training Cost	$(\$5,000 \text{ fee} + \$5,400 \text{ expenses} + \$16,000 \text{ labor}) \times 4 \text{ people} = \$105,600$
Project Cost	$10 \text{ KSLOC} / 25 * \$100 \text{ per hour} = \$40,000$
Life Cycle Benefits	$(46,646 \text{ maint and develop hours}) * \$100 \text{ per hour} = \$4,664,600$
Benefit/Cost Ratio	$\$4,664,600 \text{ benefits} / \$145,600 \text{ costs} = 32:1$
ROI%	$(\$4,664,600 \text{ benefits} - \$145,600 \text{ costs}) / \$145,600 \text{ costs} = 3,104\%$

Figure 6: ROI for PSP

techniques. (PSP has as its underlying foundation, the notion that if software engineers find twice as many defects before testing as during testing, the result will be greater project precision and product quality.)

PSP is not merely meant to be an academic classroom training methodology, but is designed to convince software managers and engineers with personal empirical data to be bold enough to transfer these techniques into everyday practical use with similar benefits.

\$16,000. Add \$5,000, \$5,400, and \$16,000 for a total of \$26,400 per person for PSP training. Multiply \$26,400 by four people and that comes to \$105,600 to train four people to perform PSP.

- **Implementation Cost:** Now let's examine the cost of implementing PSP by our four PSP-trained engineers. Let's assume the project will develop 10,000 software source lines of code (SLOC), once again, which is not unlikely for a web project in modern times. At an average productivity rate of 25 SLOC per hour, that comes to approximately 400 hours. At \$100 per hour, that comes to \$40,000 for our four PSP-trained engineers to produce 10,000 SLOC using PSP. (See Rico [14] for an in-depth analysis of PSP metrics, models, effort, and costs.)

- **Total Cost:** So, we add the training cost of \$105,600 to the implementation cost of \$40,000, and we arrive at a total cost of \$145,600 for four PSP-trained engineers to produce 10,000 SLOC using PSP.
- **Total Life Cycle Benefits:** The estimated maintenance hours for 10,000 SLOC after our four PSP-trained engineers apply PSP are zero. The estimated maintenance hours for 10,000

SLOC without PSP are 41,800. So, our four PSP-trained engineers have saved 41,800 maintenance hours on their very first application of PSP. (Maintenance savings are underestimated by up to four times. The maintenance hours assume a world class testing capability which few organizations actually have.) Typical software development hours for 10,000 SLOC are 5,088. However, software development hours with PSP are only 242, for an additional savings of 4,846 hours. Add

41,800 maintenance hours saved to 4,846 development hours saved for a total of 46,646 saved software maintenance and development hours. Multiply 46,646 by \$100 and the estimated savings are an impressive \$4,664,600. (See Rico [14] for an in-depth analysis of software maintenance effort with and without PSP.)

- **B/CR:** (The formula for B/CR is benefits divided by costs.) Therefore, divide \$4,664,600

by \$145,600 and the B/CR for PSP is 32:1.

- **ROI%:** (The formula for ROI% is benefits less costs divided by costs times 100.) Therefore, first subtract the \$145,600 in PSP costs from the \$4,664,600 in PSP benefits and divide the results by the \$145,600 in PSP costs and multiply by 100 for an impressive ROI% of 3,104%.

(Remember, the total payback period is only three staff months, so it is unnecessary to use complex discounting methods to

determine the value of the investment, which are more applicable to capital investments in plants, buildings, and facilities. In fact, the \$145,600 in total PSP cost was completely recovered during the first hours of

applying PSP. The remaining 399 PSP hours were all profit. See Rico [14] for an in-depth analysis of breakeven points for PSP.)

TSP

TSP, an expansion of PSP, guides software engineering teams in developing software products. Use of TSP improves quality and productivity of software engineering teams while helping them meet cost and schedule constraints. TSP is designed for teams of up to 20 members, and

Model	Estimation
Training Cost	(\$9,000 fee + \$8,100 expenses + \$20,000 labor) x 4 people = \$148,400
Project Cost	10 KSLOC / 6.12 * \$100 per hour = \$163,400
Life Cycle Benefits	(45,254 maint and develop hours) * \$100 per hour = \$4,525,400
Benefit/Cost Ratio	\$4,525,400 benefits / \$311,800 costs = 14:1
ROI%	(\$4,525,400 benefits - \$311,800 costs) / \$311,800 costs = 1,351%

Figure 7: ROI for TSP

larger multi-team TSP processes are designed for teams of up to 150 members. However, these larger scale TSP versions have not been made publicly available, as is the case with much of the TSP. Several completed textbooks on TSP have been withheld by Carnegie Mellon University at the time of this writing for unknown reasons.

Now, let's examine the dynamics of TSP cost, benefit, and ROI analysis using Rachlin's [6] equations for B/CR and ROI%.

- **Training Cost:** Let's begin by modeling the training costs for implementing TSP on a four-person project. The SEI's price for TSP training is \$4,000 per person. The costs of the airline, hotels, meals, and parking are about \$2,700 for one week. The length of time for TSP training is 5 days or 40 business hours. At a minimum cost of \$100 per hour, training time comes to \$4,000. Add \$4,000, \$2,700, and \$4,000 for a total of \$10,700 per person for TSP-specific training. Add the \$26,400 for PSP training to the \$10,700 for TSP training and the total overall TSP costs come to a breathtaking \$37,100 per person. Multiply \$37,100 by four people and that comes to a budget-busting \$148,400 to train four people to perform TSP.
- **Implementation Cost:** Now let's examine the cost of implementing TSP by our four TSP-trained engineers. Let's assume the project will develop 10,000 software source lines of

code (SLOC), once again, which is not unlikely for a web project. At an average productivity rate of 6.12 SLOC per hour, that comes to approximately 1,634 hours. At \$100 per hour, that comes to \$163,400 for our four TSP-trained engineers to produce 10,000 SLOC using TSP. (See Humphrey [28] for an in-depth analysis of TSP metrics, models, effort, and costs.)

- **Total Cost:** So, we add the training cost of \$148,400 to the implementation cost of \$163,400, and we arrive at a total cost of \$311,800 for four TSP-trained engineers to produce 10,000 SLOC using TSP.
- **Total Life Cycle Benefits:** The estimated maintenance hours for 10,000 SLOC after our four TSP-trained engineers apply TSP are zero. The estimated maintenance hours for 10,000 SLOC without TSP are 41,800. So, our four TSP-trained engineers have saved 41,800 maintenance hours on their very first application of TSP. (Maintenance savings are underestimated by up to four times. The maintenance hours assume a world class testing capability, which few organizations actually have.) Typical software development hours for 10,000 SLOC are 5,088. However, software development hours with TSP are only 1,634, for an additional savings of 3,454 hours. Add 41,800 maintenance hours saved to 3,454 development hours saved for a total of

45,254 saved software maintenance and development hours. Multiply 45,254 by \$100 and the estimated savings are an impressive \$4,525,400. (See Rico [14] for an in-depth analysis of software maintenance effort with and without Test.)

- **B/CR:** (The formula for B/CR is benefits divided by costs.) Therefore, divide \$4,525,400 by \$311,800 and the B/CR for TSP is 14:1.
- **ROI%:** (The formula for ROI% is benefits less costs divided by costs times 100.) Therefore, first subtract the \$311,800 in TSP costs from the \$4,525,400 in TSP benefits and divide the results by the \$311,800 in TSP costs and multiply by 100 for an impressive ROI% of 1,351%.

(Remember, the total payback period is only eleven staff months, so it is unnecessary to use complex discounting methods to determine the value of the investment, which are more applicable to capital investments in plants, buildings, and facilities. In fact, the \$311,800 in total TSP cost was completely recovered during the first hours of applying TSP. The remaining 1,600 TSP hours were all profit. See Rico [14] for an in-depth analysis of how to estimate breakeven points.)

SW-CMM

SW-CMM is a set of minimum criteria for evaluating the software engineering management

capabilities of U.S. military suppliers. SW-CMM is a derivative work of Crosby's [29] Maturity Grid, a product of the ITT Corporation. Radice [30] then copied and adapted Crosby's Maturity Grid for IBM, entitling it as IBM's Process Grid. Humphrey [31] then copied and adapted Crosby's and Radice's Maturity Grid and Process Grid, entitling it as Carnegie Mellon University's Process Maturity Grid. Paulk [32] then transformed Humphrey's work into what we now know as Carnegie Mellon University's SW-CMM.

While no one would argue that the SW-CMM is less than the ideal approach to SPI, SW-CMM has become the de facto international standard for SPI. In fact, the majority of organizations applying SW-CMM are not from the U.S. DoD community, but from the international commercial industry. In fact, only less than one percent of U.S. military suppliers apply SW-CMM.

Now, let's examine the dynamics of SW-CMM cost, benefit, and ROI analysis using Rachlin's [6] equations for B/CR and ROI%.

- **Deployment Cost (Level 2):** Let's begin by modeling the deployment costs for implementing SW-CMM for

four projects as a representative sample of a software producing organization. Rico [33] makes the following estimates: 66 hours for 6 policies, 264 hours for 24 procedures, 512 hours for 32 documents, 304 hours for 76 work authorizations, 464 hours for 116 records, 544 hours for 136 reports, and 304 hours for 76 meeting minutes. The total deployment hours for SW-CMM Level 2 are 2,458. Multiply 2,458 by \$100 and that comes to \$245,800.

- **Deployment Cost (Level 3):** Rico [33] makes the following

Model	Estimation
Deployment Cost	(7,105 implementation and prep hours) x \$100 per hour = \$710,500
Assessment Cost	1,000 hours * \$100 per hour + \$40,000 fee = \$140,000
Life Cycle Benefits	(121,642 maint and develop hours) x \$100 per hour = \$12,164,200
Benefit/Cost Ratio	\$12,164,200 benefits / \$850,500 costs = 14:1
ROI%	(\$12,164,200 benefits - \$850,500 costs) / \$850,500 costs = 1,330%

Figure 8: ROI for SW-CMM

estimates: 77 hours for 7 policies, 154 hours for 14 procedures, 1,280 hours for 80 documents, 176 hours for 44 work authorizations, 592 hours for 148 records, 336 hours for 84 reports, and 192 hours for 48 meeting minutes. The total deployment hours for SW-CMM Level 3 are 2,807. Multiply 2,807 by \$100 and that comes to \$280,700.

- **Assessment Preparation Costs:** Let's estimate four projects of five people in 13 indoctrination

courses at 2 hours each which totals 520 hours. Let's similarly estimate four projects of five people in 13 response-conditioning courses at 2 hours, each which also totals 520 hours. Finally, let's estimate four projects of five people in one 40 hour mock assessment or two 20 hour mock assessments for total of 800 hours. Now, let's add 520 indoctrination hours, 520 response conditioning hours, and 800 mock assessment hours for a total of 1,840 hours.

Finally, let's multiply 1,840 by \$100 for a total of \$184,000 in assessment preparation costs.

- **Total Deployment Costs:** Combine \$245,800, \$280,700, and \$184,000 for a total SW-CMM Level 2 and 3

deployment cost of \$710,500.

- **Assessment Cost:** The SEI estimates that an assessment requires up to 3,208 hours of internal labor (not including the assessors effort). However, for our four projects of five people let's estimate 62 hours for planning, 234 hours for preparation, 646 hours for the appraisal itself, and 57 hours of follow-up which totals 1,000 hours. (This doesn't include the assessor's time, and the SEI estimates over three times more

internal effort.) So, now multiply 1,000 by \$100 for a total labor cost of \$100,000 plus \$40,000 in assessment fees for a total assessment cost of \$140,000.

- **Total SW-CMM Cost:** Take a deep breath and add the \$710,500 in total deployment costs to the \$140,000 in assessment costs for a total SW-CMM cost of \$850,500.
- **Total Life Cycle Benefits:** Let's assume each of our four projects also build 10,000 SLOC software products. Let's also assume that each of our four projects apply Inspections to satisfy their SW-CMM Level 3 goals. Now, we're ready to begin estimating the benefits. Let's assume each of our four projects saves an average of 27,867 maintenance hours by performing Inspections for total maintenance savings of 111,466 hours. Now, let's assume our productivity doubles at SW-CMM Level 3 as reported by Diaz [34], which results in a per project savings of 2,544 hours for a total of 10,176 development hours saved. Add the 111,466 hours in maintenance savings to the 10,176 hours in development savings for a total of 121,642 hours saved at SW-CMM Level

3. Multiply 121,642 by \$100 to arrive at an estimated savings of \$12,164,200.

- **B/CR:** (The formula for B/CR is benefits divided by costs.) Therefore, divide \$12,164,200 by \$850,500 and the B/CR for SW-CMM is 14:1.
- **ROI%:** (The formula for ROI% is benefits less costs divided by costs times 100.) Therefore, first subtract the \$850,500 in SW-CMM costs from the \$12,164,200 in SW-CMM benefits and divide the results by the \$850,500 in SW-CMM

used for supplier discrimination and selection.

This description of ISO 9001 is by no means meant to trivialize the importance of this quality management system standard. In fact, organizations may not trade products and services in Europe without certification to this standard. ISO 9001 has taken a greater foothold throughout the world than any other standard of its type, including SW-CMM or any other approach to SPI.

The latest version, ISO 9001:2000, closely aligns itself with ISO 12207, an international software life cycle standard, for interpretation, application, and use by software organizations. It's unclear how CMMI, a peer of ISO 9001:2000 aligns with ISO 12207, as CMMI seems to supersede ISO 12207 in terminological use and fundamental architecture. In fact, CMMI seems to overstep the scope of ISO 12207 substantially.

Model	Estimation
Deployment Cost	12.6 months * 173.33 hours per month * \$100 per hour = \$218,396
Assessment Cost	640 hours * \$100 per hour + \$48,000 fee = \$112,000
Life Cycle Benefits	(27,726 maint and develop hours) x \$100 per hour = \$2,772,600
Benefit/Cost Ratio	\$2,772,600 benefits / \$330,396 costs = 8:1
ROI%	(\$2,772,600 benefits - \$330,396 costs) / \$330,396 costs = 739%

Figure 9: ROI for ISO 9001

costs and multiply by 100 for an impressive ROI% of 1,330%.

ISO 9001

ISO 9001 at most is a generalized international standard for any kind of quality management system for the delivery of any kind of product or service. ISO 9001 is better characterized as a minimum set of strategic criteria for the design of any kind of quality management system, to be

Now, let's examine the dynamics of ISO 9001 cost, benefit, and ROI analysis using Rachlin's [6] equations for B/CR and ROI%.

- **Deployment Costs:** Let's begin by modeling the costs for implementing ISO 9001 in a 20-person software organization. El Emam's [35] cost model results in 2,184

hours to prepare an organization for ISO 9001 registration that is currently non-compliant with 84% of its requirements. Multiply 2,184 by \$100 and that comes to \$218,396. (El Emam's model is not calibrated for ISO 9001:2000.)

- Assessment Costs: Let's estimate four projects of five people at 32 hours each which totals 640 hours to prepare for the assessment. Multiply 640 by \$100 for a total of \$64,000 in assessment preparation costs. Add a \$48,000 assessment fee to the \$64,000 assessment preparation cost for a total assessment cost of \$112,000.
- Total Deployment Costs: Combine \$218,396 and \$112,000 for a total ISO 9001 deployment cost of \$330,396 for ISO 9001 registration.
- Total Life Cycle Benefits: Let's assume each of our four projects also build 10,000 SLOC software products. Now, we're ready to begin estimating the benefits. Let's assume each of our four projects has a 15% increase in maintenance savings, which is consistent with ISO 9001 experiences. Multiply 41,800 maintenance hours by 15% for 6,270 maintenance hours saved per project. Multiply 6,270 by 4 for a total maintenance savings of 25,080 hours. Now, let's assume each of our four projects has a 13% increase in productivity, which is consistent with ISO 9001 experience. Multiply 5,088 development hours by 13% for

661 development hours saved per project. Multiply 661 by 4 for a total development savings of 2,646 hours. Now, add the 25,080 maintenance hours saved to the 2,644 development hours saved for a total of 27,726 total maintenance and development hours saved. Finally multiply the 27,726 maintenance and development hours saved by \$100 for a total of \$2,772,600 in savings by using ISO 9001.

- B/CR: (The formula for B/CR is benefits divided by costs.) Therefore, divide \$2,772,600 by \$330,396 and the B/CR for ISO 9001 is 8:1.
- ROI%: (The formula for ROI% is benefits less costs divided by costs times 100.) Therefore, first subtract the \$330,396 in ISO 9001 costs from the \$2,772,600 in ISO 9001 benefits and divide the results by the \$330,396 in ISO 9001 costs and multiply by 100 for an impressive ROI% of 739%.

CMMI

CMMI is the newest set of minimum criteria for evaluating the systems and software engineering management capabilities of U.S. military suppliers. Once again, this is the latest derivative of Crosby's [29] Maturity Grid circa 1979 as an ITT employee.

CMMI has several distinguishing features. Primarily, it now applies to systems engineering management practices, rather than just software engineering

management. It has a stronger focus on the use of integrated product development teams. And, it comes in two forms:

- Staged: The staged model will continue to group and prioritize strategic process criteria by increasing complexity into five stages.
- Continuous: The continuous model enables acquisition and supplier personnel to group strategic process criteria on a case-by-case basis.

CMMI, unlike its SW-CMM predecessor, also has a stronger focus on the end-to-end systems engineering life cycle. This is a departure from the SW-CMM, which focuses on software management versus development processes. CMMI now begins to blur the line between supplier selection models and system engineering standards. (In addition, its individual specific practices will be treated more like requirements and less like guidelines as in the case of the SW-CMM's key practices.)

CMMI may very well have breached the gap between minimum supplier selection criteria and outright industry systems engineering standard. This is somewhat ironic as CMMI's alter ego, ISO 9001:2000, has now become more strategic, while CMMI has become more tactical.

Now, let's examine the dynamics of CMMI cost, benefit, and ROI analysis using Rachlin's [6] equations for B/CR and ROI%.

- **CMMI Policies and Procedures:** Let's begin by modeling the costs for implementing CMMI policies and procedures for four projects as a representative sample of a systems engineering organization. Rico [36] makes the following estimates: CMMI Level 2 requires 2,091 hours to develop 56 policies and procedures and CMMI Level 3 requires 3,771 hours to develop 101 policies and procedures. So, 5,862 hours are required to develop CMMI Level 2 and 3 policies and procedures. Multiply 5,862 by \$100 and that

comes to \$586,200. Half of this is software engineering, which amounts to \$293,100.

- **CMMI Evidence of Use:** Rico [36] also makes the following estimates:

CMMI Level 2 requires 10,304 hours to develop 138 products for four systems engineering projects and CMMI Level 3 requires 20,533 hours to develop 275 products for these projects. So, 30,837 hours are required to develop CMMI Level 2 and 3 products. Multiply 30,837 by \$100 and that comes to \$3,083,700. Half of this is software engineering, which amounts to \$1,541,850.

- **CMMI Implementation Costs:** Now add \$293,100 for CMMI

Level 2 and 3 policies and procedures and \$1,541,850 for CMMI Level 2 and 3 products for four projects, and the result is \$1,834,950 for software engineering.

- **Assessment Preparation Costs:** Let's estimate four projects of ten people in 20 indoctrination courses at 2 hours each which totals 1,600 hours. Let's similarly estimate four projects of ten people in 20 response conditioning courses at 2 hours, each which also totals 1,600 hours. Finally, let's estimate four projects of ten people in

Model	Estimation
Deployment Cost	(18,349 implementation and prep hours) x \$100 per hour = \$2,074,950
Assessment Cost	1,760 hours * \$100 per hour + \$64,615 fee = \$240,615
Life Cycle Benefits	(121,642 maint and develop hours) x \$100 per hour = \$12,164,200
Benefit/Cost Ratio	\$12,164,200 benefits / \$2,315,565 costs = 5:1
ROI%	(\$12,164,200 benefits - \$2,315,565 costs) / \$2,315,565 costs = 425%

Figure 10: ROI for CMMI

one 40 hour mock assessment or two 20 hour mock assessments for total of 1,600 hours. Now, let's add 1,600 indoctrination hours, 1,600 response conditioning hours, and 1,600 mock assessment hours for a total of 4,800 hours. Finally, let's multiply 4,800 by \$100 for a total of \$480,000 in assessment preparation costs. Half of this is software engineering, which amounts to \$240,000.

- **Total Deployment Costs:** Combine \$1,834,950 and \$240,000 for a total CMMI Level 2 and 3 deployment cost of \$2,074,950 for software engineering.

- **Assessment Cost:** For our four projects of five people, let's estimate 636 hours for the plan and prepare for appraisal stage. Let's estimate 1,018 hours for the conduct appraisal stage. And, let's estimate 106 hours for the report results stage. This totals to 1,760 hours. Multiply 1,760 by \$100 for an internal labor estimate of \$176,000.

Add an assessment fee of \$64,615 for a total assessment cost of \$240,615. (Assessment costs were based on labor distributions from Carnegie Mellon University [37].)

- **Total CMMI Cost:** Once again, take a deep breath and add the \$2,074,950 in total deployment costs to the \$240,615 in assessment costs for a total CMMI cost of \$2,315,565.
- **Total Life Cycle Benefits:** Let's assume each of our four projects also build 10,000 SLOC software products. Let's also assume that each of our four projects apply Inspections to satisfy their CMMI Level 3 goals. Now, we're ready to

begin estimating the benefits. Let's assume each of our four projects saves an average of 27,867 maintenance hours by performing Inspections for total maintenance savings of 111,466 hours. Now, let's assume our productivity doubles at CMMI Level 3 as with the SW-CMM, which results in a per project savings of 2,544 hours for a total of 10,176 development hours saved. Add the 111,466 hours in maintenance savings to the 10,176 hours in development savings for a total of 121,642 hours saved at CMMI Level 3. Multiply 121,642 by \$100 to arrive at an estimated savings of \$12,164,200.

- **B/CR:** (The formula for B/CR is benefits divided by costs.) Therefore, divide \$12,164,200 by \$2,315,565 and the B/CR for CMMI is 5:1.

- **ROI%:** (The formula for ROI% is benefits less costs divided by costs times 100.) Therefore, first subtract the \$2,315,565 in CMMI costs from the \$12,164,200 in CMMI benefits and divide the results by the \$2,315,565 in CMMI costs and

multiply by 100 for an impressive ROI% of 425%.

B/CR and ROI% differ from Rico [1]. This is partly due to a variation in assumptions based on refined data resulting from continuing CMMI analysis. Rico [1] assumed a lower cost for CMMI policy and procedure development while factoring in an estimate of system engineering savings. This resulted in a B/CR

for systems engineering while simultaneously factoring out the systems engineering savings resulting in a substantially lower B/CR and ROI% for CMMI.

In retrospect, both estimates may be based upon sound assumptions and may accurately model the costs of context-specific CMMI application and use. Therefore, it is assumed that variance in B/CR and ROI% estimates may

represent a valid range of estimated CMMI values for ROI.

COSTS/BENEFITS

This article, the section on Examples in particular, focused on some of the factors or drivers of SPI that were most sensitive to costs and benefits. By no means does this article attempt to exhibit an exhaustive scholarly study of the cost and benefit factors of

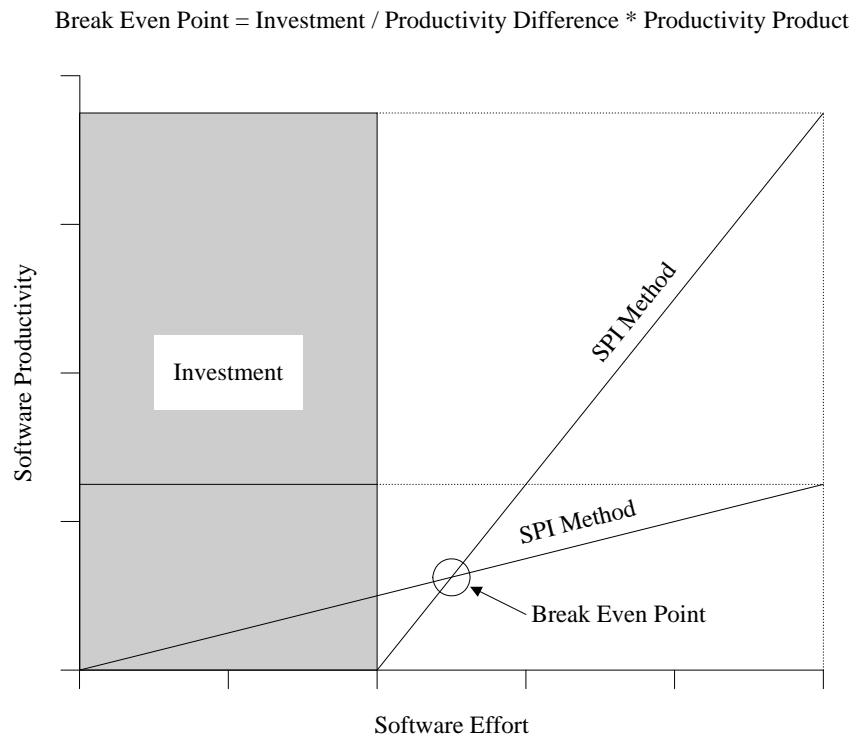


Figure 11: Breakeven Point Model

for CMMI of 11:1 and an ROI% of 1,044%.

However, this article now assumes the CMMI policy and procedure cost is doubled due to the added complexity of both systems and software engineering policies and procedures. This article then factors out the cost of CMMI policies and procedures

SPI.

However, this article is meant as a starter kit to help software managers and engineers begin to understand relevant models in ROI, sensitive cost and benefit factors, and how to combine the models with the factors to produce relevant estimates of ROI for SPI.

TYPICAL COSTS

The typical cost drivers or factors of SPI that are essential inputs into the ROI estimating process, include:

- Training fees, labor hours, and travel costs.
- Policies, procedures, processes, and life cycles.
- Project costs, activity costs, and administration.
- Documents, reports, records, and memos.
- Indoctrination costs of custom processes.
- Response conditioning costs.
- Mock appraisal costs and appraisal costs.

While, Rico [14] attempts to provide an in-depth analysis of the benefits of SPI, it does not similarly attempt an in-depth analysis of the costs of SPI. In order for any ROI estimate to be accurate, it must include an in-depth analysis of the total life cycle costs.

Many of the costs of SPI are hidden. Hidden costs for SPI include travel costs, overhead or fully-burdened hourly costs, and the costs associated with effort intensive SW-CMM and CMMI assessments.

The goal of accurate cost estimation is not necessarily to make estimates of ROI believable for publication as many assert, but rather to protect SPI initiatives from underestimation, which may hurt the application of an

approach to SPI. It's not as important to be believable, as it is to accurately budget SPI initiatives, and then accurately estimate the resulting ROI before, during, and after the deployment of the SPI method.

TYPICAL BENEFITS

While, miscalculating the costs of SPI is certainly a common pitfall, not understanding the benefits of SPI is a far more common malady inhibiting the field of SPI worldwide. Simply put, the benefits of SPI just aren't understood very well, and many assume SPI must proceed with or without benefits for some higher purpose that is not clearly understood. And, sadly, many leading researchers continue to refute the notion that SPI has any benefits at all.

Once again, this article is not intended to exhibit a thorough analysis of the benefits of SPI. It is, however, intended to point out factors sensitive to producing benefits, and show software managers and engineers how to exploit these factors for the purpose of estimating ROI. Typical benefits of SPI include:

- Higher quality (fewer defects).
- Lower maintenance (less rework).
- Higher productivity (low development cost).
- Faster cycle times (quick time-to-market).
- Greater value (more product features).

- Greater variety (more product variations).
- Higher customer satisfaction (more contracts).

DATA VALIDITY

No analysis of metrics and models for SPI is complete without the requisite plea for greater data validity. Some researchers have been complaining that practitioners don't collect enough data to substantiate their claims about the benefits of SPI. They often fail to recognize that potentially valid data are all around, and that it is not necessary to collect decades of data to begin making assertions about the costs and benefits of SPI. However, let's focus on some less politically self-serving issues of data validity like:

- Data Accuracy.
- Data Completeness.
- Benefit Isolation.
- Process Compliance.

DATA ACCURACY

Data accuracy and precision are very important, especially among small, resource-constrained, and fast-paced software organizations. Data accuracy isn't too much of a problem for extremely large organizations or extremely large programs ranging in the hundreds of millions or billions of dollars. In fact, it is pretty common practice for large organizations to issue stop-work orders on expensive SPI initiatives because the political climate isn't exactly right. Smaller firms are bound by

the constraints of guaranteeing that every dollar spent has some promising ROI. Factors affecting data accuracy include:

- Number of people.
- Number of hours.
- Training fees.
- Travel costs.
- Project and maintenance costs.
- Number, size, and variety of products.
- Estimated, actual, and residual quality.

DATA COMPLETENESS

Data completeness is closely related to data accuracy. However, data completeness has to do more with ensuring that all factors have been included, versus the precision or accuracy of any one value. Leaving out an important cost or benefit driver can have dire consequences on the outcome of a SPI initiative, especially for smaller, resource-constrained software organizations. Factors affecting data completeness include:

- Use of bottom-up versus top-down techniques.
- Creation of complete work breakdown structures.
- Inclusion of as many costs as possible.
- Use of fully burdened costs.

- Not forgetting training costs.
- Not omitting labor hours for training.
- Noting that 70% of assessment cost is internal labor.

BENEFIT ISOLATION

Benefit isolation is another term for ensuring the accuracy and validity of the benefits of SPI. It involves the use of experimental and survey research to quantify the benefits of a particular SPI approach. Experimental research is an effort intensive approach to using experimental control groups to

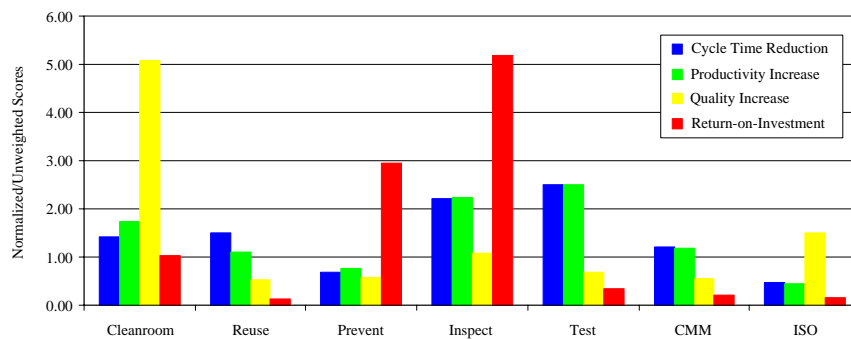


Figure 12: Choosing the Right Solution for SPI

ensure benefits actually occur. Survey research is less effort-intensive than experimental research, and often involves tempering quantitative results with qualitative objections. Approaches to benefit isolation include:

- Use of control groups to validate benefits.
- Identification of peripheral contributions to benefits.
- Exploitation of powerful cost and quality models.

- Retaining consultants to conduct benchmarking.
- Use of surveys to isolate benefit contribution.

PROCESS COMPLIANCE

Here's an interesting perspective on data validity. Even the finest experimental researchers fail to measure process compliance before attributing costs and benefits to causes. For example, researchers will often have software developers apply a particular SPI method without requisite training or measuring the degree to which the developer

applied the method. When, the SPI method doesn't yield the reported costs and benefits, researchers often claim there are no costs and benefits to SPI. Insignificant costs and benefits are more likely

due to inappropriate or even non-application of the SPI method altogether. Even the most primitive SPI methods yield impressive results when deployed correctly. Factors affecting process compliance include:

- Use of professional policy and procedure principles.
- Design of simple processes for maximum compliance.
- Aiming for high process compliance (especially when training is the primary deployment method).

- Measurement of process compliance.
- Noting that low process compliance will invalidate the results (good or bad).

ADVANCED ISSUES

There are just a few more considerations with respect to estimating ROI for SPI. These issues include:

- Estimating Breakeven Points.
- Choosing the Right Solution.
- Avoiding Common Myths.
- Using Comprehensive ROI Processes

ESTIMATING BREAKEVEN POINTS

The breakeven point is generally the place in time when an approach to SPI begins to yield its reported benefit. Technically speaking, the breakeven point occurs when the costs equal the benefits. However, breakeven points aren't necessarily time-dependent, but more often than not are dependent upon number of units produced. That is, how many units must be produced until the cost of production is paid for and the software organization begins to yield a profit.

Breakeven point analysis is a problem of linear optimization. The breakeven points for SPI are commonly believed to exist far out in the future, perhaps even years or decades beyond the initial application of SPI. While, breakeven point analysis is certainly a worthy topic for

complete exposition, it will not be analyzed in very much detail here. However, Rico [14] has an excellent exposition of breakeven point analysis, and often exhibits breakeven points of one or two hours of initial application for the best SPI methods.

CHOOSING THE RIGHT SOLUTION

Again, Rico [14] exhibits an in-depth analysis of the benefits of eight approaches to SPI:

- PSP.
- Cleanroom Methodology.
- Software Reuse.
- Defect Prevention.
- Inspections.
- Test.
- SW-CMM.
- ISO 9001.

The objective in pointing out Rico [14] is to stimulate the awareness that not all approaches to SPI are created equal. In fact, the best methods exhibit a B/CR of up to 1,300:1 over the worst but common approaches to software development. It is the fundamental responsibility of software managers and engineers to consider multiple approaches to SPI, and implement the best ones. It's time to stop implementing the worst approaches to SPI just because everyone else is doing it. This is especially true for commercial software organizations that have the latitude to implement the SPI methods that will yield optimal

benefit to cost ratios, rather than following standards out of mere popularity.

AVOIDING COMMON MYTHS

Unfortunately, many still believe that approaches to SPI are merely fanciful American fads that in fact have no benefits. SPI antagonists often believe:

- Software process improvement has no ROI.
- Process improvement takes a long time.
- Process improvement is too expensive.
- Process improvement can't be performed in a few hours, days, weeks, or months.
- Process performance can't be measured in only a few hours, days, weeks, or months.
- Process improvement is only for large, mission critical programs.

The field of SPI, while still in its infancy, is about to undergo a metamorphosis, and obsolete every approach to SPI mentioned in this article. At a very minimum, this article can serve as an archaeological record for comparison of outdated methods to killer approaches to SPI that have unprecedented levels of low costs and optimal benefits.

USING COMPREHENSIVE ROI PROCESSES

Also, Rachlin [6 and 18] are must-reads. The basic equations exhibited by this article are only the first step in the application of

scholarly ROI processes. Rachlin's ROI process is the de facto international standard for scholarly ROI analysis.

RECOMMENDATIONS

This is the most important part of this article. This section is one of discovery, reflection, and future direction. Again, many of the methods in this article reflect the early notions of the former century. The approaches to SPI which have yet to be discovered are the ones this article so vividly points to. These recommendations are a unique outcome of these analyses, and were not formulated in advance:

- Pinpoint High-ROI Factors: It's not necessary to identify every conceivable cost and benefit factor when producing early, top-down estimates of ROI. The law of diminishing returns applies here. There are only a few significant drivers of costs and benefits. Become familiar with them, and learn how to exploit them.
- Target High-ROI Approaches: This article is sufficient to point out the approaches to SPI, which yield the greatest benefits at the least possible cost. And, it gently reminds the reader that the best approaches are yet to come.
- Minimize Cost Incurrence: Choose a low-cost, low-risk approach to SPI. It's probably not wise to bite off more than one can chew. Selecting low-cost solutions to SPI can guarantee successful, early returns.

- Avoid Cost-Intensive Approaches: Don't be glutton for punishment. This article sufficiently exposes the approaches to SPI which are sure to drain your organization's assets. It is the reader's responsibility to understand the devastating effects of adopting the most expensive approaches to SPI.
- Avoid Training-Intensive Approaches: The market seems to have a process of natural selection built into it. Training-intensive approaches are generally unsuccessful in the marketplace because of their great expense, immense difficulty, and lack of sufficient tools for deployment beyond the classroom.
- Look for Low-Cost Automated Solutions: The future of SPI isn't in large overly-bureaucratic and manually-intensive approaches to SPI. The future is in low-cost, non-invasive automated tools that perform the software management tasks in spite of us. These are the tools that will leave an indelible mark on the 21st century.
- Use Professional Methods for Analyzing ROI: This article provides a valuable service by guiding readers toward relevant methods in ROI analysis and estimation. However, even the process of ROI is subject to low-cost automation. Don't get too wrapped-up in manual, laborious, and effort-intensive ROI processes. They're good reference tools, but look for low

cost automation to ROI analysis embedded in web-based project management tools.

BIOGRAPHY

David F. Rico is a software process improvement (SPI) consultant specializing in cost and benefit analysis. He helped design a \$250M software engineering toolset and the spacecraft software for NASA's \$20B space station in the 1980s, performed graduate studies under SEI Level 5 space shuttle managers, helped a \$40B Japanese corporation design a CMM self assessment tool in 1993, designed a software cost model for 37 kinds of U.S. Navy fighter aircraft, helped reengineer 36 logistics depots for America's largest foreign military customer, played key roles in the design of U.S. military intelligence satellite constellations, and has supported 15 software engineering process groups (SEPGs) over the last decade. He's been an international keynote speaker, published numerous articles, and holds a B.S. in Computer Science and a Master's Degree in Software Engineering (with 19 years of experience).

CONTACT INFORMATION

dave@davidfrico.com
http://davidfrico.com

BIBLIOGRAPHY

- [1] Rico, D. F. (2002). Software process improvement: Modeling return on investment (ROI). 2002 Software Engineering

- Institute (SEI) Software Engineering Process Group Conference (SEPG 2002), Phoenix, Arizona.
- [2] Kan, S. H. (1995). Metrics and models in software quality engineering. Reading, MA: Addison-Wesley.
- [3] Pham, H. (2000). Software reliability. Springer-Verlag.
- [4] Humphrey, W. S. (1995). A discipline for software engineering. Reading, MA: Addison-Wesley.
- [5] McGibbon, T. (1996). A business case for software process improvement (Contract Number F30602-92-C-0158). Rome, NY: Air Force Research Laboratory—Information Directorate (AFRL/IF), Data and Analysis Center for Software (DACS). <http://www.dacs.dtic.mil/techs/roispi2>
- [6] Rachlin, R. (1997). Return on investment manual: Tools and applications for managing financial results. Armonk, NY: M. E. Sharpe.
- [7] Lim, W. C. (1998). Managing software reuse: A comprehensive guide to strategically reengineering the organization for reusable components. Upper Saddle River, NJ: Prentice Hall.
- [8] Poulin, J. S. (1997). Measuring software reuse: Principles, practices, and economic models. Reading, MA: Addison Wesley.
- [9] Reifer, D. J. (2002). Making the software business case: Improvement by the numbers. Upper Saddle River, NJ: Addison-Wesley.
- [10] Sanders, J. (2001). SPI 10 Years On: Torturing the Evidence. 2001 Joint Euroforum/Dutch Software Process Improvement Network Conference (SPIDER 2001), Utrecht, Netherlands.
- [11] Diaz, M., & King, J. (2002). How CMM impacts quality, productivity, rework, and the bottom line. Crosstalk, 15(3), 9-14.
- [12] Humphrey, W. S. (2001). Winning with software: An executive strategy. Reading, MA: Addison-Wesley.
- [13] Billings, C., Clifton, J., Kolkhorst, B., Lee, E., & Wingert, W. B. (1994). Journey to a mature software process. IBM Systems Journal, 33(1), 4-19.
- [14] Rico, D. F. (2000). Using cost benefit analyses to develop software process improvement (SPI) strategies (Contract Number SP0700-98-D-4000). Rome, NY: Air Force Research Laboratory—Information Directorate (AFRL/IF), Data and Analysis Center for Software (DACS). <http://www.dacs.dtic.mil/techs/abstracts/rico.html>
- [15] Turban, E., & Meredith, J. R. (1994). Fundamentals of management science (6th ed.). Boston, MA: McGraw Hill.
- [16] Schuyler, J. R. (1996). Decision analysis in projects: Learn to make faster, more confident decisions. Upper Darby, PA: Project Management Institute.
- [17] Reifer, D. J. (2002). Making the software business case: Improvement by the numbers. Upper Saddle River, NJ: Addison-Wesley.
- [18] Snell, M. (1997). Cost benefit analysis for engineers and planners. London, England: Telford.
- [19] Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, 12(7), 744-751.
- [20] Humphrey, W. S. (1996). Using a defined and measured personal software process. IEEE Software, 13(3), 77-88.
- [21] Humphrey, W. S. (2000). Introduction to the team software process. Reading, MA: Addison-Wesley.
- [22] Humphrey, W. S. (1989). Managing the software process. Reading, MA: Addison-Wesley.
- [23] ANSI standard for quality management system—Requirements (ANSI/ISO/ASQ Q9001-2000). Milwaukee, WI: American Society for Quality (ASQ).
- [24] Carnegie Mellon University (2001). Capability maturity model[®] integration (CMMISM), Version 1.1

- CMMISM for systems engineering, software engineering, and integrated product and process development (CMMI-SE/SW/IPPD, V1.1) Staged representation (CMU/SEI-2002-TR-004 ESC-TR-2002-004). Pittsburgh, PA: Software Engineering Institute (SEI).
- [25] Rico, D. F. (n.d./2002). Software process improvement (SPI): The past vs. future of SPI methods [WWW document]. URL <http://davidfrico.com/spi-futurepdf.htm>
- [26] Siy, H. P. (1996). Identifying the mechanisms driving code inspection costs and benefits. Unpublished doctoral dissertation, University of Maryland, College Park
- [27] Rico, D. F. (n.d./2001). 524-page PSP 2.1 software life cycle [WWW document]. URL <http://davidfrico.com/psp-pols-procshtm.htm>
- [28] Humphrey, W.S. (2000). The team software processsm (TSPsm). (CMU/SEI-2000-TR-023). Pittsburgh, PA: Software Engineering Institute.
- [29] Crosby, P. B. (1979). Quality is free. New York, NY: McGraw-Hill.
- [30] Radice, R. A., Harding, J. T., Munnis, P. E., & Phillips, R. W. (1985). A programming process study. IBM Systems Journal, 24(2), 91-101.
- [31] Humphrey, W. S. (1987). A method for assessing the software engineering capability of contractors (CMU/SEI-87-TR-23). Pittsburgh, PA: Software Engineering Institute.
- [32] Paulk, M. C., Weber, C. V., Curtis, B., & Chrissis, M. B. (1995). The capability maturity model: Guidelines for improving the software process. Reading, MA: Addison-Wesley.
- [33] Rico, D. F. (n.d./2001). SEI level 2 thru 5: Cost model [WWW document]. URL <http://davidfrico.com/sw-cmm-costpdf.htm>
- [34] Diaz, M., & Sligo, J. (1997). How software process improvement helped motorola. IEEE Software, 14(5), 75-81.
- [35] El Emam, K., & Briand, L. C. (1997). Costs and benefits of software process improvement (IESE-Report 047.97/E). Kaiserslautern, Germany: University of Kaiserslautern, Fraunhofer-Institute for Experimental Software Engineering.
- [36] Rico, D. F. (n.d./2001). Capability maturity model integration: CMMI introductory overview [WWW document]. URL <http://davidfrico.com/cmmipdf.htm>
- [37] Carnegie Mellon University (2001). Standard CMMISM appraisal method for process improvement (SCAMPISM), Version 1.1: Method definition document CMU/SEI-2001-HB-001. Pittsburgh, PA: Software Engineering Institute (SEI).