Proceedings 4th International Conference on Software Reusability, April 1996

A Comparison of Approaches to Reuse Investment Analysis

John Favaro
Intecs Sistemi S.p.A.
Via Gereschi, 32-34
56127 Pisa - Italy
favaro@pisa.intecs.it

Abstract

Reuse economics has been the subject of vigorous study over the past several years. Although significant progress has been made in the areas of reuse metrics and cost estimation, much work to date in reuse investment analysis has not always reflected accepted mainstream financial analysis practices. This paper compares several approaches that have been described in the reuse literature, points out known problems and indicates remedies.

Keywords: Net present value, cash flow, discount rate, internal rate of return, payback, profitability index, book value, amortization.

1. Introduction

The field of software reuse economics tries to bridge the gap between the technical and the financial aspects of reuse-oriented software development. That bridge starts from the perspective of the software engineer, and ends with the perspective of the corporate financial manager. Software engineers are unfamiliar with the latter perspective, but it is essential if software reuse is to be successfully institutionalized. As Pfleeger [13] has stated,

> It is important for software engineers to be able to translate the language of reuse into the language of accounting, so that reuse investment can be compared with other possible corporate investment alternatives.

This article considers several representative approaches to reuse investment analysis that have been described in the reuse literature, and reviews their known strengths and weaknesses from the perspective of corporate financial analysis, in an attempt to contribute to the software engineer's understanding of this rather different perspective.

2. The role of investment analysis in software reuse economics

Software reuse economics broadly encompasses three kinds of activities:

- *Reuse metrics*—the measurement of reuse-related characteristics of software;
- *Cost estimation*—the estimation of costs and benefits associated with reusable software development (often supported by reuse metrics);
- *Reuse investment analysis*—the evaluation of investment decisions.

Some examples of the results of each activity appear in the table.

Activity	Examples
Reuse metrics	• Percentage of reused code • Number of reuses of a component
Cost estimation	• Cost to make a component reusable • Savings in avoided work
Investment analysis	• Return on investment for a reusable component • Comparison or ranking of alternative investments

Much progress has been made in reuse metrics, and extensive work in cost estimation has also been

carried out in many places, including CSR and NEC [12], IBM [14], SPC [16], and NATO [11]. Two recent surveys are [5] and [15]. Reuse metrics and cost estimation originate in the domain of software engineering, a fact reflected in the large body of solid work available. But the relative lack of comparably solid work in reuse investment analysis (it is often treated as though it were equivalent to cost estimation) reflects a need for software engineers to become familiar with a different domain.

2.1 The investment analysis context

Reuse investment analysis is (or should be) in the domain of the corporate financial analyst, whose concerns are not directly addressed by the software engineering perspective. The first difference to keep foremost in mind about the perspective of corporate investment analysis—as opposed to metrics and cost estimation—is that reuse is *only one alternative* for the company. In a corporate context, investment analysis is concerned only with the best way to allocate capital and human resources.

Given this context, there is in fact *always* an alternative to investing in a program of reuse: an equivalent investment in the capital markets that provides some expected yearly rate of return. This is the fundamental yardstick against which any reuse-oriented investment can and must be comparable.

Since capital investments are analyzed with respect to periods of time (e.g. a "seven-year savings bond"), reuse projects must be analyzed with the same approach in order that comparisons be possible. An investment analysis method's treatment of the effects of time is thus an essential factor.

2.2 Cost estimation and cash flow analysis

Cost estimation in reuse economics corresponds to the task of *cash flow analysis* in corporate finance. A candidate reuse project has potential cash flows which could be positive (such as savings from avoided work) or negative (such as work to generalize a component). In [18] these are characterized as "benefits" and "costs." Although progress is rapidly being made in techniques for reuse-oriented cost estimation, it is and will remain a very challenging and difficult task—as indeed it is in all of corporate finance. Techniques for quantifying economic benefits that are of particular interest in reuse, such as decreased time-to-market, are emerging only now [8]. But agreement is being reached among many that working time (e.g. "engineering hours" [7]) expressed in dollar amounts is an especially useful and realistic way of capturing economic benefits and costs associated with a reuse program. (Considerable work has also been done in the measuring of non-economic benefits, but these are out of the scope of this paper.)

Cash flows are forecast over a suitable time horizon which could be anywhere from one year to infinity, depending on the particular circumstances. It is here that special concerns in the software reuse field—such as the rapid obsolescence of new technology—should be taken into consideration. For example, in a rapidly changing domain, the time horizon might reasonably be limited to three or four years. It is important that the time horizon be "neutral" in the sense that it reflects only the estimated meaningful life of the project's cash flows, *not* desired characteristics such as "early returns."

The output of cost estimation typically is represented in tabular form as in the following:

C_0	-5000
C_1	+2000
C_2	+3000
C_3	+6000

This example shows a negative cash flow C_0 (the "initial investment") of -5000 dollars followed by positive forecasts for three more years.

2.3. The line between cost estimation and investment analysis

In corporate finance, the estimation of costs/benefits, in the form of cash flows, is considered to be an activity prior to investment analysis. That is, it is essential that investment analysis should be based *only* on the cash flows yielded by cost estimation as shown above. This clear demarcation allows a critical requirement to be fulfilled: the ability to compare arbitrarily different kinds of alternatives (e.g. software reuse versus sheep farming). In the reuse literature, this demarcation is not always clearly visible. Thomas [19] relates this phenomenon to the Goal-Question-Metric (GQM) paradigm [20], which observes that measurement without the guidance of clear, well-understood goals tends to be ill-focused. (We return to this problem in Section 5 in the context of *value-based management*.) Until software engineers become familiar with the purpose and needs of reuse investment analysis, the proper focus of measurement and cost estimation activities will remain elusive.

3. Comparison of approaches

With the line clearly drawn between cost estimation ("cash flow analysis") and investment analysis, we are now in a position to compare investment analysis approaches. There is much agreement in the world of corporate financial analysis on the characteristics that an investment analysis method should exhibit in order to be useful and reliable:

- it should depend as much as possible only on forecast cash flows, and not on subjective and arbitrary factors such as accounting practices or managers' instincts;
- it should have a quantifiable *acceptance rule* (or *criterion*) to guide the investment decision;
- it should be suitable for comparing and ranking candidate projects, both singly and in combinations;
- it should be able to deal with arbitrarily large or small projects;
- it should be able to handle projects of arbitrarily long or short duration.

In the remaining sections, several different approaches that have been described in the literature will be examined in the light of this list of desirable characteristics.

3.1 Net present value (NPV)

The net present value (NPV) approach has been mentioned in several sources in the literature, but has been used most extensively in work at Hewlett-Packard. In [7], the application of the NPV approach to evaluate two multi-year corporate reuse projects is described. At the component level, the use of the approach is illustrated to rank components, either in terms of priority or as alternatives. In [8], techniques are also described to account for time-to-market and risk in a reuse context. That work is firmly grounded in accepted principles of corporate finance and forms a point of departure for the following discussion.

Most people are comfortable in dealing with money amounts from the past in terms of today's dollars. ("That house cost 15000 dollars in 1960, corresponding to 130000 dollars today.") Since investment planning deals with future rather than past money amounts, the same exercise is possible in reverse. ("In ten years this will be only worth half as much as today.") This leads directly to the concept of the *present value* of an investment—the value today of a predicted future cash flow. The *discounted cash flow* formula for present value is defined as

$$PV = C_t / (1 + k_t)^t$$

where t ranges over all future time periods under consideration, and

C_t = future cash flow in period t
k_t = discount rate in period t

The *net* present value is calculated by subtracting the original investment from the present value (or equivalently, adding the initial investment as a negative value):

$$NPV = C_0 + PV$$

The acceptance rule is simple: invest in a project if its net present value is greater than zero.

The discount rate is also known as the *opportunity cost* of the project, because it corresponds to the rate of return expected from an equivalent investment on the capital markets—thus representing the cost of taking the "opportunity" to invest in the project. (As mentioned earlier, this is *always* an alternative.) The opportunity cost is a figure to be estimated by project planners (although it is often done at the corporate level). Furthermore, the discount rate could be different in different periods (e.g. short-term versus long-term rates)—although normally a single rate is assumed over the life of the project.

As an application of the NPV approach, consider the following scenario. A company analyst wishes to evaluate two alternative ways to create a product line:

- Base it on procured Commercial-Off-The-Shelf (COTS) software. There is a large initial procurement cost, with subsequent high returns (due to avoided work), but the COTS software will be outdated and must be replaced with another purchase after three years;
- Have an in-house program of reuse in order to create and maintain the product line, with considerable up-front costs, but much higher returns when the program is up and running.

Here the cash flow analysis will include cost-of-product issues (e.g. the COTS solution may require payment of royalty for every product sold). Suppose

that the forecast cash flows over four years from his own and his software engineers' cost estimation activities yields the following:

Cash Flows	COTS	Reuse
C_0	-9000	-4000
C_1	5000	-2000
C_2	6000	2000
C_3	7000	4500
C_4	-4000	6000
all C_i	5000	6500
NPV at 15%	2200	2162

Here the COTS-based scenario has a slightly higher NPV and therefore is preferred. It is worthwhile noting a few points about even this hypothetical case:

- the COTS approach and the reuse approach are quite different technically, yet they can be compared directly in this way, based solely on their predicted cash flows;
- the two cash flow patterns are very different, illustrating that any arbitrary pattern is possible;
- the NPV approach is sensitive to the timing of cash flows. Note, for example, that the sum of the cash flows is higher for the reuse-based scenario—yet the overall NPV shows the penalizing effect of time on the value of later returns [8]. ("Time is money.")

The fact that all NPV values are expressed in the same units of today's dollars also implies that they are sensitive to the size or *scale* of a project: large or small returns are directly reflected in the NPV quantity.

Furthermore, NPV possesses the important quality of being additive. That is, for project P and project Q,

$$NPV_{P \text{ and } Q} = NPV_P + NPV_Q$$

which means that *combinations* of projects can be evaluated. For example, consider a proposed project to construct a reusable inference engine (IE) module equipped with a modern graphical user interface (GUI). An analysis may reveal an NPV of +3200 dollars, a very encouraging prospect. But a finer, separate analysis of the inference engine and the GUI subcomponents may reveal that NPV_{IE} equals +5000 dollars, but NPV_{GUI} equals -1800 dollars. That is, the negative value of the GUI subcomponent is "camouflaged" by the high positive value of the inference engine subcomponent. In this case, it is preferable to develop only the inference engine subcomponent.

In NPV analysis, risk is accounted for both in the cost of capital and cash flow forecasts. In practice, NPV analysis is generally not carried out as a *point* estimation (yielding a single number) but as a *mean* estimation, accompanied by various forms of risk analysis (e.g. sensitivity analysis, scenario analysis, and decision analysis).

In summary, the net present value approach exhibits the following important characteristics for reuse investment analysis:

- It incorporates the effects of time on the value of a project, permitting realistic comparison with alternative capital investment possibilities;
- It does not depend on arbitrary factors such as a company's bookkeeping practices, managers' instincts, or the current business climate—only on forecast cash flows and the opportunity cost of capital for a proposed project.
- Since values are measure in units of today's dollars, they are additive, can be ranked, and are sensitive to scale. Alternatives large and small, reuse-related or non-reuse related, can be compared and combined.

In the following, we examine the behavior of other proposed methods with respect to these characteristics.

3.2 Payback

Payback—the time or number of uses required to recover the cost of an investment—has appeared more often in the reuse literature than any other approach. In work at the Software Productivity Consortium [1] [16] the *payoff threshold value* for a component is defined as

$$N_0 = E / (1-b)$$

where

- E = relative cost of developing a component for reuse
- b = relative cost of integrating the component
- N_0 = number of times a component must be used before its cost is recovered

For example, cost estimation was carried out in [4] for a set of reusable components developed for the user interface in a shipboard data handling project. The component set included both domain-specific "vertical" components (e.g. a forms handler) and domain-independent "horizontal" components (e.g. a directed-graph manager). The payback for one large "vertical" component was calculated to be $N_0 = 13$ uses, whereas for a small "horizontal" component the payback was only $N_0 = 2$ uses. Generally, the acceptance rule applied to payback is that projects must recover their costs within a certain *cutoff* date (e.g. "within the first year") in order to be accepted.

The idea of "payback," "break-even," or "cost recovery" is intuitively very appealing; but for investment analysis, this approach presents several problems. First, the choice of cutoff date is generally arbitrary and subjective. Secondly, payback is not sensitive to patterns of cash flows—in particular, it ignores all cash flows after the cutoff date. And when cash flows are not discounted (usually the case for payback) equal weight is given to *all* cash flows. Consider a scenario in which there is a choice between (a) hiring a programmer who produces benefits at an even rate; (b) making an equal investment in some components with slightly delayed returns; (c) making a large investment in a reuse program with larger delayed returns.

Cash Flows	Choice (a)	Choice (b)	Choice (c)
C_0	-2000	-2000	-4000
C_1	1000	500	-1500
C_2	1000	1500	5500
C_3	1000	1000	6000
Payback	2 years	2 years	2 years
NPV at 15%	283	226	2799

As the table indicates, all of the choices produce a payback within two years. But choice (a) produces earlier returns than (b) and thus has a slight advantage; and choice (c) produces much higher returns in the last period, resulting in a far greater value. This is an example of the problem of *scale*: the true value of the project is not taken into account by the payback approach. The problem of scale is also exhibited in the example cited from [4]: the vertical component with a payoff after 13 uses was on a far larger scale than the simple horizontal component with its payoff after only 2 uses. Is the small investment with rapid payback for the horizontal component really better than the large investment with delayed payback for the vertical component? For purposes of the investment decision, those two payback numbers are effectively meaningless—only a discounted cash flow analysis would allow a direct, meaningful comparison.

Finally, the example also shows that the choice of cutoff period affects whether short-lived or long-lived projects are accepted—an arbitrary and error-prone approach, with a tendency to penalize forward-looking reuse programs.

In summary, payoff is a useful way to *communicate* an idea about the worthiness of a project. It is an intuitively appealing notion, and is easy to grasp by managers not involved in the details of cost estimation. But in fact it is an *ad hoc* approach, which is useful for communicating the *results* of investment analysis (as done in [7]), but not for the analysis itself.

3.3 Average return on book value

The reuse literature has long encouraged a view of software as a *capital asset* [17], to be treated like tangible assets such as machinery. The debate on whether software development should be considered as *capital investment* in real assets or salary-related *operating expenses* is likely to continue for some time.

This issue relates to the concept of *amortization* as proposed by CSR and NEC [12]. In this approach, an amortization schedule for investment for reusable work products is agreed upon, and deducted as appropriate from future cash flows from those work products.

This approach corresponds to the *book rate of return* technique in corporate accounting. The book rate of return of an investment is calculated by dividing the average profits from predicted future cash flows (minus amortization costs) by the average net book value of the investment. The acceptance rule is that a project is accepted if its book rate of return meets some target set by the analyst, such as the company's current book rate of return or that of the industry as a whole—for example, in [16], a "minimum acceptable return" of 20% is described for a hypothetical project under consideration.

As an illustration, consider now a $20000 investment in a four-year reusable workproduct development program, for which a straightforward constant amortization schedule ($5000 per year) is agreed:

Year	Gross Book Value of Investment	Cumulative amortization	Net book value of investment
Year 0	20000	0	20000
Year 1	20000	5000	15000
Year 2	20000	10000	10000
Year 3	20000	15000	5000
Year 4	20000	20000	0

Thus, the average net book value of the investment in work products is 50000/5 = 10000 dollars.

Now consider some possible scenarios for cash flows on this investment, including "early," "middle," and "late" returns. The numbers in parentheses represent the net cash flows after amortization:

Year	Early	Middle	Late
Year 1	13000 (8000)	10000 (5000)	7000 (2000)
Year 2	11000 (6000)	10000 (5000)	9000 (4000)
Year 3	9000 (4000)	10000 (5000)	11000 (6000)
Year 4	7000 (2000)	10000 (5000)	13000 (8000)

All three scenarios produce an average net income of 5000 dollars. Therefore, the average book rate of return of all three scenarios is 5000/10000 = 50 percent. Yet clearly the "early returns" scenario is preferable. This approach is entirely insensitive to the variation in cash flow patterns.

Even worse, the calculation is dependent on the choice of amortization schedule, which might be linear, accelerated, or some other choice made by the accountant—and is also constrained by tax laws. Furthermore, amortization is only applied to capital investment, not operating expenses. A consistent policy must be developed for deciding which investment in reusable workproducts is to be considered capital investment (or *capitalized* by the accountant) and which is to be considered as operating expenses (or *expensed*). Recalling the introductory remarks of this section, this is likely to be a difficult and subjective task—and in any case is likely to be the task of the accountant. (In this regard, it is also worth noting that there is a strong traditional tendency by accountants to expense rather than capitalize, due to tax considerations [9].)

Finally, the choice of target book rate of return is arbitrary and subjective—how is it decided that 20% is a "minimum acceptable" rate of return?

Or the choice may depend on the company's current rate of return. But only the merits of the project under evaluation should be taken into consideration—a company that is in financial trouble, with a low current book rate of return, may get even more deeply into trouble by setting its target threshold too low.

In summary, this approach is vulnerable to accounting distortions and too many other subjective factors to be satisfactory.

3.4 Internal rate of return (IRR)

Work carried out at IBM [14] introduces the *internal rate of return* (IRR) approach as the most common way of expressing corporate return on investment. It goes on to note that IRR is related to NPV, and illustrates both figures calculated side-by-side in a cash flow scenario.

IRR is a way of defining the rate of return of a long-lived asset. IRR and NPV are related as follows: *IRR is the discount rate which makes NPV equal to zero*. That is,

$$C_0 = C_t / (1+IRR)^t$$

for all time periods t under consideration. The acceptance rule says to accept a project if its IRR is greater than the opportunity cost of the project. Since they are both based on discounted cash flows, IRR and NPV can in fact both be used to produce equivalent results. However, proper use of IRR is more difficult, and its calculation is subject to several disturbing anomalies.

While the opportunity cost is *estimated* by project planners, the IRR is *calculated* from forecast cash flows—that is, it is a *derived* figure, and can be undermined solely by certain patterns of cash flows in a project. For example, in [14], after the initial investment, the cash flows in the illustrated project were all positive. This need not be the case, of course. Alternating cycles of component development, reuse, maintenance, etc., could produce alternating positive and negative cash flows, as in the following:

C_0	+1000
C_1	-3600
C_2	+4320
C_3	-1728
IRR in percent	+20
NPV at 10 percent	-0.75

This scenario has an IRR *greater* than the cost of capital, but a *negative* NPV. Thus the IRR acceptance rule breaks down here and leads to an incorrect decision.

The problem of alternating cash flows leads to other technical anomalies in the calculation of IRR, as in:

C_0	-4000
C_1	+25000
C_2	-25000
IRR in percent	*both* 25 *and* 400
NPV at 10 percent	-1934

This cash flow scenario (from [2]) produces *two* IRR values (due to multiple sign changes). Furthermore, the following scenario yields *no IRR at all*:

C_0	+1000
C_1	-3000
C_2	+2500
IRR in percent	*none*
NPV at 10 percent	+339

An investment analysis method must be flexible enough to handle all cash flow scenarios. As seen above, the IRR approach is problematic in this respect.

IRR also exhibits problems with regard to the scale of projects. Consider again a scenario similar to that in [4]: A decision must be made whether to pursue a strategy of developing small low-level horizontal components or large vertical components. The forecast cash flows over two years are as follows:

	Horizontal Component	Vertical Component
C_0	-100	-10000
C_1	+200	+15000
IRR (percent)	100	50
NPV at 10%	82	3636

The IRR of the small horizontal component is double that of the large vertical component—but the NPV of the vertical component dwarfs that of the horizontal component, and thus it is clearly the preferred choice. IRR was unable to capture and express the differences in scale of the two projects. The IRR *could* be used properly in this case by constructing an awkward scheme of incremental investments—but NPV is much more straightforward.

Finally, it should be noted that since the IRR is a single calculated value, it cannot model multiple discount rates (for example, when short-term interest rates are different from long-term interest rates).

In summary, IRR is subject to too many anomalies and constraints to make it a preferred approach to reuse investment analysis.

3.5 Profitability Index (PI)

The idea of examining the ratio of "benefits" to "costs" is intuitively appealing, and appears in numerous variations throughout the reuse literature, such as [16], where "return on investment" (ROI) is defined as

ROI = Savings/Investment - 100

and [18], where the "quality of investment" (Q) is defined as the ratio of "reuse benefits" (B) to "reuse investments" (R), or Q = B/R.

This idea appears in another variation in the NATO reuse standards [11], where two measures are presented for comparing reusable software components, as a way of setting priorities for allocation of scarce resources. The first is the *cumulative discounted cash flow* (CDCF), defined as the sum of the annual discounted cash flows minus the original investment (called *accession cost* in the text) :

$$CDCF = C_i - C_0$$

where

C_i = discounted cash flow in year *i*

C_0 = original investment.

As defined there, CDCF is none other than net present value. (This points up the problem of conflicting nomenclature in the literature.) The NATO text then presents as an alternative comparison measure the *profitability index* , which it defines as $CDCF/C_0 = NPV/C_0$.

This definition is different from the one used in mainstream corporate finance, where the profitability index (also known as the *benefit-cost ratio*) is defined as the *present* value (rather than the *net* present value as in the NATO text) of future cash flows divided by the initial investment:

$$PI = PV/C_0$$

The acceptance rule for PI states that a project with a profitability index greater than one should be accepted.

But once again, problems arise for projects of different scales. Consider the same scenario as presented in the section on IRR:

	Horizontal Component	Vertical Component
C_0	-100	-10000
C_1	+200	+15000
PI	1.82	1.36
NPV at 10%	82	3636

Here, too, a misleading result is obtained: the profitability index for the horizontal component is higher than for the vertical component, in spite of a much lower net present value. As for the case of IRR, the correct result can be obtained with PI by the use of incremental cash flows. But it is an awkward and error-prone exercise, and ultimately unnecessary. In summary, the profitability index is misleading on problems of scale, is not additive, and exhibits no advantages over net present value. As for payback, the profitability index or similar expressions for "return on investment" are primarily useful for communicating the *results* of investment analysis (e.g. to upper management) in a less rigorous and more intuitively immediate manner.

4. Summary and conclusions

A summary of reviewed approaches and associated issues is shown in the following table.

Approach/Rule	Issues
Net Present Value (NPV) *Rule*: Accept if NPV > 0	• Most acceptable, realistic approach • Values are additive, allowing project combinations to be evaluated • Takes differences of scale into consideration
Payback *Rule*: Accept if payback within some specified target time period	• Usually does not discount cash flows • Arbitrary cutoff dates for payback • Ignores flows after cutoff date • Not sensitive to scale
Average Return on Book Value *Rule*: Accept if predicted rate of return greater than some target	• Insensitive to cash flow patterns • Dependent on accounting practices • Arbitrary targets
Internal Rate of Return (IRR) *Rule*: Accept if IRR greater than the opportunity cost of project	• Uses discounted cash flows • No direct economic significance • Subject to mathematical anomalies • Not sensitive to scale
Profitability Index (PI) *Rule*: Accept if PI > 1	• Conceptually closest to NPV • Values not additive • Not sensitive to scale

In the field of software reuse economics, much progress has been reported in metrics and cost estimation, whose techniques arise directly from the software engineering domain. Less progress is evident in investment analysis, partly because mainstream business practices have not yet sufficiently been taken into consideration by software engineers unfamiliar with them. Intuitively appealing notions such as "break-even point," "payback," and "return on investment" are often used without consideration for known shortcomings.

The issues discussed in this article have all been dealt with in mainstream corporate finance [2], whose techniques are directly applicable to reuse investment analysis. Related work on software investments can also be found in textbooks such as [21], [22].

To return to the statement by Pfleeger in the beginning of this text, it is desirable for software engineers to become conversant with these issues so that well-grounded, persuasive business cases can be built for reuse programs.

5. Future directions: the broader context of value-based reuse management

The broader context of this discussion is embodied in the principles of "value-based management" [10], which seeks to make the governing objective of a company *clear* and *consequential*. Specifically, it defines the maximization of *economic value* as the most desirable governing objective, and one of its fundamental tools is the discounted cash flow analysis method of the NPV approach.

The principles of value-based management are particularly relevant to management for reuse, because they can provide the missing focus (as noted in Section 2) on the true value-creating factors, rather than more subjective criteria such as market share or even quality. They also yield some surprising alternative insights on previous ideas. For example, the problems of comparing and ranking alternatives discussed in this article are based on the concept of "scarce resources," also known as *capital rationing*. This reflects the implicit assumption also seen in the reuse literature that choices are *forced* by limited capital budgets. ("Capital is limited but free of cost.") But value-based management challenges the idea that capital need be considered to be scarce—rather, it can be obtained, at a price, on the capital markets. ("Capital is unlimited but expensive.") Such alternative perspectives hint at ways to implement innovative proposals such as the *reuse bank* suggested in [12].

Finally, value-based management yields useful insights about how to organize for reuse. For example, in the reuse literature it is often suggested to spread out costs of component development across business units, e.g. in a centralized group. But value-based management encourages the full allocation of costs to business units to yield maximum clarity in the associated cash flows. Separate departments for component production are thus regarded critically. The effects of this on reuse organization have yet to be explored fully.

Full value-based reuse management could offer a way to link the chain of all activities in software reuse economics, rendering them clear, focused, and consequential. But first, the weakest link must be strengthened. The essential activity of reuse investment analysis is the proper valuation of a project as an investment of expensive corporate resources—both capital and talent. More case studies need to appear in the literature [7] in which different projects are compared and ranked according to a systematic and coherent approach that is accepted in the mainstream corporate finance world.

Acknowledgements

This paper benefited from work at the European Software Institute [3], and from discussions with W. Lim and I. Thomas. Special thanks for careful review of the material on corporate financial analysis are due to Kenneth Richard and Paul Frederick of Marakon Associates (London)—who also point out the complementary role that software engineers have to play in educating financial analysts and managers in the proper application of their financial techniques to software reuse investments.

6. References

[1] B. Barnes *et. al.*, "A Framework and Economic Foundation for Software Reuse," *Proc. Workshop on Software Reusability and Maintainability*, October 1987.

[2] R. Brealey and S. Myers, *Principles of Corporate Finance*, McGraw-Hill, New York, 1981.

[3] European Software Institute, "Technical Report on Reuse," Bilbao, May 1995.

[4] J. Favaro, "What Price Reusability?," *Proc. First Symp. on Environments and Tools for Ada*, New York, 1990, pp. 115-124.

[5] W. Frakes and C. Terry, "Software Reuse and Reusability Metrics and Models," *ACM Computing Surveys*, 1995 (to appear).

[6] J. E. Gaffney and T. A. Durek, "Software Reuse—Key to Enhanced Productivity; Some Quantitative Models," *Information and Software Technology*, 31:5, June 1989.

[7] W. Lim, "Effects of Reuse on Quality, Productivity, and Economics," *IEEE Software*, Sept. 1994, pp. 23-30.

[8] R. Malan and K. Wentzel, "Economics of Software Reuse Revisited," *Proc. 3rd Irvine Software Symposium*, Irvine, 1993.

[9] Marakon Associates, *Private Communication*, 1995.

[10] J. M. McTaggart, P.W. Kontes, M.C. Mankins, *The Value Imperative*, The Free Press, New York, 1994.

[11] NATO, "Standard for Management of a Reusable Software Component Library," NATO Communications and Information Systems Agency, August 1991.
[12] S. L. Pfleeger and T. B. Bollinger, "The economics of reuse: new approaches to modelling and assessing cost," *Information and Software Technology*, Vol. 36 (8), 1994, pp. 475-484.
[13] S. L. Pfleeger, "Measuring Reuse: A Cautionary Tale," *submitted for publication*, 1995.
[14] J. S. Poulin and J.M. Caruso, "A Reuse Metrics and Return on Investment Model," *Proc. 2nd. Int. Workshop on Software Reusability*, IEEE Computer Society Press (May 1993).
[15] J. S. Poulin, "Measuring Software Reusability," *Proc. 3rd. Int. Conference on Software Reusability*, IEEE Computer Society Press (November 1994).
[16] SPC, "Reuse Adoption Guidebook," Report SPC-92051-CMC, Version 01.00.03 (Nov. 1992).
[17] P. Wegner, "Capital-Intensive Software Technology," *IEEE Software* 1(3), July 1984.
[18] B. Barnes and T. Bollinger, "Making Software Reuse Cost Effective," *IEEE Software* (1 1991), pp. 13-24.
[19] I. Thomas, *Private Communication*, 1995.
[20] V. Basili and D. Weiss, "A methodology for collecting valid software engineering data," *IEEE Transactions on Software Engineering*, SE-10 (3), pp. 728-738, Nov. 1984.
[21] B. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.
[22] L. Putnam and W. Myers, *Measures for Excellence: reliable software on time, within budget*, Prentice Hall, 1992.