Software Quality Journal, 8, 211–231, 1999
© 2000 Kluwer Academic Publishers. Manufactured in The Netherlands.

Technology Review: Adapting Financial Measures: Making a Business Case for Software Process Improvement*

WARREN HARRISON warren@cs.pdx.edu
Department of Computer Science, Portland State University, Portland, OR 97207-0751

DAVID RAFFO AND JOHN SETTLE {david,johns}@sba.pdx.edu
School of Business Administration, Portland State University, Portland, OR 97207-0751

NANCY EICKELMANN eickelmann@nasa.gov
WVU Software Research Laboratory, NASA IV & V Facility, Fairmont, WV 26554

Abstract. Software firms invest in process improvements in order to benefit from decreased costs and/or increased productivity sometime in the future. Such efforts are seldom cheap, and they typically require making a business case in order to obtain funding. We review some of the main techniques from financial theory for evaluating the risk and returns associated with proposed investments and apply them to process improvement programs for software development. We also discuss significant theoretical considerations as well as robustness and correctness issues associated with applying each of the techniques to software development and process improvement activities. Finally we introduce a present value technique that incorporates both risk and return that has many applications to software development activities and is recommended for use in a software process improvement context.

Keywords: Return on investment, business case for SPI, process improvement, ROI

1. Introduction

Software firms invest in process improvements in order to benefit from decreased costs and/or increased productivity sometime in the future. In order to make the business case for implementing a software process improvement effort the projected benefits must be weighed against the costs of implementing the program. Since software process improvement typically focuses on doing things "right the first time," the benefits may not accrue for months or even years, though the costs are usually incurred immediately. When evaluating future benefits, it is prudent to consider both the *timing* (i.e., how long until we start to see the benefits?) as well as the *risk* (i.e., how likely are the actual future benefits to vary from their projected values?

*This work has been funded in part under NASA Contracts NCC 2-1152 and NCC 2-979 in conjunction with NASA's IV & V Facility, Fairmont, WV.

In section 2, we review some of the main techniques from financial theory for evaluating the risk and returns associated with proposed investments. We also discuss significant theoretical considerations associated with applying each of the techniques, as well as the robustness and correctness of utilizing each of the techniques in practice. In section 3, we explore how these techniques might be applied to software development and process improvement activities. Using this background as a starting point, we introduce a present value technique that incorporates both risk and return in Section 4. This technique has many applications to software development activities and is recommended for use in a software process improvement context.

2. Financial measures in planning

Planning implies that an effort or activity is being considered for future implementation. The proper financial analysis of a candidate action requires not only looking at the anticipated costs and returns, but also the timing of the returns and the distribution of possible outcomes for those returns. The issues of timing and risk are typically addressed using "present value" analysis methods that attempt to convert future returns into their equivalent "present values."

In this section we discuss the subject of valuing projects by reviewing the various techniques available from financial theory and discussing some of the practical considerations associated with applying them. These concepts form the foundation of the contemporary study and application of finance (Brigham and Gapenski 1997) and (Brealey and Myers 1996). However, we have adapted these techniques to apply within the specific context of software development.

2.1. Present Value (PV) based methods—general discussion

Present Value (PV) based methods are conceptually correct, and are therefore believed to be the best investment planning methods. PV based methods (including NPV, IRR, Profitability Index or Cost-Benefit Ratio, etc.) value a project at the discounted value of all the incremental returns attributable to its adoption. Discounting reflects both the timing and the risk of the projected returns. By converting projected future returns into "present value equivalents," *projects with differing return patterns can be compared.*

In traditional financial applications, returns are typically the funds that the project generates, or their *cash flows*. *Incremental cash flows* are those cash flows that would not have occurred were the project not undertaken. In general, the utility received from software process improvements is not in terms of "cash generated," but rather cost savings. Usually this involves effort savings from rework avoidance, but it may also stem from reduced support costs, less maintenance effort, reduced mission risk, etc. It is important to understand that cost savings are interpretable as incremental cash inflows since a cost that goes away increases overall cash inflow. In this paper, we will simply refer to a project's "returns" and not make any distinction between cost savings and cash generated.

2.1.1. ***Discounting for time.*** Returns that are realized sooner are considered more valuable than returns that are realized later. Exactly how much more valuable is a function of the "discount rate," which is a factor that quantifies the "time value of money." The present value of a return n periods in the future is computed using the periodic discount rate r, as:

$$\text{Present value} = \text{Return}/(1 + r)^n$$

This is known as "discounting." While the discount rate is most often expressed in terms of annual discount rate, any rate, be it annual, quarterly or monthly can be used, as long as the periods by which the investment is discounted are in the same units.

Example 1 illustrates the effect of differences in the timing of returns. Even though one choice is projected to return \$150,000 less in cost savings than the other choice, it is considered preferable because its returns accrue after only one year rather than three years, as does the other choice.

Example 1. A project manager has the option of either purchasing a new testing tool for \$250,000 or using the same resources to hire and train additional Test Engineers. It has been projected that the new testing tool would provide \$600,000 in cost savings within a year by automating several aspects of the testing effort. The effort savings would allow fewer testers to be assigned to the project. Thus, it is not expected that any defects would be found that would not be found with the current testing staff. However if the manager chose instead to hire additional Test Engineers, at the end of three years, it is expected that the additional staffing will be responsible for \$750,000 in rework cost savings by finding additional defects prior to release. Assuming an annual discount rate of 15%:

Testing Tool—\$600,000 in one year

$$PV_{\text{tool}} = \$600{,}000/(1.15)^1 = \$521{,}739$$

Hire Testers—\$750,000 in 3 years

$$PV_{\text{hire}} = \$750{,}000/(1.15)^3 = \$493{,}421$$

Under these assumptions, the tool option would be the preferred course of action since the returns, even though they are substantially less, accrue within a year rather than waiting for three years.

2.1.2. ***Recognizing on-going returns.*** The discussion so far assumes that investments will yield a single return at some point in the future. However, in the case of software process improvements, the cost of implementing a given initiative may very well yield on-going returns, because the improvements may affect all future software projects.

Extending the concept of Present Value from a single future return to an on-going stream of future returns is fairly straightforward. In its simplest form, we can simply total up the Present Value of each future return as in Example 2. This example continues to make the assumption that each individual project has a single return received as a "lump sum," but that the costs invested in implementing the process improvement for Project 1 also benefits Project 2. Consequently, the improvement yields two distinct returns, one in Year 2 and one in Year 3.

Example 2. A code reading initiative is expected to yield returns of $100,000 in reduced rework two years in the future on one project, and another $100,000 of reduced rework three years in the future on another project. The Present Value for this initiative at a discount rate of 10% is:

$$PV_{\text{project1}} = 100{,}000/1.10^2 = 82{,}645$$

$$PV_{\text{project2}} = 100{,}000/1.10^3 = 75{,}188$$

for a total Present Value of $157,833.

The computation can become tedious if the proposed process improvement continues to offer costs savings for a long time (say the life of the organization). In a case of a relatively long stream of reduced rework costs, the returns can be considered as a perpetual annuity, whose Present Value is computed as:

$$PV_{\text{annuity}} = \text{Return}/\text{discount rate}$$

At a periodic discount rate of 15% the Present Value of a perpetual annuity and the sum of a flow of constant returns every period the two methods converge after about 35 periods. This is essentially saying that returns more than 35 periods in the future really don't add much in terms of Present Value.

2.1.3. Discounting to reflect risk. Just as projects that do not provide a return until many periods into the future are less valuable than projects that start to payoff sooner, projects whose projected payoff is more certain are more valuable than projects whose return is less certain. The discounting process likewise accommodates this differential.

Typically an overall assessment of the risk of the return is made, then a premium is added to the risk-free discount rate to reflect the uncertainty in the projected return. This yields the effective discount rate.

The discount rate k is determined by $(k = r_f + \phi)$, where r_f denotes the risk-free rate and ϕ denotes the premium for the amount of risk taken. The risk-free rate reflects only the time value—i.e., the forgone "enjoyment" of not having the resources available for a given period of time. On the other hand, ϕ is related to the degree of risk—in this context, risk means not receiving the cost savings that we had originally anticipated. Consequently, discounting simultaneously adjusts returns for two essentially different burdens: *time* and *risk*. The present value of a project is then:

$$PV = \sum \text{Return}_i/(1 + k)^i$$

where Return_i is the cash flow at period i and k is the effective periodic discount rate. Example 3 adds a risk premium to the scenario reflected in Example 1, and illustrates the integration of time *and* risk when evaluating projects. As can be seen in Example 3, adjusting the discount rate to reflect uncertainty of returns, changes the outcome of the analysis in a specific, quantifiable way.

Example 3. The projection of the returns we can expect from introducing a new testing tool may become uncertain after speaking with a number of current users of the tool. If the estimate of returns from acquiring a new tool in Example 1 is in question, we may wish to assign a 15% risk premium to the discount rate for PV_{tool}. Assuming a 15% base (risk-free) discount rate as before and also a 15% risk premium for selecting the testing tool, we end up with a combined discount rate of 30% for the testing tool (but just 15% for the additional test engineer choice, since it is ostensibly less risky):

Testing Tool—\$600,000 in one year

$$PV_{\text{tool}} = \$600{,}000/(1.30)^1 = \$461{,}538$$

Hire Testers—\$750,000 in 3 years

$$PV_{\text{hire}} = \$750{,}000/(1.15)^3 = \$493{,}421$$

Due to the increased uncertainty in the returns we expect from the tool, its desirability is reduced. However, what is notable about this example, is that its desirability is reduced by a specific quantifiable amount.

2.1.4. Reason for conceptual superiority of Present Value methods. The idea behind Present Value is that if you have an amount of cash equal to the Present Value, you could (it is assumed) invest that cash in a competitive financial market instrument for a period equivalent to the life of the project. This investment would then provide comparable returns at the market rate. Therefore, the project would need to have returns at least equal to the financial market; otherwise, so the reasoning goes, why choose the project—why not simply invest the resources in the market?

This conceptual advantage continues as we consider risk as well. When participating in a competitive financial market, the investor always has a wide variety of investment opportunities of varying risk from Government Bonds to investments in internet start-ups. Of course, as the risk increases, so does the return expected by investors. Thus, when determining the present value of a project, the discount rate should reflect the level of risk inherent to the project returns.

2.1.5. Shortcomings and difficulties. The Present Value concept depends upon a number of assumptions. We discuss some of these key assumptions below. It should be noted that the assumptions described below are common to many methods used to assess the value of real projects.

- *The "right" discount rate is hard to find.* Most discussions of discounted cash flow techniques assume that the "right" discount rate is obvious. However, determining both the appropriate risk-free discount rate and the appropriate risk premium are very hard problems. Most techniques to deriving discount rates [such as the Capital Asset Pricing Model, or CAPM—see Brigham and Gapenski (1997) or Brealey and Myers (1996)] assume a readily accessible system, such as the stock market to derive appropriate costs of capital and risk. No such systems currently exist for tactical-level decisions such as investing \$100,000 in a process change. Nevertheless, because the collection of such decisions in a firm has consequences for its overall risk and return, it is possible to argue for certain values of the discount rate k, as we do in the final section.
- *The discount rate is seldom static over time.* In the earlier discussion, the discount rate k is a uniform discount rate applied equally to all periods. This takes a simple approach to risk—one which is probably too simple in most situations. There can be complex patterns of risk (e.g. contingencies, or patterns where risk is reduced over time). This can be addressed by applying different discount rates to different periods, on the grounds that risk is encountered unevenly over the life of the project. For example, the initial two years might be the riskiest because critically defining events are expected to resolve much uncertainty over that length of time. After two years, the project returns will stabilize and become more certain (for better or worse, depending on which way the two years go). In this case, we could discount all returns past two years to the two-year mark using a relatively low discount rate k_b, then discount the results, and the one- and two-year returns at a higher rate k_a. That is, for periods 3 though the life of the project calculate V_b, the present value of the return discounted at k_b over periods $3 \ldots n$, and add to the present value of the first two periods discounting everything at k_a.

$$V_b = \sum_{i=3}^{n} \text{Return}_i/(1 + k_b)^{i-2}$$

$$\text{PV} = \text{Return}_1/(1 + k_a)^1 + \text{Return}_2/(1 + k_a)^2 + V_b/(1 + k_a)^2$$

This can be made much richer by allowing future returns to take on values contingent upon earlier outcomes and decisions that result from those outcomes. For example, \$1 million may be spent in an effort to assess the promise of a new process improvement. After a year, if the proposed improvement is likely to be unsuccessful, it can be abandoned, limiting the loss to the \$1 million. If it is successful, more may be invested into full-blown institutional adoption. This kind of contingency has long been modeled using decision trees. Recently, the contingency pricing principles learned from stock option models have been brought to bear on the problem.

- *Returns can be difficult-to-impossible to project.* Present Value methods rely on the ability to quantify benefits and costs. The further into the future we go, the less likely it is that the consequences may even be guessed. In many cases, it is the uncertainty of the projections as opposed to any inherent risk of the investment that increases the risk premium.
- *It is often difficult to assess intangible benefits.* For many process improvements or other activities, benefits and costs accrue that are difficult to quantify. For instance, many software process improvement advocates point to its intangible benefits (Eickelmann 1999).
- *It is often difficult to know what costs are truly incremental.* In addition, to the above considerations is the problem of determining what costs are truly incremental. Cost allocation systems for overhead, physical plant, heating, etc. often allocate currently existing costs through a complicated cost accounting structure. These are not actually incremental costs since they exist whether the project is carried out or not (although the project may influence the future additions of such costs). When measuring performance after the fact, one has the additional difficulty of not knowing what *would* have happened had the opposite decision been made.

2.2. *PV-based methods described*

Most contemporary financial analysis techniques used to support project planning methods (these are typically known as Capital-Budgeting techniques) are based on Present Value.

2.2.1. Net Present Value (NPV). Net Present Value (NPV) in a very real sense values the project as an asset of the organization. It considers the project as a "money factory" which will produce a stream of returns; when discounted, these returns produce a present value. Subtracted from that amount is the cost of carrying out the project. The difference is the *Net Present Value* of the project.

In short, NPV is computed as:

$$\mathrm{NPV} = \sum_{i=1}^{n} \mathrm{Return}_i/(1+k)^i — \mathrm{I}$$

where I represents the initial investment cost of the project. This is illustrated in Example 4.

Example 4. A code reading initiative is expected to cost $50,000 to institute. These costs involve training, infrastructure support, etc. The incremental returns (i.e., the rework savings after the cost of performing the code readings is subtracted out) of this improvement are expected to total \$100,000 of reduced rework two years in the future. The Net Present Value for this initiative at a discount rate of 10% is:

$$\begin{aligned}\text{NPV} &= 100{,}000/1.10^2 - 50{,}000\\ &= 32{,}645\end{aligned}$$

Net Present Value is by far the most universally accepted approach to capital budgeting.

2.2.2. Internal Rate of Return (IRR). Internal Rate of Return (IRR) is perhaps the most misunderstood measure of projected project performance. IRR is defined as the discount rate in the NPV equation that causes the calculated NPV to be zero (it is *not* the Return on Investment).

Because IRR is a rate of return measure (albeit a measure of projected rate of return), it is quite popular. Executives seem to like to think in terms of rates of return. Perhaps the reason for this preference is that rates of return are more directly comparable to identified or imagined market rates of return. NPV is less direct: since the decision-maker is required to specify the market opportunity cost of capital, k. The project is expected to return this rate k on an amount equal to PV. Since one need only invest PV, the Net Present Value is the profit or loss immediately returned from going ahead in addition to being paid back for the initial investment at the specified discount rate. This is illustrated in Example 5.

Example 5. The code reading initiative from Example 4 is expected to cost \$50,000 to institute. The returns on this improvement are expected to be \$100,000 of reduced rework two years in the future. The internal rate of return for this initiative is:

$$\begin{aligned}\text{IRR} &= 41.4\%\\ \text{NPV} &= 100{,}000/1.414^2 - 50{,}000\\ &= -0\end{aligned}$$

A project is considered acceptable if its IRR is in excess of the market rate of return on comparably risky investments. If two or more projects are being compared in a situation where only one can be done, the temptation is to take the project with the highest IRR. Although this seems to make sense, it runs into difficulties so serious as to be disabling, as discussed below.

IRR is considered a time-value measure because (a) it deals with returns and their timing and (b) it obtains a rate of return measure that can be compared against rates available on comparably risky investments. These attributes lend it some strength. However, it has several weaknesses, all of which are somewhat subtle. The upshot is that IRR measures can be remarkably misleading.

- *IRR is not necessarily a unique measure for a given project.* Close examination of the NPV formula reveals that it is a polynomial equation of degree n in k. This means that it can have as many as n solutions, depending on how often the polynomial crosses the x-axis (NPV $= 0$) when NPV is plotted against k.

  Consider a simple example where the IRR is either 0 or 100%: The investment is \$100 (return at time 0 $= -100$); the cash flow at Time 1 is \$300; and the cash flow at Time 3 is $-\$200$. It is easy to see that this entails a zero rate of return, because the undiscounted cash flows add up to zero. Here is how it may be seen to be 100%: One invests \$100 at time 0; at Time 1 this accumulates at 100% to \$200; one then withdraws \$300, leaving a deficit of \$100; this deficit grows to \$200 at Time 3, when it is paid off. The problem here, and the reason for the blow to intuition, is that in the first period we are "lending to the project," and would naturally like to get as high a return as possible. However, in the second period, we are "borrowing from the project," and would like to get by with as low a return as possible. That is, it is attractive to have IRR $> k$ only if we are "lending to the project." This is normally the case with investments, but situations crop up in which there is confusion about whether one is a borrower or lender, or a mix of both.

  The situation of alternating positive and negative returns often occurs during a software development project when evaluating the impact of process changes. Typically, changes to the process provide alternating effort increases for improved V & V activities and savings later on for various costs that are avoided. Moreover, when comparing two different projects or process alternatives, it is quite common to discover that the sequence of cash flows representing the *difference between* the two projects can alternate from positive to negative quite capriciously. In short, the ambiguity between borrowing and lending situations makes the application of IRR to many situations quite problematic.
- *IRR will not give good answers when the scale of two mutually exclusive investments differs.* IRR may induce the decision-maker to take on a small project with a high IRR in preference to a much larger project with a lower IRR—even if the larger project creates more value. The larger project might still have an IRR greater than k, and therefore present the opportunity to put a good deal more money to work at a rate better than that available in the market (*for a comparable level of risk*). NPV will directly measure the value added. In short, IRR shows the size of the fish relative to the pond (k) but ignores the size of the pond (I).
- *IRR is an internal rate of return.* If we don't want to receive the returns on the schedule that we project to have them arrive, then we will need either to borrow or reinvest. Suppose for the moment that we are interested only in the highest accumulation of returns as of n, the end of the project. Then as returns come in, they need to be reinvested. At what rate can they be reinvested? If one wants to maintain the same degree of risk, then the answer is k. But then the overall rate of return on the project, from Time 0 until Time n, will be a blend of IRR and k (lower than IRR but higher than k).

  This blended rate is referred to as the Modified Internal Rate of Return, or MIRR. It can be shown that the definition of MIRR is the rate k for which the future value of the returns is equal to the investment compounded to the end of the project life: i.e.

$$\text{MIRR} = \sum_{i=1}^{n} \left(\text{Return}_i(1+k)^{n-i}/I\right)^{1/n}$$

where k represents the cost of capital. MIRR gives results equivalent to NPV, but rather than show value created, it shows extra return earned on a given investment. Therefore, it retains all of the conceptual superiority of NPV except for the precise value measure.

*2.2.3. **Profitability Index.*** Profitability Index (PI) is variably defined as PV/I. It simply takes PV and scales it to the initial outlay, to get a kind of PV "bang-for-the-buck" measure as shown in Example 6.

Example 6. The code reading initiative from Example 4 is expected to cost $50,000 to institute. The returns of this improvement are expected to be $100,000 of reduced rework two years in the future. The Net Present Value for this initiative at a discount rate of 10% is:

$$\begin{aligned} \text{PV} &= 100{,}000/1.10^2 \\ &= 82{,}645 \end{aligned}$$

for a Profitability Index of 82,645/50,000 or 1.65.

The Profitability Index is appealing to managers who must decide between many competing process improvements with positive NPVs, but who have limited investment resources. The idea is to take the process improvement options with the highest PI first, until the investment budget runs out. This is not bad, but can sub-optimize the investment portfolio. Rank ordering by PI may rule out a large project with a good NPV because it is just over the limit. For example:

Project	Investment	NPV	PI
A	100,000	30,000	1.3
B	100,000	25,000	1.25
C	300,000	60,000	1.20
D	200,000	20,000	1.1

If the capital budget is $400,000, the Profitability Index ranking technique will pick A and B first, leaving inadequate resources for C. Therefore, D will round out the budget, and the overall NPV will be 75. However, using an integer programming approach will recommend taking projects A and C for a total NPV of 90.

PI also suffers from the scale problem, in a way analogous to IRR.

Overall, PI is recommended as a secondary measure used to augment NPV to help optimize the allocation of investment dollars.

2.3. *Non-present value-oriented methods*

These methods are reported because, in spite of serious conceptual weaknesses, they are often used in practice. The reasons for their popularity vary, but it is generally because they are simple and easy to explain to decision-makers without a sophisticated financial framework.

2.3.1. Payback. Payback is the time it takes to get the initial investment back out of the project. Projects with short paybacks are preferred, although the cutoff time is completely arbitrary. The intuitive appeal is reasonably clear: it is easy to calculate, communicate, and understand. Because of this, it is the least likely to confuse managers without a formal background in financial issues.

However, if payback period is the only criterion used, then there is no recognition of any cash flows, small or large, to arrive after the cutoff period. Furthermore, there is no recognition of the opportunity cost of tying up funds.

Inasmuch as discussions of payback tend to coincide with discussions of risk, it is probable that a short payback period is viewed as a way to respond to higher risks. However all evaluation criteria are arbitrary.

2.3.2. Average Rate of Return (ARR). ARR can stand for either "Average Rate of Return" or "Accounting Rate of Return" because traditionally, it considers accounting net income rather than cash flows. It does this by including depreciation/amortization in the computation of the cash flow (meaning it treats depreciation as an out-of-pocket expense, reducing the cash flow). Its proponents argue that ultimately publicly traded organizations are evaluated on accounting measures of net income (which is reduced by depreciation and amortization), and so capital budgeting techniques should be based on this approach to determining cash flow.

There are a variety of different ARR measures based on some notion of benefits divided by some notion of investment. Here are some example definitions, the first one being the one normally mentioned in textbooks.

Average projected after-tax profit/Average investment
Average projected after-tax profit/Initial investment
Average projected after-tax cash flow/Average investment
Average projected after-cash flow/Initial investment

Profit and investment can have varying definitions, so various accounting adjustments may or may not be made.

2.4. *Summary of planning measures*

Table 1 summarizes the various planning measures, their definitions, investment acceptance criteria as well as significant problems associated with the measure.

Table 1. Summary of planning measures

Criterion	Definition	Accept if:	Problems
Net Present Value (NPV)	PV − I	NPV > 0	
Internal Rate of Return (IRR)	Discount rate such that NPV = 0	IRR > r	Scale Different results with alternating cash flows Reinvestment rate
Profitability Index (PI)	PV/I	PI > 1	Scale
Payback	Years to recover I	< cutoff	No time value Arbitrary cutoff No credit beyond cutoff Scale problem?
Average Rate of Return (ARR)	Average income/ average I	> some standard	No time value Not cash flow Arbitrary standard

I = initial investment.

Clearly the approach with the fewest difficulties is Net Present Value. This is evidenced by NPVs cross-industry popularity in practice.

3. Application of financial planning methods to software process improvements

The motivation for software process improvement is better quality, more timely delivery and reduced effort. In order to make a business case for proposed process improvement activities, it must be shown that the value received is greater than the cost of the activity.

The value of increased quality due to software process improvements mainly derives from cost savings due to rework avoidance, though many other benefits have been documented, such as reduced time-to-market, improved customer satisfaction, etc. Early prevention of defects reduces the cost of rework later in the life cycle and the practice of reuse improves productivity (Dion 1993; Lipke and Butler 1992). Where the cost and value of a given process improvement can be accurately quantified, the application of financial decision-making models can be straightforward. In this discussion, it will be assumed that an accurate quantification of the costs and returns is available.

3.1. *Present Value based approaches*

The use of Present Value based models has been addressed in various aspects of software engineering in the past, such as quality assurance (Slaughter et al., 1998), software maintenance (Vienneau 1995) and to assess the impact of process changes (Raffo 1996; Raffo and Kellner 1999). The cost savings derived from rework avoidance due to software process improvements can be viewed as positive cash flow. In order to implement process improvements, there are incremental costs. These costs can be considered an “investment.” Thus we have the classical investment model:

investment $\rightarrow$ future cash inflow

3.1.1. Net Present value. Net present value is an ideal method for valuing a particular software process improvement effort. This is because most such efforts involve performing work “up front” (i.e., “investing”) for a future payback. Present Value techniques are specifically designed to consider the pertinent factors—investment, returns, the timing of the returns and risk. Present Value and Net Present Value techniques are well suited to handling the complex returns sequences associated with real tradeoff decisions where future incremental returns can be either positive or negative and their magnitudes and timings can be uncertain.

The monetary value of the return itself may not be particularly significant within the context of a team or organization with a fixed budget allocation. For instance, it would be the rare manager who would voluntarily offer to return a portion of his budget if he found that it could be better spent elsewhere. However, NPV can be used as a relative measure to compare alternative projects to determine the most effective allocation of a budget. An ever-present fact given a finite budget is that dollars spent in one place are not available to be spent elsewhere. Present Value analysis can give some idea of the “bang for the buck” the user gets from a particular effort.

In most treatments of NPV, the discount rate is taken to be a given. However, as discussed earlier, the discount rate is comprised of two parts: a base, “risk-free” rate which reflects the value of returns to be received in the future with certainty, and a “risk premium” that reflects the uncertainty of a future pay-off.

In the case of a software development activity such as a requirements review, the risk derives from the fact that the benefits of the activity are not a “sure thing.” They may be either more or less beneficial than we expect. For instance, there may be few errors or problems to find, the review process itself may be ineffective, or the follow-through may be lacking so even when a problem is found, it is not addressed. The bottom line is that in spite of the effort expended in the activity, the amount of rework avoided may or may not be what had been projected.

Obviously, the more risky the investment, the higher the risk premium and the greater the discount rate, since a rational investor will expect a greater potential payback from a riskier investment.

3.1.2. Using NPV for selecting among software process improvement efforts. In order to use a Net Present Value based approach to compare software process improvement opportunities, five capabilities must be present:

1. the ability to predict the cost of implementing the improvement,
2. the ability to predict the expected return of the improvement,

3. the ability to predict the timing of the expected returns of the improvement,
4. the ability to quantitatively assess the "risk" involved in the improvement's returns,
5. the ability to establish a reference risk premium (or, equivalently, a reference cost of capital) for the firm's "typical" or reference project.

3.1.3. Predicting cost. Predicting the costs of a process improvement can be challenging. It is easy to miss certain costs. For example, when implementing a process change, there are a number of costs associated with implementing the change. Raffo (1996) provides a list of general cost categories encountered when implementing process changes. These cost categories include: causal analysis meetings, process definition activities, dissemination and training costs, tool costs, communication and motivation costs, coaching costs, and verification and enforcement costs. In a case study at a leading software development firm, these costs were significant (422 staff hours for verification and enforcement costs for one process change). These costs did not include the costs for executing the process change.

Many times some or all of these costs are overlooked or estimated, inaccurately causing potential errors in the economic analysis. Moreover, standard metrics programs typically do not capture many start-up or implementation costs.

Good data may even be difficult to obtain for core process performance measures. For instance, although staff effort is one of the key components of software project cost, companies often collect effort data that is designed to suit accounting and payroll purposes rather than performance assessment. As a result, companies often have effort data rolled up to reflect monthly totals for an entire project. This data is unlikely to support more than gross estimates for the performance of any given development activity.

3.1.4. Predicting returns and timing. Predicting the returns and their timing can be more difficult than predicting the costs. Nevertheless, there is no dearth of attempts at assessing the benefits of various software process improvements. For instance, in McGibbon (1996), the benefits of inspections (as well as the benefits of a variety of other process improvements) are modeled as a function of rework costs (RC):

$$\mathrm{RC} = \mathrm{R} \cdot \sum d_i t_i$$

where d_i is the number of defects detected in phase i, t_i is the amount of time in hours to detect and fix an error in phase i, and R is the average cost per hour to find and fix an error.

The key to making a good economic analysis is to have either good historical data within your organization, or access to "industry standards" (which won't quite fit your own case). For instance, O'Neill (1989, 1995, 1996) provides interesting data pertaining to inspection preparation and meeting costs. In addition, he observes that inspections typically detect 10–20 errors per thousand lines of code.

However, in order to determine the return associated with detecting these errors earlier in the life cycle, we need to know what would the cost of these errors be if they were either (a) found later or (b) not found until after the product was released. It is often difficult to determine the return because little industry data exists as to the actual costs of correcting defects later in the life cycle, and organizations seldom tend to keep track of such data.

Also, in practice we can not always be certain what the timing of the returns would be—for instance, should the benefit of finding an error early through inspections accrue when the error is found? When the product is undergoing testing (which is when the error might otherwise be found and corrected)? When the product is released? As our earlier examples have shown, deferring the returns can have a big impact on the perceived value of the process improvement when making investment decisions.

4. Risk and the reference risk premium

Anyone who has ever spent time looking at past process data understands that real outcomes seldom fit the nice, mathematical prediction models developed by researchers. However, these models often provide a good basis for an expected outcome, with potential outcomes being clustered around that point. A good source of data on past experiences can go a long way towards understanding what the clustering looks like. Lacking quantitative, organization-specific historical data, other sources of information can be tapped to assess risk. For instance, consultants, small, ad hoc experiments, expert judgement, etc.

Selecting an appropriate discount rate for Net Present Value is very important. However, when using NPV to *compare alternate projects*, we find the following situation:

$$k_1 = r_f + \phi_1 = r_f + p(\text{Risk}_1) \text{ for project 1}$$

$$k_2 = r_f + \phi_2 = r_f + p(\text{Risk}_2) \text{ for project 2,}$$

where ϕ is the risk premium for a project of Risk_j; p is the "price of risk," i.e., the additional risk premium needed for each unit of risk; and Risk_j is the measure of units of risk for project j.

The risk free rate, r_f is constant for both projects—the only thing that changes is the risk premium ϕ_i if the projects are of unequal risk. If we have a reference cost of capital (say, k) that is used for typical, i.e., average-risk projects, then we can judge the risk of the project we are considering relative to the reference, and adjust k accordingly. That is, the risk premium is the single distinguishing factor in k.

In many practical situations, nonfinancial criteria are used in mixture with financial criteria. For example, a space shuttle mission places great value on success and safety as well as time, cost, and efficiency. Some may argue that even success and safety are ultimately measured in dollars; however, (a) this is often difficult, and (b) it is controversial, (although becoming less so).

Relative financial performance of processes, tools and activities within the context of the mission is therefore more appropriate than *absolute* financial performance. This is because alternative financial markets are not available to shift investments into at a "market rate" of return. Thus, the "alternative" use of resources that is typically implied in most financial discussions is not a factor. On the other hand, efficient use of resources within the context of the mission means that more resources will be available for use across activities, resulting in improved mission effectiveness. In this case, *efficiency is effectiveness*. This idea is supported in Quirk and Terasawa (1987), which discusses public sector discount rates. In many cases, the opportunity costs are constrained to alternative uses within a particular government agency.

We conclude therefore, that adjusting the discount rate k for relative risk can help maximize the efficiency of resource allocation for software process improvement activities (Harrison 1999a,b).

In order to determine the risk of a process improvement activity (and hence the appropriate risk premium) we need to establish how variable the project returns might be. This variation gives an idea how risky a given effort might be. Example 7a illustrates the variability that may be encountered when performing a code inspection and summarizes that variability using a Standard Deviation of the possible outcomes.

Example 7a. When performing a code inspection each flaw found during the inspection will result in avoiding a future correction that costs \$1,000 two years from now. Based on past data, we believe that there is a 10% chance we'll find no flaws, a 20% chance we'll find 10 flaws, a 40% chance we'll find 20 flaws, a 20% chance we'll find 30 flaws and a 10% chance we'll find 40 flaws. This is summarized as follows:

Flaws Detected	Probability	Costs Saved	PV × 10%
0	10%	0	0
10	20%	10,000	8,264
20	40%	20,000	16,528
30	20%	30,000	24,793
40	10%	40,000	33,057

The expected cost savings discounted at a risk-free rate of 10% is \$16,529 with a standard deviation (i.e., $\text{sqrt}(\sigma^2 = \Sigma(\text{PV}(X) - \text{PV}(\mu))^2 \times p(X)))$ of \$9,696.

Taken in isolation, it is unclear whether a given project is of average risk, less than average risk or more than average risk. In order to develop a risk premium, a project should be compared to alternate projects. However, if we assume that the code inspection project is of "representative risk"—i.e., represents the typical risk of process improvement efforts within our investment environment the code inspection activity might be used as a reference for the risk of other projects. Suppose we have the alternative in Example 7b.

Example 7b. The \$10,000 from Example 7a is spent on helping develop unit test plans rather than code inspections. Every flaw that is found due to the improved unit test plans again saves \$1,000 in future rework two years in the future. Because we know a little more about this situation, we may feel the Projected Returns are more reliable:

Flaws Detected	Probability	Costs Saved	PV × 10%
0	5%	0	0
10	10%	10,000	8,264
20	70%	20,000	16,528
30	10%	30,000	24,793
40	5%	40,000	33,057

The expected discounted cost savings is still \$16,529 but now it has a standard deviation (i.e., sqrt($\sigma^2 = \Sigma(\mathrm{PV}(X) - \mathrm{PV}(\mu))^2 \times p(X)$)) of \$7,282.

The standard deviation of the outcome distribution is a reasonable measure of risk. However, in order to control for the magnitude of different projects, we use the coefficient of variation:

$$\sigma/\mu$$

The relative difference between the two projects can serve as a proxy for a risk premium from which a relative risk-based discount rate can be derived. To determine relative risk, however, we cannot use risk adjusted rates for the discounting (Brealey and Myers 1996, p. 255) because this confounds the measuring of risk with the adjustment for it. Instead, σ and μ are to be computed based on Present Values that have been determined using the risk free rate of return. If the discounting in Examples 7a and 7b is at the risk free rate of 10%, we obtain:

$$\mathrm{cv}_{\text{inspection}} = 9{,}696/16{,}529 = 0.59$$

$$\mathrm{cv}_{\text{test plans}} = 7{,}282/16{,}529 = 0.44$$

and compute a relative measure of risk for both improvements:

$$\mathrm{Risk}_{\text{inspection}} = 0.59/0.44 = 1.34$$

$$\mathrm{Risk}_{\text{test plans}} = 0.44/0.59 = 0.74$$

which implies that the test plan improvement project is 74% as risky as is the code inspection project, or conversely, the code inspection effort is 34% more risky than the test plan improvement project.

If the inspection improvement is treated as the reference project, the minimum acceptable return for this project is 20% and the risk free rate r_f is 10%, then the reference risk premium $\phi_{\text{inspection}}$ is 10%. The risk premium for the test plan improvement project can be obtained by using the reference cost of capital and adjusting it for the appropriate relative risk measures:

$$\begin{aligned} k_{\text{inspections}} &= 0.20 \\ \phi_{\text{inspections}} &= 0.20 - r_f \\ &= 0.10 \\ \phi_{\text{test plans}} &= \phi_{\text{inspections}} \times \text{Risk}_{\text{test plans}} \\ &= 0.10 \times 0.74 \\ &= 0.074 \\ k_{\text{test plans}} &= r_f + \phi_{\text{test plans}} \\ &= 0.10 + 0.074 \\ &= 0.174 \end{aligned}$$

Now, $k_{\text{test plans}}$ needs to be applied to the expected value of the returns (\$20,000) for the test plans project

$$\begin{aligned} \text{PV}_{\text{test plans}} &= \$20{,}000/(1.174)^2 \\ &= \$14{,}493 \end{aligned}$$

From this result, additional capital budgeting measures, such as Net Present Value can be computed.

Other potential software process improvement projects could be evaluated using this method to factor in: risk and time. Thus, the NPV of project i, is computed as:

$$\text{NPV}_i = \sum_{t=1}^{n} \text{returns}_t/(1 + k_i)^t - \text{I},$$

where

$$k_i = r_f + \left(k_{\text{reference}} - r_f\right) \times \left[(\sigma_i/\mu_I)/(\sigma_{\text{baseline}}/\mu_{\text{baseline}})\right].$$

This approach is clearly oriented towards environments in which the cost of capital is probably not being used as an external standard. Its main goal is to help provide guidance on resource allocation based on timing and risk of returns within the context of the organizational mission, and relative to other candidate projects.

4.1.1. Valuing a metrics repository capability. It should be clear that the capabilities needed to financially analyze software process improvement efforts are highly dependent on data. More and better data leads to less variable projections of returns. Less variable projections of returns yields a smaller risk premium, which in turn results in a greater present value. It is clear that the decreased risk due to good historical data from a metrics repository can actually be used to quantify the value of a metrics initiative within an organization.

If improved data reduces the variability of our decisions, we can quantify its effects by considering the returns projected without the improved data as the baseline, and compute the relative NPV for the project given the improved data. Assuming the variability of the projection decreases, the ratio between the baseline coefficient of variation and the project i coefficient of variation will also decrease, yielding a lower Risk_i and consequently a lower discount rate:

$$\text{Risk}_i = (\sigma_i/\mu_i)/(\sigma_{\text{baseline}}/\mu_{\text{baseline}})$$

$$\text{NPV}_i = \sum_{t=1}^{n} \text{cash flow}_t/\left[1 + r_j + \text{Risk}_i \times \left(k_{\text{reference}} - r_f\right)\right]^t - \text{I}$$

The difference between the baseline NPV (zero) and the NPV of the projection with the improved data is essentially the value of the data improvements.

This is a well accepted approach to valuing information in many other fields and can be productively applied within the software engineering community.

5. Summary

When introducing a new software process improvement activity, it is often important that a business case be made for the activity in order to justify expending resources. For any process improvement activity benefits and costs (positive and negative returns) are accrued. Each return has a certain magnitude, timing and risk (or uncertainty) associated with it. There are several different approaches that can be used to value software process improvement activities. This report summarizes a variety of approaches from the domain of capital budgeting and finance that are available for evaluating the returns associated with process improvement. It identifies a number of key financial measures that can be used to judge the relative merits of various process improvement activities on a financial basis. This report also discusses key assumptions and issues associated with applying each technique.

Overall, we recommend that present value methods that incorporate both risk and the discounting of cash flows be used to evaluate software process improvement efforts as these methods address the key factors associated with the timing of returns, magnitude of returns and risk of returns.

We view the proposed methods and techniques discussed in this report as being useful in a forward looking, planning context to help choose which process improvement activities to conduct for each project. In order to create more accurate estimates of the financial performance of various process improvement activities, additional metrics pertaining to development effort and rework costs should be collected.

References

Brigham, E., and Gapenski, L. 1997. *Financial Management: Theory and Practice*, 8th ed., Dryden Press, Chapter 9.

Brealey, R., and Myers, S. 1996. *Principles of Corporate Finance*, 5th ed., McGraw-Hill, Chapters 5 and 12.

Curtis, W. 1995. Building a cost-benefit case for software process improvement. Notes from tutorial given at the *Seventh Software Eng. Process Group Conf.*, Boston, MA, May.

Dion, R. 1993. Process improvement and the corporate balance sheet, *IEEE Software* July, pp. 28–35.

Eickelmann, N. 1999. Development of return on investment (ROI) model and metrics for software system independent verification and validation. Center Software Initiative Proposal (CSIP) for the NASA Software IV & V Facility, January 12.

Harrison, W., Raffo, D., and Settle, J. 1999a. Process improvement as a capital investment: risks and deferred paybacks. *Proc. Pacific Northwest Software Quality Conf.*, Portland, Oregon, October.

Harrison, W., Raffo, D., and Settle, J. 1999b. Measuring the value from improved predictions of software process improvement outcomes using risk-based discount rates. *First Workshop on Economics-Driven Software Eng. Res.*, Los Angeles, CA, May 17.

Hayes, W., and Zubrow, D. 1995. Moving on up: data and experience doing CMM-based software process improvement. *Seventh Software Eng. Process Group Conf.*, Boston, MA, May 23.

Herbsleb, J., Zubrow, D., Siegel, J., Rozum, J., and Carleton, A. 1994. Software process improvement: state of the payoff, *Am Programmer* 7(9):2–12.

Jones, C. 1996. The pragmatics of software process improvements, *Software Process Newslett. Software Eng. Technical Council Newslett.*, No. 5, Winter, pp. 1–4.

Lipke, W., and Butler, K. 1992. Software process improvement: a success story. *CrossTalk*, No. 38, November, pp. 29–31, 39.

McGibbon, T. 1996. A business case for software process improvement. Data Analysis Center for Software State-of-the-Art Report, prepared for Rome Laboratory, September 30. WWW URL: *http:// www.dacs.dtic.mil / techs / roi.soar / soar.html.*

O'Neill, D. 1989. *Software Inspections Course and Lab*, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.

O'Neill, D. 1995. National software quality experiment: results 1992–1995. *Proc. Seventh Annual Software Technol. Conf.*, Salt Lake City, UT, April 9–14, 1995, Hill Air Force Base, UT, Software Technology Support Center.

O'Neill, D. 1996. National software quality experiment: results 1992–1996. *Proc. Eighth Annual Software Technol. Conf.*, Salt Lake City, UT, April 21–26, 1996, Hill Air Force Base, UT, Software Technology Support Center.

Quirk, J.P., and Terasawa, K.L. 1987. The choice of discount rate applicable to government resource use: theory and limitations. Rand Publication Series R-3464-PA & E, December.

Raffo, D.M. 1996. Modeling software processes quantitatively and assessing the impact of potential process changes on process performance. Ph.D. Dissertation, Graduate School of Industrial Administration, Carnegie Mellon University, Published by UMI Dissertation Services, Ann Arbor, MI.

Raffo, D.M., and Kellner, M.I. 1999. Predicting the impact of potential process changes: a quantitative approach to process modeling. *Elements of Software Process Assessment and Improvement*, IEEE Computer Society Press.

Raffo, D., Settle, J., and Harrison, W. 1999. Estimating the financial benefit and risk associated with process changes. *First Workshop on Economics-Driven Software Engineering Res.*, Los Angeles, CA, May 17.

Slaughter, S., Harter, D., and Krishnan, M. 1998. Evaluating the cost of software quality, *Commun. ACM* August, pp. 67–73.

Vienneau, R. 1995. The present value of software maintenance, *J. Parametrics* April, pp. 18–36.

Warren Harrison is Professor of Computer Science at Portland State University and Clinical Associate Professor of Medical Informatics and Outcomes Research at Oregon Health Sciences University. He received a B.S. in Accounting from the University of Nevada, an M.S. in Computer Science from the University of Missouri-Rolla and a Ph.D. in Computer Science from Oregon State University. He has held appointments at Bell Telephone Laboratories, Lawrence Livermore National Laboratories, and the University of Portland where he served as an Assistant Professor of Business Administration. His research interests include software measurement, decision-making under uncertainty and software quality assurance.

David Raffo is Assistant Professor of Business Administration at Portland State University. He received the B.S.E. in Industrial Engineering from the University of Michigan, the M.M.E. in Manufacturing Engineering, the M.S. in Industrial Administration and the Ph.D. in Manufacturing and Operation Systems from Carnegie Mellon University. He has held positions with the IBM Corporation and Arthur D. Little. His research interests include software process modeling and simulation.

John Settle is currently Associate Professor of Finance at Portland State University. He received a B.A. in Mathematics from Pomona College, an M.S. in Mathematics, an MBA, and a Ph.D. in Finance at the University of Washington. He has held appointments at Santa Clara University, Washington State University, and Portland State University. His research interests include required rates of return, observed market returns and the application of financial concepts to software engineering.

Nancy Eickelmann is Research Assistant Professor at the West Virginia University NASA IV & V Facility. She received a B.S. in Finance and an M.B.A. in Information and Decision Systems from San Diego State University, an M.S and a Ph.D. in Computer Science from the University of California-Irvine. She was nominated a Hughes Doctoral Fellow and served on the research staff at Hughes Research Laboratories for four years. She also served as a senior member of the technical staff of the Advanced Technology Research Programs at Microelectronics and Computer Technology Corporation in Austin, Texas. Her research interests include measurement and evaluation of software test processes, product-line development paradigms and architecture-based development and verification technologies.