# The Economic Impact of Learning and Flexibility on Process Decisions

**Hakan Erdogmus,** *National Research Council Canada*

Learning and flexibility have much to do with whether a development process makes economic sense.

The perception that software projects have unusually high failure rates has fueled the debate on software process. In the 1990s, the Standish Group estimated that the total economic toll of cancelled or overrun software projects could reach several tens of billions of dollars in terms of wasted effort and opportunity costs.[1] Worse yet, even when delivered on time and on budget, software rarely has any significant salvage value when it fails to meet user needs.

Arguments abound regarding the sources of this apparent crisis. Subjective preferences for one development process over another often underlie the reasons given for failure. However, the case for a particular approach to software development must rest on more than just process dogma. Plain old economics is a good place to start.

This article argues for an economic basis for rationalizing process selection decisions. It demonstrates how, under conditions of uncertainty, learning (an undertaking's capacity to resolve uncertainty) and flexibility (the ability to exploit the learning outcomes) affect such decisions. Uncertainty is a critical driver in process selection because it's ubiquitous in software development and it determines the degree of flexibility and learning needed to maximize economic value.

## A process flexibility continuum

To illustrate the impact of flexibility and learning, I compare two stereotypes representing the opposite halves of the process flexibility continuum shown in figure 1. This continuum is similar to Barry Boehm's spectrum,[2] but it focuses on process granularity rather than planning effort. The flexibility continuum represents, from left to right, an increase in the frequency of incremental delivery points with a simultaneous decrease in increment size. Methods that rely on a single or few iterations (Winston Royce's waterfall model[3] and some military standards[4]) are located toward the extreme left. Methods that rely on frequent, small iterations (agile methods such as Extreme Programming and Scrum[5]) are located toward the right. Boehm's spiral model[6] would be somewhere right of the midpoint. Although process frameworks such as the Rational Unified Process and the Capability Ma-


0740-7459/05/$20.00 © 2005 IEEE

turity Model[4] are commonly associated with the rigid side, they can actually cover a wide range but increasingly favor the left half as the level of ceremony and maturity grows.

The stereotypical *sequential* processes represent the rigid (left) half of the flexibility continuum. It involves a single, large commitment of resources in the beginning, sequential execution of distinct phases in the middle, delivery of a complete product in the end, and delayed benefits. The stereotypical *iterative* processes represent the flexible (right) half of the continuum. It's characterized by several small investments associated with multiple iterations of essentially the same process, an evolving product delivered incrementally through these iterations, and end-of-iteration decisions.

Using these stereotypes and simple models, I show that high-uncertainty environments economically favor processes near the flexible end. The value perspective taken here is that of the delivered system's customer, with the assumption that the development team's compensation is proportional to the benefits the customer reaps from the project.

## What really counts

Fundamental assumptions regarding the nature of software and the goal of software process are partly responsible for the apparent division between opposite process camps.

The view that software is essentially a *hard* artifact, and therefore that it can't easily be changed, dominates the traditional school of thought. If software is a hard artifact, it makes sense to engineer it using a carefully planned and tightly controlled process.

Another common assumption is that software process aims to achieve repeatability and predictability. Repeatability implies that similar inputs produce similar outputs: if requirements and constraints are well articulated, you should be able to build a system that satisfies them by rigorously executing the usual phases in a single pass. Predictability refers to the much sought-after property that inputs determine cost and schedule. Consequences of project overruns frequently drive the quest for repeatability and predictability. However, being on time and on budget, although a popular management motto, is neither the absolute nor the most important measure of success. It does not attack a paramount consideration head-on: value creation.

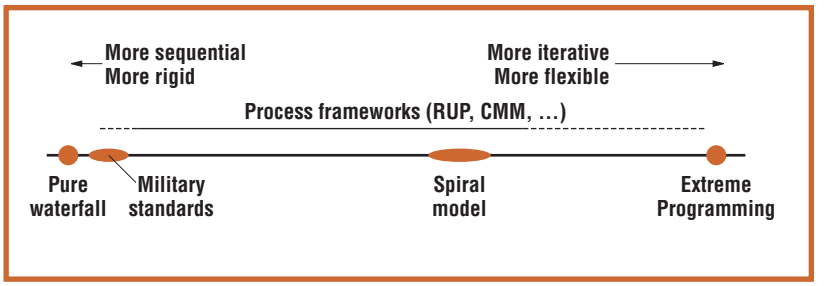


**Figure 1. The flexibility continuum.**

Others, including agile approach proponents,[5] think of software as a *soft* artifact, intrinsically amenable to change and subject to inevitable pressures to change it. Such characterizations of software aren't novel. In his influential essay, "No Silver Bullet," Frederick Brooks discusses four essential difficulties associated with developing software systems.[7] Among these, changeability and conformity in particular underlie the soft characterization. Software obeys no natural physical laws, which produces constant pressures to bend it to new, previously unanticipated needs. Ignoring these fundamental characteristics, Brooks argues, would be ignoring software's full potential.

Supporters of the soft characterization recognize that when software development's ultimate goal is to create economic value,[8,9] factors other than repeatability and predictability come into play. Value creation has several dimensions. Increasing software development's efficiency through quality and productivity improvements is the most obvious way that software process can contribute to value creation. Under conditions of uncertainty, flexibility and learning constitute another. I exploit this latter, less well-known dimension. Brooks advocated rapid prototyping and incremental development as promising directions. Indeed, both are examples of practices that support flexibility and learning to cope with uncertainty.

But why and how do flexibility and learning increase economic value? I address the question on an intuitive level first, through a qualitative typification of projects that follow the two stereotypical process styles. The typification leads to caricaturized models that allow a concrete, quantitative demonstration.

## Sequential vs. iterative projects

Figure 2a illustrates the sequential stereotype. Because benefits arrive some time after product delivery, the return on investment relies on cost savings or income generated in the

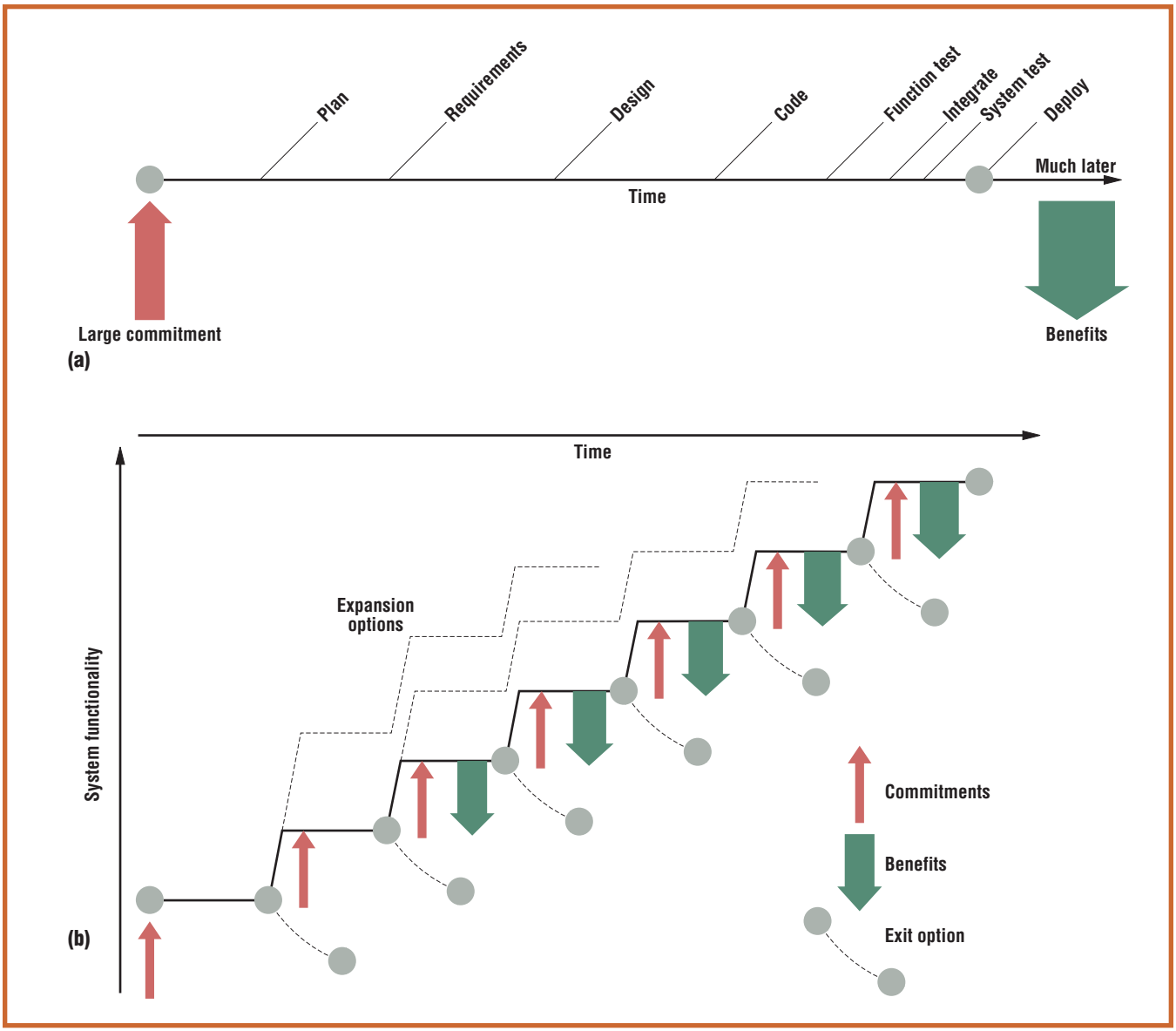


**Figure 2. Depictions of (a) sequential and (b) iterative projects. The width of the arrows indicates the size of the commitment or the amount of derived benefits.**

long term. Delayed benefits translate into lower current worth, so to break even in present-value terms, such projects require larger overall benefits spread over longer postdelivery periods.

In low-uncertainty settings, late benefits aren't as dramatically penalized because a lower return compensates for the risk borne. Low-uncertainty projects have clear goals, well-understood requirements, and straightforward benefits. Because they're typically driven by internal needs, such as automation and productivity, rather than market forces or an external user base's ambiguous needs, their uncertainties tend to be internal and technical. Neither process flexibility nor learning ability is particularly beneficial under these conditions: development can proceed predictably from one milestone to another without requiring multiple passes. Rigorous upfront planning is justified when flexibility is unnecessary and the odds of adhering to a well-mapped-out plan are high. In this way, avoiding rework maximizes economic value.

In contrast, an iterative project resembles a stage-gate process with increasing cumulative stakes. Figure 2b depicts this stereotype. As the project moves up the staircase, uncertainty decreases. Keeping the iterations small keeps the stakes low at each stage. The stages are guarded by gates where different actions are possible.

Iterations enable incremental delivery. Each iteration normally results in a functioning system, but with a limited feature set. This is possible because an iteration incorporates all the usual development activities—from planning through testing and deployment—on a small scale. When the project delivers high-priority features in early iterations, it can start generating significant business value relatively quickly.

However, incremental delivery makes economic sense only when the customer can get value from an incomplete product. Some systems, such as enterprise software, often provide business value with partial functionality delivered in small increments, but others, such as command-and-control systems, can be nearly all-or-nothing propositions.

Iterative and incremental development supports learning and flexibility. Each iteration provides a learning opportunity, and each step in the process involves a decision point at which the project team can put the learning outcomes to good use. Many options exist at these points. If the project's yield falls short of expectations, the project stakeholders can abort it, avoiding further losses. This choice constitutes an *exit* option. Or, the stakeholders can increase, change, or decrease the project's scope, giving rise to *expansion, switching,* and *contraction* options. Such options add economic value by reducing risk and improving the project's prospects. They're not of much value when uncertainty is low but can be worth a lot when uncertainty is high. Consequently, high-uncertainty settings economically favor iterative projects when incremental delivery is feasible.

### An iterative project's value

Consider a multi-iteration project that allows delivery of business value in functional increments of arbitrary size. Each iteration $t$ is of a fixed duration, or *period*. Thus the project delivers a functioning system with an extended set of features at the end of each period. I make three additional assumptions regarding the project characteristics.

First, the project's total cost is fixed at US$900K, with every iteration costing the same amount—that is, an iteration's cost equals the total cost divided by $n$, where $n$ is the total number of iterations.

A second assumption is that the project starts accruing benefits after the first iteration. Benefits are cumulative: when an iteration ends, the features implemented during that iteration continue to generate the same amount of benefit in each subsequent period.

The third assumption concerns uncertainty and learning. The project's benefits are uncertain, subject to three factors:

- *Outcomes*. Iterations result in *optimistic* or *pessimistic* outcomes. The optimistic estimate of the next iteration's per-period benefit equals the actual outcome of the current iteration. The pessimistic estimate of the next iteration's benefit equals half its optimistic estimate. Initially, optimistic and pessimistic outcomes are equally likely.
- *Learning power*. Uncertainty decreases gradually with each iteration, so the project's future prospects depend on its past performance. If an iteration yields an optimistic outcome, the likelihood of a pessimistic outcome in the next iteration decreases by $k \times 100$ percent (where $0 \le k \le 1$). Conversely, if an iteration yields a pessimistic outcome, the likelihood of an optimistic outcome in the next iteration decreases by $k \times 100$ percent. The factor $k$ represents an iteration's *relative learning power*, or the ability of a single iteration to resolve project uncertainty.
- *Embedded flexibility.* An exit option separates consecutive iterations. After each iteration, stakeholders reevaluate the project's prospects. If its prospects are poor, they exercise the exit option, aborting the project. Note that individual outcomes don't determine the project's ultimate success or failure. A pessimistic outcome can generate sufficient future benefits to outweigh the future costs. The exit decisions are dynamically based on future costs and outcomes alone.

Table 1 shows the calculation of total period and lifetime benefits for a four-period, three-iteration project, conditional on successive optimistic and pessimistic outcomes. In this example, the first iteration's optimistic and pessimistic benefit estimates are $200K and $100K per period, respectively.

Suppose each iteration's relative learning power is estimated to be 50 percent ($k = 0.5$). Figure 3a depicts a decision tree corresponding to this project. The tree's upper and

**High-uncertainty settings economically favor iterative projects when incremental delivery is feasible.**

**Table 1**

**Period and lifetime benefits (US$, thousands) with successive optimistic and pessimistic outcomes**

| | Period 1 | | Period 2 | | Period 3 | | Period 4 | | Lifetime total | |
|---|---|---|---|---|---|---|---|---|---|---|
| Iteration | Optimistic | Pessimistic | Optimistic | Pessimistic | Optimistic | Pessimistic | Optimistic | Pessimistic | Optimistic | Pessimistic |
| 1 | 0 | 0 | 200 | 100 | 200 | 100 | 200 | 100 | 600 | 300 |
| 2 | 0 | 0 | 0 | 0 | 200 | 50 | 200 | 50 | 400 | 100 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 200 | 25 | 200 | 25 |
| Period total | 0 | 0 | 200 | 100 | 400 | 150 | 600 | 175 | 1200 | 425 |

lower branches correspond to optimistic and pessimistic outcomes, respectively. For each iteration,

- the figures in brackets indicate the cost to be incurred if the iteration is undertaken;
- the figures under the Probability columns indicate the changing probabilities of iteration outcomes according to the uncertainty model;
- the figures under the Outcome/P*n* total columns indicate the period benefits conditional on the associated iteration's outcome. For each node, the top number is the per-period benefit for iteration $n - 1$, and the lower, blue number is the total benefit for period $n$.

We compute the project's expected net value by folding the decision tree back, starting with the leaf nodes and moving toward the root. The nodes under the Cost/Net value columns involve exit options. These nodes mark the decision points. The lower, green number gives the residual expected net value for each decision point. The decision to continue or exit depends on the project's future outlook, represented by the subtree rooted at the associated node. If the decision is negative, the exit option is exercised and the project aborted. The expected net value corresponding to a decision node equals zero when the optimal decision is to exit.

Consider a positive decision first by calculating the expected net value at node A. This node corresponds to the exit option directly after iteration 2, following two prior optimistic outcomes. The probability of another optimistic outcome for iteration 3 is relatively high, at 87.5 percent. According to table 1, the total estimated benefit for this state is \$600K. The probability of a pessimistic outcome is $1 - 0.875 = 12.5$ percent, with a total estimated benefit of \$500K (\$100K from iteration 3, plus \$200K each from iterations 1 and 2). So, the total expected benefit is $\$600K \times 0.875 + \$500K \times 0.125 = \$587.5K$.

If we undertake iteration 3, the project incurs a cost of \$300K, resulting in a net benefit of $\$587.5K - \$300K = \$287.5K$. Because the net benefit is positive in this case, we forego the exit option, undertake iteration 3, and set the expected net value corresponding to node A to \$287.5K. This amount represents the project's residual value after two optimistic outcomes.

Now consider a negative decision at node B. In this case, not only are the outcomes' probabilities reversed, but the total benefit estimates for iteration 3 are relatively low. This is because iteration 1 and 2 outcomes were pessimistic, halving the benefit estimate twice. At node B, the optimistic estimate for iteration 3 is \$50K, whereas the pessimistic estimate is a mere \$25K. The expected total benefit after iteration 3 is consequently $\$200K \times 0.125 + \$175K \times 0.875 \approx \$178K$, not enough to offset the \$300K it would cost to execute it. (Table 1 shows a total period benefit of \$175K corresponding to the bottommost branch.) We therefore exercise the exit option, setting the expected net value corresponding to node B to zero.

Folding the decision tree back in the same manner ultimately yields a value of approximately \$4K at the decision tree's root (node C). This is the overall project's expected net value. The iterative project is therefore expected to barely break even.

In the iterative stereotype, a future iteration's execution depends on the outcome of past iterations. The just-completed iteration's outcome determines the next iteration's estimate. So, project stakeholders decide to exit

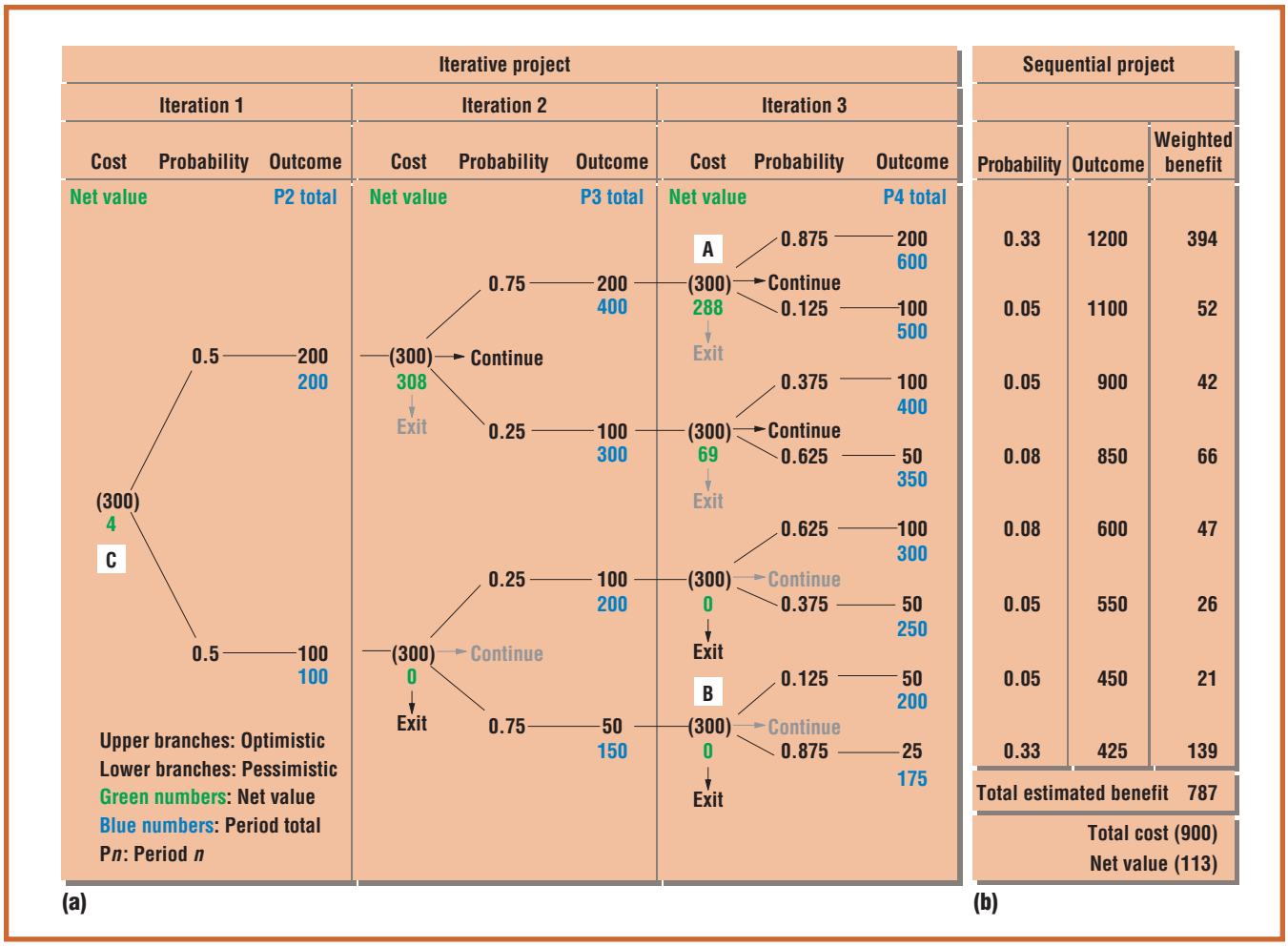


**Figure 3. Valuation of (a) a three-iteration project and (b) an equivalent sequential project. Amounts are in thousands of US dollars.**

or continue based on a "yesterday's weather" principle. When the next iteration's cost (the commitment immediately required to continue the project) exceeds the estimated net value of the remainder of the project, the stakeholders cancel the project, avoiding future losses.

## A sequential project's value

To contrast this strategy with a more traditional one, consider an equivalent sequential project. To avoid bias, we assign the sequential project the same expected total cost and total benefits under similar outlooks, but unlike the iterative project, we carry out the project in a single increment with no intermediate decisions. The sequential project's total cost is set to \$900K as before. Its eight possible outcomes correspond to the decision tree's eight final branches for the three-iteration project. Figure 3b specifies these outcomes along with their associated probabilities. To maintain the equivalence, the probability of each outcome of the sequential project equals the product of the probabilities along the corresponding path in the iterative project's decision tree. The outcome's benefit equals the cumulative benefit along this path (the sum of the lower, blue figures under the Outcome/P*n* total columns in figure 3a), as table 1 illustrates.

The sequential project's net value is its total expected benefit of \$787K less the total cost (\$900K), yielding a negative result of –\$113K. The total expected benefit is the sum, over all outcomes, of the weighted benefits (last column of figure 3a), where the weighted benefit is simply the product of an outcome's total benefit with the associated probability. The sequential project's negative net value makes it unattractive. Although its iterative counterpart is expected to break even, the sequential project will likely incur a comparatively significant loss even though the projects' uncertainty, costs, and benefits are similar.

Then what precisely is responsible for the

## Further Reading

Several resources discuss present worth and time value concepts (which I excluded from the article) in the software development context. John Favaro and colleagues' article in the *Annals of Software Engineering* gives an overview,[1] and parts 1–6 of Steve Tockey's book, *Return on Software*, gives a more detailed treatment.[2] Tockey's text describes relevant economic techniques, including decision trees, in a way that is accessible to software professionals.

Jim Highsmith's *Agile Project Management* is a high-level account of the philosophy and principles of agile software development.[3] Chapters 2 through 4 stress the role of early benefits, learning, and risk management. The 2003 *Computer* article by Craig Larman and Vic Basili provides a comprehensive history of iterative and incremental development models.[4] Barry Boehm and Richard Turner's *Balancing Agility and Discipline* explains the many trade-offs involved in choosing the right process flavor for a software project.[5]

The Real Options theory addresses the value of flexibility and learning under uncertainty. A chapter I co-authored with John Favaro in *Extreme Programming Perspectives* includes a conceptual overview, examples pertaining to agile software development, and a reading list.[6] For the advanced reader, my *Engineering Economist* article illustrates a valuation technique capable of handling multiple sources of uncertainty in commercial software development.[7]

The May/June 2004 issue of *IEEE Software* contains a focus section on return on investment.[8] The articles by Todd Little and by Mark Denne and Jane Cleland-Huang tackle value capture with incremental development. Also included in the focus section is a guest editors' reading list with additional resources on software economics.

### References


1. J.M. Favaro, K.R. Favaro, and P.F. Favaro, "Value-Based Software Reuse Investment," *Annals of Software Eng.*, vol. 5, Sept. 1998, pp. 5–52.
2. S. Tockey, *Return on Software: Maximizing the Return on Your Software Investment*, Addison-Wesley, 2004.
3. J. Highsmith, *Agile Project Management*, Addison-Wesley, 2004.
4. C. Larman and V.R. Basili, "Iterative and Incremental Development: A Brief History," *Computer*, June 2003, pp. 47–56.
5. B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, 2003.
6. H. Erdogmus and J. Favaro, "Keep Your Options Open: Extreme Programming and the Economics of Flexibility," *Extreme Programming Perspectives*, L. Williams et al., eds., Addison-Wesley, 2002, pp. 503–552.
7. H. Erdogmus, "Valuation of Learning Options in Software Development under Private and Market Risk," *The Engineering Economist*, vol. 47, no. 13, 2002, pp. 304–353.
8. *IEEE Software*, special issue on return on investment, May/June 2004, pp. 18–75.


difference in net values? The short answer is, of course, learning and flexibility. The exit options give the iterative project a flexibility that's lacking in the sequential project. Iterations enable learning, and the "yesterday's weather" approach allows optimal exit decisions based on the learning outcomes. The result is the pruning of the decision tree's unprofitable lower branches (figure 3a), which in turn increases the iterative project's net value.

## The effect of learning power and process granularity

In the iterative project's uncertainty model, the parameter $k$ (where $0 \le k \le 1$) represents a single iteration's relative learning power within the project. For $k = 0$, iterations have no learning power—that is, the underlying stochastic process is memoryless. However, this boundary value is hypothetical. In reality, $k$ rarely equals 0 because some learning almost always takes place. For $k = 1$, iterations have maximum learning power. A single iteration resolves all uncertainty, and the outcomes of all future iterations are known once the first iteration's outcome has been revealed.

Note that learning here is cumulative. If $k = 0.5$, a single iteration reduces uncertainty by 50 percent. Then, cumulative learning power over two iterations equals $0.5 + (0.5)(0.5) = 0.75$—that is, two successive iterations reduce uncertainty by 75 percent. Cumulative learning power approaches its upper limit of 1 when the number of iterations approaches infinity.

As relative learning power increases, flexibility becomes easier to leverage. Consequently, the iterative project's net value increases linearly relative to the net value of the comparable sequential project, as figure 4a shows. The difference between the iterative project's net value and its sequential counterpart's net value is close to \$100K when $k = 0$, but it increases by about 40 percent (to almost \$140K) when $k = 1$. The higher the learning power, the better an iteration's ability to resolve uncertainty and the better the relative economics of iterative and incremental development. This insight is fundamental, and independent of the uncertainty model's specifics. It stresses the importance of learning in high-uncertainty environments.

Under uncertainty, net value also tends to increase as process granularity increases. Process granularity increases decision-making opportunities (in the current example, the number of exit options), which in turn increases flexibility. Figure 4b demonstrates this effect for $k = 0.5$. The figure compares the net values for equivalent projects, from one to four iterations. The projects are equivalent in that they all have the same total cost and total expected benefit, with iteration costs and outcomes adjusted accordingly. For example, I adjusted the iteration cost of the four-iteration project to one-fourth of the three-iteration project's total cost.

The net value curve in figure 4b exhibits di-

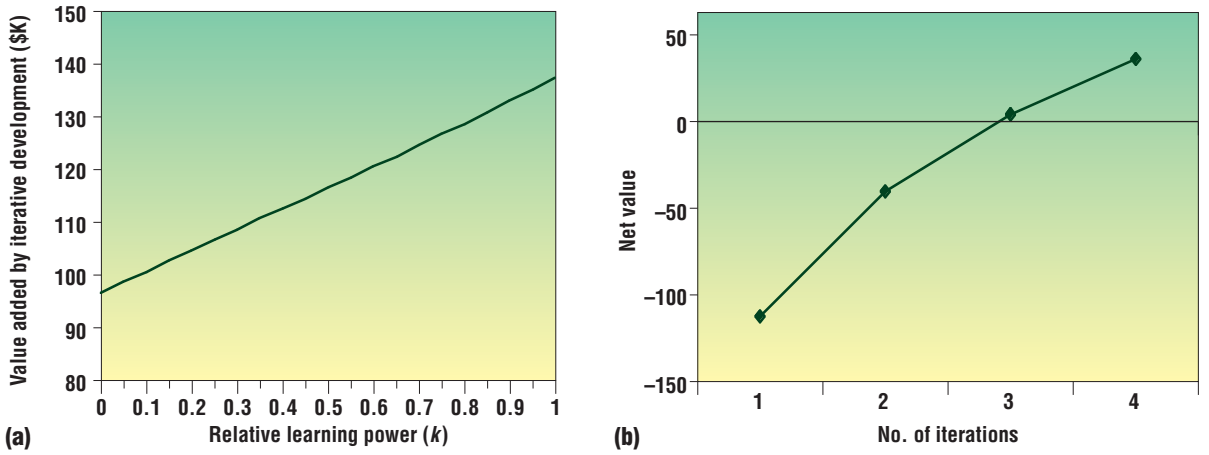


**Figure 4. Effect of (a) learning power and (b) number of iterations on economic value created.**

minishing returns. It isn't possible to indefinitely add significant value by shortening the iteration size and increasing its frequency. This behavior is consistent with the intuition that projects have an optimum iteration size. It results partially from the dampening effect of uncertainty reduction: uncertainty decreases from one iteration to another by a certain percentage, but cumulative reduction never quite reaches 100 percent.

The latter analysis ignores the effect of transaction costs, however. An iteration's cost normally includes two components: a fixed overhead that's independent of the iteration's size (which I don't model) and a variable component that is proportional to its size (which I do model). The fixed overhead—the transaction cost—would cause the net value curve in figure 4b to exhibit a negative returns behavior after reaching a maximum, strengthening the argument for the existence of an optimum iteration size.

I performed these comparisons on an equal expected cost-benefit basis to abstract away variations in planning and rework overhead incurred; I didn't assume any cost or benefit advantage for one stereotype over the other to avoid bias. For highly certain projects, such variations and present worth effects can indeed ultimately determine economic outcomes.

Real-world projects are characterized in many more dimensions than can be modeled with caricaturized representations. The models used here nevertheless account for two fundamental sources of economic value under uncertainty: learning power, which captures gradual resolution of uncertainty, and process granularity, which captures embedded flexibility.


## Acknowledgments

I thank the reviewers for their invaluable suggestions.


## References


1. *The CHAOS Reports*, Standish Group, 1994–1998; www.standishgroup.com/chaos_resources/index.php.
2. B. Boehm, "Get Ready for Agile Methods, with Care," *Computer*, Jan. 2002, pp. 64–69.
3. W. Royce, "Managing the Development of Large Software Systems," *Proc. 9th Int'l Conf. Software Eng.*, IEEE CS Press, 1987, pp. 328–338 (reprinted from *Proc. WESCON*, 1970).
4. C. Larman and V.R. Basili, "Iterative and Incremental Development: A Brief History," *Computer*, June 2003, pp. 47–56.
5. C. Larman, *Agile and Iterative Development: A Manager's Guide*, Addison-Wesley, 2004.
6. B. Boehm, "A Spiral Model for Software Development and Enhancement," *Computer*, May 1988, pp. 61–72.
7. F.P. Brooks, "No Silver Bullet: Essences and Accidents of Software Engineering," *Computer*, Apr. 1987, pp. 10–19.
8. B. Boehm, "Value-Based Software Engineering," *Software Eng. Notes*, vol. 28, no. 2, Mar. 2003.
9. J.M. Favaro, K.R. Favaro, and P.F. Favaro, "Value-Based Software Reuse Investment," *Annals of Software Eng.*, vol. 5, Sept./Oct. 1998, pp. 5–52.


## About the Author


**Hakan Erdogmus** is a senior research officer with the Software Engineering Group at the National Research Council's Institute for Information Technology in Ottawa, Canada. His current research interests center on agile software development and software engineering economics. He received his doctoral degree in telecommunications from the Institut National de la Recherche Scientifique, Université du Québec, and a master's degree in computer science from McGill University, Montréal. He is a member of the ACM and the IEEE Computer Society. Contact him at the National Research Council of Canada, 1200 Montreal Rd., Ottawa, Ontario, Canada K1A0R6; hakan.erdogmus@nrc-cnrc.gc.ca.