Software Design as an Investment Activity: A Real Options Perspective

KEVIN SULLIVAN
University of Virginia
sullivan@cs.virginia.edu
http://www.cs.virginia.edu/∼sullivan
Tel. (804) 982-2206; FAX: (804) 982-2214

PRASAD CHALASANI
Los Alamos National Laboratory
and Carnegie Mellon University
chal@cs.cmu.edu
http://www.c3.lanl.gov/∼chal

SOMESH JHA
Carnegie Mellon University
sjha@cs.cmu.edu
http://www.cs.cmu.edu/∼sjha

VIBHA SAZAWAL
University of Virginia
email@cs.washington.edu

June 3, 1997; Revised September 28, 1998

Abstract

Key software design principles–e.g., information hiding, spiral processes, and guidelines for timing design decisions—remain idiosyncratic, ad hoc, and not unified or explained in theory. This makes them hard to teach, learn, and use in principled ways. We see the potential to build stronger foundations for software design by viewing it as an investment activity and by relating design concepts directly to well developed financial investment theories. In this paper we focus on the specific issue of the value of flexibility in the face of uncertainty. Uncertainty puts a premium on flexibility to change products and plans, but flexibility also incurs costs. The theories on which earlier approaches to software engineering economics were based—primarily static net present value—cannot account for the value of flexibility. Real options theory, which seeks to make this value tangible in capital investment through an analogy to financial (e.g., stock) options, provides a framework within which we analyze software design concepts. We sketch our interpretation of important design principles in real options terms, making the idea precise for the specific case of optimal timing of design decisions. We discuss the validity of the approach for software; impediments to quantitative application; and how qualitative options thinking can nevertheless improve design decision making.

1 Introduction

Thirty years after software development was named an engineering discipline, contemporary software design doctrine remains largely atheoretical. Concepts such as information hiding [36], spiral process models [8], and the delaying of design decisions [21] remain idiosyncratic, ad hoc, and not unified or explained by serious mathematical theory. This lack denies software development full status as an engineering discipline. It makes it hard for people to teach and to understand software design, especially the deep but hidden connections among ideas. What is the relationship of information hiding to spiral models, for example? We believe that it also denies designers the ability to reason most effectively about design tradeoffs.

This paper contributes to a theoretical account of core software design principles. A central idea that software design is usefully viewed as an investment activity under uncertainty, the goal of which is to optimize value added over time. Applying investment ideas in software engineering is not new [6]. What is new in this paper is the use of more sophisticated financial engineering ideas, in general, and of theories about the value of flexibility in capital projects, in particular. We aim to outline a new economic account of flexibility-oriented software design principles. We do not claim to provide a fully developed or tested theory, but rather a framework within which to gain a richer and better unified understanding of a set of critical notions in software design.

Uncertainty is pervasive in software design. Many principles provide means to manage value under uncertainty. Flexibility is essential because it enables the designer to change course dynamically to avoid losses and to exploit opportunities as uncertainties are resolved over time. It is thus natural that basic software design principles focus on the creation and use of flexibility to change products, time design decisions, respond to risks in development processes, and so forth.

We provide an initial outline of a rigorous account the value of flexibility in software, for which there is no adequate account in the literature. Until recently, even the finance world has lacked means to grapple with the value of flexibility in capital investment. Real options theory seeks a solution by interpreting flexibility in terms of "options" whose values are estimated using techniques for pricing financial (e.g., stock) options. We appeal to real options to help us to better understand key software design principles.

Section 2 defines software design as an investment activity whose goal is value added over time. Section 3 discusses the structural orientation of current design principles and suggests a value-based reorientation. Section 4 discusses inadequacies of traditional software engineering economics. Section 5 introduces real options. Section 6 outlines options pricing by dynamic programming. Sections 7 and 8 present examples. Section 9 shows how real options can provide a sound basis for qualitative design principles related to the timing of design decisions. Section 10 discusses an options interpretation of information hiding. Section 11 gives an options interpretation of the spiral model. Section 12 addresses related work. Section 13 discusses theoretical limitations. Section 14 concludes.

2 Software Design as a Value-Seeking Activity

We begin with one basic axiom: Software design broadly speaking is an activity of investing valuable resources under uncertainty with the goal of optimizing value added over time [43]. There is always some definition of value for which the axiom is valid. Design can be characterized as a multiobjective optimization problem, for example, in which value is a vector measure of performance, cost and risk with Pareto-optimal solutions as acceptable outcomes.

However, we take value in purely economic terms. Value means money. In our formulation, software design seeks to make those investments (e.g., in architecture, restructuring, etc.) that optimize profit. This approach frames the software design as an optimization problem in financial economics. The equation of design to investing provides a bridge that will permit us to explore the idea that existing ad hoc design principles can be explained in terms of mathematical finance.

The axiom is reasonable. First, it is exactly the axiom at the heart of modern corporate finance: The overriding objective of management of the publically held firm is to make those decisions that maximize the owners' wealth as reflexted in the present value of the firm in light of future uncertainties (risks and opportunities)[9]. Second, seeing design as a value-seeking activity is not new: "Designers see and seek value in new designs [2]." Third, some successful software companies have value-added as an explicit goal. Microsoft, for example, has a culture that puts a strong emphasis on value added. "Everybody in a business unit has exactly the same ... job description, and that is to ship products. Your job is not to write code, your job is not to test, your job is not to write specs. Your job is to ship products.... You're trying not to write code. If we could make all this money by not writing code, we'd do it [12, p.45]." Fourth, prominent managers such as Strassmann bemoan the prevalence of software design decisions that fail to focus on value added:

> The proper measurement of the success of software investments requires more than compliance with models of technical perfection. Technical excellence of programming code is highly desirable, but clearly insufficient. Nowadays, the most profitable and popular software in the world is notorious for its bugs and glitches. Economic utility and independent measures of customer satisfaction must be the ultimate arbiters of all judgement about the utility of software [41, p. 147].

Principles for achieving technical excellence are central to software engineering practice and research. We already have many principles for structuring products and processes to achieve technical ends. The problem that we address in this paper is that we lack an adequate account of how some of these principles operate, at a fundamental level. The intellectual basis of our field would be improved by the existence of such an account. Moreover, if you accept that value added is the objective of design, then today we lack well founded guidance on how to use key techniques, even in principle, to optimize products and processes for value added.

3 From Structure to Value Added as Desideratum

Software design principles today generally do not make explicit the objective of value added. They tend to emphasize structural concerns instead. Of course structure is linked to value, but the link has not been made clear. To optimize for value, a designer must be able reason about how and under what conditions investments in structure add value. A decision to restructure a system, for example, incurs costs that must be weighed against expected benefits. The issue is economical as well as technical. Unfortunately, the connections from structure to value remain murky, making it hard to trade off costs and benefits in a principled way.

Consider three design issues: product structure, process structure, and the timing of commitments to design decisions. In the product dimension, design principles focus on modular structure. Information hiding [36] is a seminal concept. More recent work on software architecture [40] continues this trend. The focus on structure is clear: Shaw and Garlan say, "As the size and complexity of software systems increase, the design and specification of overall system structure become more significant issues than the choice of algorithms and data structures.... [40, p.1]." Technical issues such as module interconnection formalisms dominate. The value added concept is not made explicit.

Process criteria are largely structural, too. They define stages of design, their ordering, and transition criteria [8]. The waterfall model pays little regard to value added. The spiral model is more value-oriented in that a project "get[s] started by a hypothesis that a particular operational mission (or set of missions) could be improved by a software effort [8]." We see this as a hypothesis that such an effort is expected to add value to the enterprise. Nevertheless, the value concept remains implicit in the formulation of the model, and no clear connection is drawn from the spiral model concept to any notion of value optimization.

In timing of design investments, we rely on informal rules of thumb. One rule that we have encountered says that a designer should delay each decision for as long as possible, to maximize the information available when the decision is made. Later in this paper we show that this heurism is not correct in general. We need a sound basis for reasoning about timing so as to be able to make principled tradeoffs, e.g., for time to market. More generally, we need a sound basis for reasoning about the conditions under which such heurisms are valid.

We argue that there is a useful common way to look at these design issues. Design principles as taught today are rules of thumb that, in essence, serve to promote flexibility. Flexibility has value under uncertainty. Uncertainty pervades software design. Our design principles thus work to a considerable degree by adding value in the form of flexibility in products and projects. This paper presents an analysis that is meant to support the claim that we can gain a deeper understanding of current design principles by making the connection to value explicit in terms of the value of flexibility under uncertainty. Information hiding provides value in the form of flexibility to change products as uncertain future demands become known. Spiral processes provide value in the flexibility to change directions as uncertainties (risks) are resolved. The flexibility to delay making commitments to risky design decisions (e.g., to ship code) adds value by allowing the designer to avoid actions that turn out to be unprofitable.

4 Valuing Flexibility in Products and Projects

Flexibility is critical in software design, but it is not the primary objective. The objective is value added. Flexibility has benefits, but in many cases it also has costs. The questions, then, are how and under what circumstances do investments in flexibility add value. Is it optimal to pay for flexibility in a given case, or is it best not to—to trade flexibility for another benefit? Modern software design principles provide little or no guidance for answering such questions.

We need a theoretically plausible account of the value of flexibility in software. However, we want to avoid generating more ad hoc and idiosyncratic explanations. We thus appeal to an established investment theory for an account of the value of flexbility under uncertainty. Appealing to finance is not a new idea. Boehm [6] pioneered applications of economics to software. What is astonishing, though, is that the value of flexibility has remained mysterious not only in software, but also in corporate finance and economics. Theoretical treatments are only now being accepted widely in the academic business community.

The traditional approach to valuing an asset (such as a project) in business and software engineering [6, 16] uses the static *net present value* (NPV) concept. The NPV of an asset is taken as the value that it adds to an enterprise. An asset is valued by estimating the discounted cash flow stream that it generates. Static NPV accounts for uncertainty over an asset's costs and benefits by averaging the cash flow streams that it would generate in each scenario.

This approach values real assets as if they were passively held bonds, the terms of which cannot be changed by the bond owner no matter how well or poorly the future turns out. The problem is that real assets are not passively held. To the extent that an asset is flexible, its owner can intervene actively to optimize value as the uncertain future is revealed. Averaging cash flow streams over all possible futures ignores the value of the ability to exploit flexibility, e.g., by abandoning a project if markets dwindle [13, 47, 48, 9, 24].

There are several theoretical formulations in finance and decision theory able to capture the value of flexibility, including dynamic net present value and extended decision analysis. They all share the ability to model dynamic optimization. Designers can change products and plans as new information comes to light. One approach, emerging from theoretical work in capital investment analysis, goes by the name of real options theory. The idea is to treat flexibility as an option—the right without an obligation to make an investment contingent on a future outcome—and to value flexibility using techniques related to financial option pricing. It is is this formulation that we explore in this paper. The topic of options pricing is rich and deep. Merton and Scholes received the Nobel Prize for their work in this area. We seek to use such ideas to analyze and explain software design principles.

5 Introduction to Real Options

A strong analogy between flexibility and financial options, such as call options on stocks, is what suggests the adaptation of financial options pricing and related techniques to value

flexibility in real assets and development projects. To understand this idea requires an understanding of basic financial options and options pricing. A financial option confers, for a period of time up to some expiration date the right but not the obligation to purchase (if a *call* option) or to sell (if a *put* option), some financial asset, such as a stock, at a given price which is called the strike price. To exercise an option is to pay the strike price—to make an investment—in return for the asset. Such a trade is irreversible, and the option itself is nullified. An option whose strike price is greater than the price of the asset is said to be *out of the money*. No rational investor would exercise that option: it would produce a loss. An option whose strike price is equal to or less than the asset price is *in the money*.

5.1 Basic Financial Options Concepts

Several options pricing concepts bear mentioning because of their relevance, by analogy, to decision making issues in software design. This subsection reviews these notions. The next subsection relates financial options to capital investment decision-making, i.e., to real options. The following section then draws the connections from real options to software design.

First, even an out of the money option on an asset has value if the uncertainty about the future makes it possible that the asset value will exceed the strike price before expiration. The value of the option today is the value of the flexibility it confers to exploit a possible future change in asset value. The option gives its owner the right to decide to invest after observing whether or not that value becomes attractive. That is a right that generally has real value.

Second, the value of an option depends on numerous factors, including the following:

- the current value of the asset (benefit of investing now) and the option strike price (cost to invest): option values increase with asset value and decreasing with strike price
- as a result of the previous point, any discrete drop in the value of the asset—as would result from the payment of a dividend on a stock—reduces the value of an option
- the nature of uncertainty over the future value of the asset: option values increase with the variance (risk) in the asset value because the option value depends on the possibility that the asset might turn out very valuable
- the time to expiration: options values decrease with the time left, as the likelihood of a dramatic change in value diminishes with the time remaining (under certain statistical assumptions about how asset values change over time)
- macro-economic factors, such as the risk-free interest rate

Third, being in the money does not imply that an option should be exercised immediately. Exercising is an irreversible act with two costs. The direct cost is the strike price. The opportunity cost is the lost value of the option: of the flexibility to decide whether to invest or not. It is worthwhile to exercise only when the asset value exceeds the sum of the strike price and

option value [13]. It can be optimal to hold rather than exercise an option, even one that's in the money.

Fourth, the optimal time to exercise an option is not always obvious. Options theory is deeply concerned with optimal policies for the timing of investment decisions. In the simple case of a call option on an asset that is not subject to dividend payments, a theorem states that it is optimal to wait as long as possible—until just before expiration—to decide whether or not to invest [13]. Nothing is lost by waiting, and potentially valuable information on the asset value is obtained. However, if the asset is subject to dividends, then early exercise is sometimes justified: Holding the option forgoes the benefits of owing the asset now, i.e., of capturing those dividends. For assets subject to dividends, there is an opportunity cost not only to investing early, but also to waiting. In real options thinking, any sharp drop in the value of an asset can be thought of as a dividend. A competitor's entry into a market is one example. Options thinking gives a principled way to reason about how to balance countervailing incentives to hurry or delay investment decisions [13].

5.2 From Financial Options to Real Options

Real options theory was developed to address the problem in capital budgeting discussed above. Traditional capital budgeting (project) justifications use NPV-based valuations. However, NPV overlooks the value of flexibility. That means that traditional techniques undervalue (and have a harder time justifying) projects that have significant value in the form of flexibility. Projects that create strategic growth options can have this character: they might even have negative NPV but still be valuable because they create options for potentially lucrative follow-on investments. Thus, traditional decision-making criteria can lead to sub-optimal capital investment decisions.

The real options field opened in 1977 when the economist Myers noted that, "part of the value of a firm is accounted for by the present value of options to make further investments on possibly favorable terms [32, p. 148]." Myers saw that, all else equal, the firm that with flexibility to exploit potentially lucrative opportunities, e.g., by dint of having made early strategic investments, is worth more that the firm that doesn't. He also saw that that value was in the form of *options* (his emphasis), and that financial options pricing might used to value that flexibility. Real options theory was thus developed precisely to address the inability of traditional capital budgeting to address strategic value.

Today the options Myers saw are called growth options [24, 32]. Many other real options are now recognized. Types of real options that have been analyzed include the following [13, 48]:

- defer decisions to invest until optimal to do so [13, 28, 32];
- default on a project that is structured in phases or abandon a project for its salvage value [19, 30, 33];

- expand or contract production if favorable or unfavorable conditions emerge [38];
- switch materials used in a product, or switch to producing another product as supply and demand conditions change [25]
- invest in a market should it become lucrative to do so (growth options) [24, 32];
- select the better of two (or more) alternatives [42]
- options on options (used to model phased investments) [45];
- interactions among real options [10, 46].

Real options theorists have developed a rich theory and body of knowledge on the value of flexibility in a wide variety of forms encountered in capital investment and corporate strategy. To help make the ideas concrete, consider an example involving the flexibility to switch the materials used to generate a product. Suppose a manager has to decide between investing in a less expensive but inflexible power plant that burns only oil, or a more expensive one that has the flexibility to be switched between oil and coal [26]. The price of coal is stable, but the future price of oil is uncertain. Is it worthwhile to pay more for flexibility?

Clearly there is a price at which the flexible plant is too costly, despite the risk of a price rise in oil. The value-maximizing decision is to invest in the inflexible plant. On the other hand, flexibility has value today because it confers the ability to save future expenses contingent on a more or less probable price rise. So the manager should be willing to pay somewhat more for the flexible plant. How much more? What are the relevant factors that must be considered in making the decision?

In a nutshell, the value-maximizing decision requires the to manager assess the value today of each of the two choices, net of costs, and to select the greater. The expected value of the inflexible plant is its expected benefits minus its expected costs, with uncertainties over both accounted for: $V_I = B_I - C_I$. Such costs and benefits are usually represented as expected present values of present and future cash flows discounted for time and risk. The expected value of the flexible plant, by contrast, is the value of its benefits assuming that it is operated using oil plus the value of the option to switch to coal should future oil prices be unfavorable, minus its costs: $V_F = B_F + O_F - C_F$. The flexible plant is the right—value-optimizing—choice if $V_F > V_I$. Assuming the benefits from the plants are equal when they burn oil, the expression evaluates to $O_F > C_F - C_I$. Does the value of flexibility exceed its added cost?

Of course, the question is obvious. What was unknown until recently was how to evaluate O_F. What is flexibility worth? What factors influence its value and how are they related? In other words, how do you reason properly quantitatively or even just qualitatively about the value of flexibility? Without an answer to this question, the decision-maker facing uncertainty—in particular the designer, who is usually in a situation of considerable uncertainty—lacks a principled basis for making value-optimizing decisions.

The insight on which real options theory is based is that flexibility can be seen as being analogous to a financial option, such as a call option on a stock, and that financial options pricing techniques can be used to estimate the value of flexibility. The outline is easy to sketch. The asset (cost savings) is the difference between the price of oil and that of coal. When the price of oil is high, the asset is valuable. If management invests in the flexible plant, it acquires a call option on this asset. The flexibility is closely analogous to the right to change a design decision for a predictable exercise price acquired by imposing an information hiding interface. The exercise price is the cost to reconfigure the plant to burn coal or to change the design decision. This option does not expire until the plant is decommissioned.

Having set up the problem as an options problem, and given estimates of the uncertainty in the future price of oil (i.e., its volatility) and other relevant factors, valuation techniques such as the use of the Black and Scholes equations [5] or a dynamic programming approach [14] can be used to estimate O_F.[1] The importance of this basic idea cannot be overemphasized: it gives the manager an ability to reason about a crucial but previously intangible source of value.

Even qualitative reasoning with options thinking is important: it provides the ability to reason in principled ways about the value of flexibility as a function of fuel price differentials, switching costs as a function of plant design, the nature of uncertainty over prices, and expected plant lifetime. The value of back-of-the-envelope calculations based on a powerful model is considerable. The notion that options value must be considered in finance is now reasonably widely accepted. Speaking on the value of flexibility to defer investment decisions—a topic we address in detail in the following sections—Ross concludes that,

> ... when evaluating investments, optionality is ubiquitous and unavoidable. If modern finance is to have a practical and salutary impact on investment-decision making, it is now obliged to treat all major investment decisions as option pricing problems [39, p. 101]. (Also quoted by Flatto [18]).

6 Pricing Financial Options

In order to understand how to value real options it is necessary to understand how financial options are priced based. In this section, we describe basic concepts in financial options theory, with a particular focus on the interplay between the value of an option and the optimal strategy for exercising it. First we define the mathematical notions and notation that we use. The we show formally how option pricing is related to the notion of optimal stopping—a notion that we already saw informally. For basic information the reader is referred to any of several excellent texts, including Hull [23] or Luenberger [27].

[1] There are some important theoretical limitations on the applicability of arbitrage-based techniques, such as the Black and Scholes approach. We discuss them at the end of this paper.

6.1 Decision Trees

Uncertainty and the value of flexibility in the face of uncertaint are at the heart of both software design and finance. Options pricing and related techniques are financial applications of general mathematical techniques used to support optimal decision making under uncertainty. There are both discrete- and continuous-time formulations of the key issues. In this paper, we employ a simple, discrete-time formulation. Our goal is to communicate the key concepts, so as to make our ideas reasonably clear; it is not to present a complete mathematical theory.

We model future uncertainty in one dimension by means of a discrete **decision tree** of finite depth N, where N represents the maximum number of future time steps (e.g. months, years, etc) that we wish to model. The root node at depth 0 represents the present time. Nodes at depth k represent possible states of the world at time k. The children of a depth k node v are the possible next-states at time $k+1$, given that the state at time k is v. If a node w is a descendant of v, we write $v \rightarrow w$. For $k = 0, 1, \ldots, N$, a **random variable** X is a mapping (or function) that associates with each node v, a real number $X(v)$. A random **process** is a sequence of random variables $\{X_k\}_{k=0}^{N}$, (often referred to briefly as the "process X_k") where for each k, $X_k(v)$ has non-zero values only for nodes at depth k. When we want to refer to the value of a random process X_k at a specific node v, we will often drop the subscript and just write $X(v)$. We define the special random variable $\delta(v)$ to be the depth of v.

As an example, consider tossing a fair coin N times. For each toss there are two outcomes, each equally likely. We represent the uncertainty in outcomes by a simple decision tree, called the **binomial tree**. Each non-leaf node has two children. Each of the 2^N paths represents a sequence of coin-toss outcomes. On any path, for $k = 1, \ldots, N$, the k'th branch is an up-branch if the k'th coin-toss lands heads (H), and it is a down-branch if it lands tails (T).

To each branch in an decision tree, we associate a **probability**. The sum of the probabilities of the branches emanating from a node is 1. In the example above, the probability of each branch is 0.5. For any node v, the probability that state v occurs, denoted $\mathbf{P}(v)$, is the product of the probabilities of the branches from the root node to v. If a node v has a branch to node w, we denote its probability by $\mathbf{P}(w|v)$. Clearly,

$$\mathbf{P}(w|v) = \mathbf{P}(w)/\mathbf{P}(v).$$

The **expectation** of a random variable X, is the probability-weighted sum of possible outcomes, denoted by $\mathbf{E}(X)$, and defined precisely as

$$\mathbf{E}(X) = \sum_{v} X(v)\mathbf{P}(v). \tag{1}$$

The concept of conditional expectation is important. Imagine we are in some state at time k, e.g., at node v at depth k. Then the **conditional expectation** of a random variable X, *given* that we are at v, is denoted by $\mathbf{E}(X|v)$, and is defined as

$$\mathbf{E}(X|v) = \sum_{w:v\rightarrow w} X(w)\frac{\mathbf{P}(w)}{\mathbf{P}(v)} \;=\; \sum_{w:v\rightarrow w} X(w)\mathbf{P}(w|v). \tag{2}$$

This is just a form of the familiar Baye's rule. Clearly this conditional expectation will in general be different from $\mathbf{E}X$, and will depend on which depth-k node v we're at. For instance in the coin-toss tree above, suppose the random variable H_k is the number of heads up to time k, on the path to a specific node in the tree. Then if v is at depth k and $m \geq k$, the conditional expectation $\mathbf{E}(H_m|v)$ will be higher if the path to v consists of more heads.

6.2 Decision Rules

We will use decision trees in this paper to model the timing of various investment decisions. For instance we might want to decide *when* (i.e. at which nodes in the tree) and *how much* to invest in building a software prototype. In general let us assume that at any node we are allowed c possible levels of investment, numbered $1, 2, \ldots, c$. We consider level 0 to represent no investment. A **decision rule** τ with respect to a decision tree T is a mapping from the set of nodes of T to the set $\{0, 1, \ldots, c\}$, with the restriction that on any path of the tree, there is at most one node v with a non-zero value of $\tau(v)$. In other words, a decision rule specifies a rule that we can follow as the state of the world changes along the tree. Whenever we are in a state v for which $\tau(v) \neq 0$, we invest at level $\tau(v)$. In the coin-toss binomial tree, an example of a decision-rule is: "invest at level 1 when the coin has landed heads 3 times", or more formally: $\tau(v) = 1$ if $H(v) = 3$, and $\tau(v) = 0$ otherwise. Those familiar with stochastic processes will recognize that decision rules are closely related to *stopping times*.

6.3 American Call Options

The simplest kinds of option is a call option. An **American call option** on an asset, say a share of stock, is a contract that gives the holder of the contract the *right* but not the obligation to buy a share of the stock at a fixed price called the **strike** (or **exercise**) price L on or before a certain **expiration** date of T time units. The holder thus has the "option" of deciding whether or not to exercise the contract, i.e., buy a share of stock at the strike price L. When the option is exercised or expires, the option ceases to exist. Thus option exercise is *irreversible*.

In the case of a financial option, the uncertainty involves the future value of the asset to which the option holder is entitled. The option gives its holder flexibility because it permits the person to wait to see how the uncertainty is resolved before deciding whether to invest in the the asset. Such flexibility clearly has value. Similarly, the presence of flexibility in a software product or development process provides value insofar as it gives the designer the ability to make a future investment in exploiting the flexibility (e.g., by changing a design decision) should it become value-enhancing to do so.

6.4 Uncertainty and Payoff

To describe an American call option formally, we need a model of the underlying uncertainty. It is typical to model the price of a stock as a depth-N decision-tree, with N being the time

to expiration, and $\{S_k\}$ a random process modeling the uncertain stock price. We can bow reason about when if ever to exercise an option. Clearly it makes no sense to exercise at a node v where $S(v) \leq L$. However, if $S(v) > L$, the optionholder can but is not obliged to exercise by investing the strike price L to obtain a share of the stock. In the case of a stock, the holder could then immediately sell the share in the market at $S(v)$, and make a profit of $S(v) - L$. The profit that can be realized by exercising the American call option at time k is thus $\max(S(v) - L, 0)$, which we refer to as the **payoff** $G(v)$. It is standard to denote $\max\{x, 0\}$ by x^+, so we can write the payoff as the random variable

$$G_k = (S_k - L)^+. \tag{3}$$

In other words, for any node v, the payoff $G(v) = (S(v) - L)^+$.

6.5 Optimal Exercise Strategy

What is the best exercise strategy for an American call option held at time k? An exercise strategy can be described by a decision rule τ that maps nodes to the set $\{0, 1\}$. If we are in state v, we exercise if and only if $\tau(v) = 1$. An example of an exercise strategy is: "exercise when the stock price exceeds a threshold λ, or when the expiration date is reached," which can be described by the decision rule τ where $\tau(v) = 1$ if $S(v) \geq \lambda$ or the depth $\delta(v)$ of the node v is N, and $\tau(v) = 0$ otherwise.

In reasoning about payoffs, it is necessary to take into account the time value of money: that a dollar tomorrow is worth less than one today. In order to discount future cash flows to the present time, we will have to assume that money can be borrowed or lent at a risk-free interest rate of r. Thus a dollar lent or borrowed at time k is worth $R = 1 + r$ dollar at time $k + 1$. It is common to refer to R as a **discount factor** since a dollar at time k, discounted to the present time (i.e. time 0) is worth $1/R^k$.

We can now discuss optimal exercise strategies. For a strategy τ and node v at depth k, the expected value of the strategy τ discounted to time k is denoted by V_k^τ, and is computed as follows. At any node w that is a descendant of v in the tree, if the option is exercised (i.e., $\tau(w) = 1$), the payoff is $G(w) = (S(w) - L)^+$, and if it is not exercised the payoff is 0. Thus in general at any node w we can write the payoff as $G(w)\tau(w)$, which is worth $G(w)\tau(w)R^{k-\delta(w)}$ at time k. Therefore the expected value of the strategy τ, discounted to time k, given that we are at a node v, is

$$V_k^\tau(v) = \mathbf{E}\left(G\tau R^{k-\delta}\middle| v\right). \tag{4}$$

Any rational option holder wants to choose the strategy τ that maximizes this expectation. We denote this maximum by the random variable V_k:

$$V_k(v) = \max_\tau V_k^\tau(v). \tag{5}$$

In other words, V_k is the best expected present value at time k realizable over all possible exercise strategies. We take this value to be (and refer to it as) the **option value** at time k, for reasons that will become clear shortly.

Since immediate exercise is a valid strategy at any time, the option value $V(v)$ must be at least as large as $(S(v) - L)^+$. In fact, if $(S(v) - L)^+ < V(v)$, this means that the immediate exercise strategy is *not* optimal, and that some other strategy will yield a strictly greater expected present value of payoff, under our assumed stock price model. Thus in this situation, it is beneficial to not exercise and wait. On the other hand, if $(S(v) - L)^+ = V(v)$, then there is nothing to be gained in waiting, at least under our assumed stock price model. In this case it *is* optimal to exercise immediately. Indeed it can be shown rigorously that the decision rule τ that achieves the maximum in (5) above is given by

$$\tau(v) = \begin{cases} 1 & \text{if } (S(v) - L)^+ = V(v) \text{ or } \delta(v) = N, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Let us look at the optimal exercise rule from a cost-benefit viewpoint. We can think of the strike price L as the "cost" of exercising the option, since this is the price one must pay to obtain a share of stock. Similarly, S_k is the benefit from exercising at time k, since this is the price one would obtain by selling the stock in the market. We just remarked above that it may not be optimal to exercise as soon as the benefit S_k exceeds the cost L. To explain this, it will be useful to view the option value V_k as representing the "value of the choice to exercise". When the option is exercised, the option and thus the flexibility to choose is killed, and this value is lost, so that V_k represents the *opportunity cost* of exercising the option. Thus exercising the option incurs two costs: the *direct* cost L and the opportunity cost V_k. From the discussion above, the optimal exercise strategy is to exercise when $(S_k - L)^+ = V_k$, which in cost-benefit terms can be stated as: *Exercise only when the benefit S_k equals the direct cost L plus the opportunity cost V_k.* This viewpoint is the one that is most useful in this paper.

In terms of real options and software design decision making, what this analysis suggests is that, if there is still uncertainty in the future, is it not always optimal to invest in an opportunity (e.g., to port a computer program to a new platform or to restructure a legacy system) as soon as it appears that doing so is expected to produce a profit. The expected profit has to be high enough to offset the loss of flexibility value involved in actually undertaking the project. Delaying preserves a valuable option to invest in the future, should conditions be favorable.

6.6 Options Pricing

To make the decision to invest or to delay in a principled way, we have to estimate the option value. To do this, we use techniques related to option pricing. In our specific formulation, with its finite time horizon, we can compute V_k for all k by a simple *dynamic programming* procedure [11]. First observe that $V_N = (S_N - L)^+$. This is clear both from formula (5) and from observing that, since the option expires at time N, there can be no advantage to waiting

to decide whether or not to invest. Stepping backward in time in the decision tree, we compute $V_k(v)$ at any depth k node v by

$$V_k(v) = \max\{(S_k(v) - L)^+, \mathbf{E}(V_{k+1}|v)/R\}. \quad (7)$$

In other words, the option value $V_k(v)$ at a depth k node v is the maximum of the immediate payoff $(S_k(v) - L)^+$ and the expected present value of the option value one time step ahead, given that we are at v. It can be shown that this backward-recursive formula for V_k and formula (5) are equivalent, regardless of the specific process that the stock price S_k follows.

7 A Simple Example

We now present an example that illustrates how real options concepts can be employed to inform design decisions. Suppose that a designer is faced with a decision whether to restructure a software system in order to "clean up" its modular structure which has degraded as the system has been maintained over time.

Committing to restructuring will incur costs. Suppose, for example, that restructuring will cost 1600 dollars.[2] The benefit, if any, will come in the form of reduced future maintenance costs. If the benefits were certain, the decision would be easy: the designer would simply compare the present value of future the future benefits, represented as a cash flow stream, against the present value of the cost of $1600.

The issue is more complex when the payoff depends on uncertain future conditions. Suppose for example that the demand for future changes is quite uncertain—that, for one reason or another, the system will have to be changed either often or hardly at all. As a simplification, let us assume that there are two possible futures: one that is favorable to an investment in restructuring, in which many changes are needed; and an unfavorable one, in which so few changes are required that it would have been better not to pay for restructuring. We represent the possible future payoffs as cash flow streams. Suppose that the payoff in the favorable scenario is 3300, and just 1100 otherwise.

7.1 The Nature of the Uncertainty

We model the uncertainty with a decision tree. Here we have a simple one-level decision tree, rooted at the present, with 1100 at one leaf and 3300 at the other. Next, we assign probabilities are assigned to each branch. Suppose that at present it is our expert judgement that each outcome has a probability of occurring of 0.5. Finally, we account for the depreciation in the value of money over time by assuming a discount factor of 1.1 per time period—a 10% discount rate. Thus, for example, a cash flow of 1100 dollars one period from now is worth 1000 dollars today, because 1000 invested today at the given discount rate would be worth precisely 1100 in the next period.

[2] Our numbers are fictional, and taken directly from Dixit and Pindyck [].

7.2 Static Net Present Value

Even in this simple example, the traditional software engineering economic approach to the investment decision, based on static NPV, does not produce an optimal decision. Recall that the traditional static NPV decision rule states that the investment should be made if the NPV is positive. The NPV turns out to be 400. Even considering the uncertainties involved, the decision under this rule is to restructure immediately, for a value added today of 400. The NPV is computed as an expected present value:

$$NPV = 0.5(-1600 + 1100/1.1) + 0.5(-1600 + 3300/1.1) = 400.$$

Suppose, however, that the designer can wait to invest until more is known about requirements changes. This flexibility is valuable. Investing now risks a loss of $-1600 + 1100/1.1 = 600$ should the unfavorable scenario emerge. Waiting permits the designer to invest only in the favorable scenario for a benefit of $-1600/1.1 + 3300/1.1 = 1546$. The value added by this dynamic strategy is greater than that added by investing today. Discounting and factoring in the likelihood of a favorable future, the dynamic NPV under the assumed probabilities is $(0.5)(3300/1.1 - 1600/1.1) = 773$. It's clearly better to wait.

7.3 Making the Option Explicit

Our analysis shows that the project with an option to restructure is worth more than the project restructured today, despite the positive NPV of restructuring. The real options "way of thinking" suggests that the software designer has to think of products and projects as portfolios of assets that include options, and that those options have to be created and managed carefully.

To make the analogy between the restructing decision and options clear, we recast our analysis into an options formalism. The cost to restructure is formulated as the strike price $L = 1600$ of an American call option on the expected benefits. This asset is worth S_t, the expected present value of the future profit stream from restructuring. S_t is a random variable. Letting subscripts denote time and discounting, $S_0 = 0.5 * (1100/1.1) + 0.5 * (3300/1.1) = 2000$. S_1 is either $3300/1.1 = 3000$ or $1100/1.1 = 1000$, each with probability $p = 0.5$.

The payoff of exercising now is $S_0 - L = 400$. A month from now it is $\max(S_1 - L, 0)$. With even odds this is either $3300/1.1 - 1600/1.1 = 1546$ or 0, for an expectation of 773. The value V_t of an option at any time t is the expected present value of future payoffs under the optimal exercise strategy. In our example, all uncertainty is resolved in the first period, so there are only two strategies: exercise now or next period. The value V_0 of our option is thus 773, since this is the expected payoff from the best exercise strategy of the two available at time $t = 0$.

Without a doubt, the software designer is in the position of holding an option to invest, and of having to manage that option carefully. A design decision to commit to restructuring has to be viewed as a decision to exercise that option. Restructure today kills the option, worth 773,

for a payoff of 400, and is therefore clearly suboptimal. Real options theory provides a basis for reasoning about the value of flexibility in software products and projects.

8 A Richer Example

In this section we present a somewhat richer example: there is still just one period in which uncertainty is resolved, but cash flows occur in multiple periods. First we discuss how to formalize multi-period cash flow streams; then we present a numerical example.

8.1 Multiple Period Cash Flow Streams

Extending the previous analyses to cash flow streams in multiple time periods is straightforward. The traditional static NPV is computed as the expected present value S_0 discounted to time 0 of the stream of profits $B_n, n \geq 0$ from the investment, with the cash flow in each period discounted in the standard fashion at a rate compounded by the number of periods:

$$S_0 = \sum_{n=0}^{\infty} \mathbf{E} B_n / R^n. \tag{8}$$

As we noted above, even if the NPV is positive, it might be better to wait to find how the underlying uncertainty is resolved. The optimal approach is to decide at any time k whether the value of expected profits discounted to time k (i.e., S_k) is sufficiently higher than the direct cost L to justify switching. The designer must compare the value of investing now against that of investing at *all* future times. The question is not just whether to invest, but when, if ever.

At any time k, the designer has the right but not the obligation to invest L (the exercise price of the option) to receive a stream of profits (reduced future costs) with an expected present value S_k (the asset price). The random variable S_k is the expected benefit of restructuring at time k , i.e., the value of the future profit stream resulting from investing at time k discounted to time k.

We computed S_0 above. The payoff from exercising at time 0 is $G_0 = (S_0 - L)^+$, since one would not invest if $S_0 < L$. Notice that this is the same as expression (3) for the payoff from an American call option on a stock at time 0. S_0 is analogous to the stock price at time 0—thus our choice of notation.

How do we generalize the expression (8) for S_0 to an arbitrary node v in the tree? To compute the benefit $S(v)$ at node v, we proceed as in expression (8), except that we discount the profits to time corresponding to the depth of node v rather than time 0, and we replace the expectation by the conditional expectation. Finally, we only perform the summation from $\delta(v)$ to ∞. Thus $S(v)$ is given by the following expression:

$$S(v) = \sum_{w : v \rightarrow w} \mathbf{E}\left[B(w) R^{\delta(v)-\delta(w)} \middle| v \right]. \tag{9}$$

Recall that $\delta(w)$ denotes the depth of node w which also corresponds to the time associated with node w. The expected benefit of restructuring at the node v is then

$$G(v) = (S(v) - L)^+, \tag{10}$$

which is the same as expression (3) for the payoff from a call option. The value V_k of this option represents the value of the investment opportunity, which would be lost if we were to exercise at time k. As described in Section 6, V_k can be computed for any k using dynamic programming. Also, we mentioned that it is optimal to exercise the option when the value V_k equals or exceeds the payoff G_k. Thus it is optimal to switch to choice C when $S_k - L \geq V_k$, or

$$S_k \geq L + V_k.$$

Informally, we should exercise our option to restructure when the benefit S_k is at least as much as the sum of the direct cost L and the opportunity cost V_k. What we end up with is a general rule for committing to costly design decisions under uncertainty:

> *If at any time k, S_k, the expected value of future profits discounted to time k is at least V_k more than the direct costs, L, then commit to the design decision, otherwise do not.*

8.2 A Design-For-Maintenance Scenario

To illustrate how to use these tools we consider a problem involving the design of internet agents [29, 31, 44]. Agents are specialized autonomous software entities, e.g., for finding inexpensive airplane tickets or filtering news. It often makes sense for one agent to use another to accomplish its task. For example, a financial portfolio management agent could use another agent to get company reports.

We imagine a system of agents in which each agent has a capability, and in which the capabilities of available agents are stored in a capability directory. Suppose that the agent-based system evolves over time, and that the designer is considering two designs:

D: Distributed directory. A copy of the directory is wired into the code of each agent. An agent that wants to find another agent consults its local directory at essentially zero cost. This approach does not follow the information hiding design criterion. Whenever an agent is added to the system, the directory code in each agent has to be changed. We denote the total cost of the code changes required when an agent is added at time n by the random variable D_n.

C: Centralized Directory. There is one *yellow pages* agent that implements the capability directory. All other agents access directory information through an interface of this agent. We let C denote the total cost of initially hard-coding the centralized directory and its associated interfaces. When a new agent is created, the yellow-page agent needs

to be changed. Also, an agent requiring a capability needs to query the yellow-page agent. We let the random variable C_n denote the total query and update costs at time n.

Which approach should the software engineer choose? If starting from scratch, there is no flexibility to delay deciding. The decision has to be made now and an NPV approach is justified. The basic tradeoff will be increased design and runtime costs of the centralized approach against the reduced future maintenance costs. (Here we ignore other benefits of modularity, including parallel development and improved manageability through abstraction.) The information hiding doctrine stipulates that absent overriding considerations, information hiding structures are better economically.

In order to compare choices C and D, we can assume that the costs that are common to both approaches are 0. The only relevant costs are C and C_n for choice C, and D_n for choice D. In fact we can view the problem as one of deciding whether or not to use choice C, and express the costs and benefits relative to choice D. There are two quantities of interest when choice C is compared with choice D:

- The direct cost of choice C, i.e., the immediate cost of implementing it, which is $L = C$.
- The monthly profit of choice C relative to D in month n for $n \geq 0$, which is $B_n = D_n - C_n$.

We view the software design problem as an investment decision problem: Should L dollars be invested in choice C? Let us assume a discount factor R. Consider the traditional NPV approach to this problem. To apply it, we first compute the expected present value S_0 (discounted to time 0) of the stream of profits $B_n, n \geq 0$ from the investment:

$$S_0 = \sum_{n=0}^{\infty} \mathbf{E}B_n/R^n, \tag{11}$$

which we refer to as the expected benefit of choice C relative to D at time 0. The NPV of the investment at time 0 is

$$NPV = S_0 - L = S_0 - C.$$

The traditional NPV rule states that if the NPV is positive, then the investment should be made, otherwise not. Thus a reasonable decision rule, when there is no design to start with, is the following:

> *If the expected present value of the future profits S_0 that would flow from choice C exceeds the direct cost C of implementing it, then go ahead and implement choice C, otherwise implement choice D.*

As noted by Boehm [7], this kind of rule is often used by software engineers, implicitly or explicitly.

Now suppose, however, there is an existing system with a distributed structure (D), and the designer is deciding whether or not to invest in restructuring to switch to choice C. In

addition to the cost C of creating the yellow pages agent, there might be a cost C^s to scrap the distributed design. Each agent must be changed so that it queries the yellow pages instead of its own local directory. Thus the direct cost of choice C is $L = C + C^s$.

Given these costs, how should the engineer decide? It is tempting to propose the following rule (compare it with the previous rule):

> *If the expected present value of future profits S_0 that would flow from restructuring exceeds the direct cost of restructuring, L, then go ahead and restructure, otherwise do not.*

By now we recognize the flaw in this analysis: The designer has the option to wait in hopes of making a better decision in the future. In addition to the direct cost L, there is an opportunity cost that represents the loss of the value of flexibility. Thus, at any time k, the value of the expected profits discounted to time k (i.e., S_k) must be sufficiently higher than the direct cost L to justify switching. The designer should compare the value of investing now (at time 0) versus investing at *all* possible future times.

8.3 Numerical Calculations

We make the analysis of this decision concrete with a numerical example. Suppose the cost C to restructure is 9000, and the cost C^s of scrapping the distributed directory structure is 1000. Thus the total direct cost L of restructuring is $C + C^s = 10000$.

For simplicity, we assume that building the yellow-page agent and scrapping the distributed directories takes 0 time. Each time step in our model represents 1 month. Time n represents the beginning of the n'th month, for $n = 0, 1, 2, \ldots$. We imagine that during the current month, or month 0, several new agents will be created, and that the associated updating cost under the distributed approach is $D_0 = 2000$. We assume that the total query/update cost under the centralized approach is $C_n = 500$ at all times n. Thus if we move to a centralized directory at the beginning of month 0, the monthly profit for month 0 would be

$$B_0 = D_0 - C_0 = 2000 - 500 = 1500.$$

Suppose an agent development technology is about to be deployed, and that if it succeeds (with probability $p = 0.5$) several new agents will be created each month, starting with month 1. This outcome is favorable for approach C and we will therefore superscript variables under this scenario with the letter f. Suppose that if this event occurs the total maintenance cost under approach D, is $D_n^f = 3000$ for all $n \geq 1$. On the other hand, if the technology fails, few new agents will be created, a situation unfavorable for approach C: the cost to switch to a centralized directory will not be paid back by the cost-savings of the information hiding restructuring. Thus, we superscript variables in this scenario by the letter u. We suppose that the corresponding update cost under choice D in this case is much lower, at $D_n^u = 400$ for all $n \geq 1$.

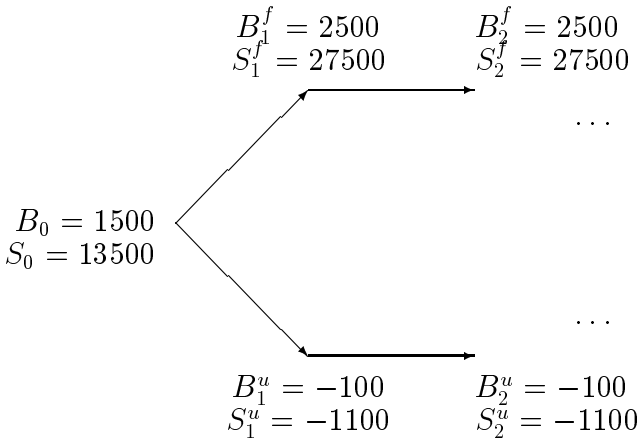

Figure 1: *Event tree for the numerical example*

Thus from month 1 onward, the monthly profit from restructuring for information hiding would be

$$B_n^f = D_n^f - C_n = 3000 - 500 = 2500, \quad n \geq 1$$

in the favorable scenario, and

$$B_n^u = D_n^u - C_n = 400 - 500 = -100, \quad n \geq 1$$

in the unfavorable scenario, each case occurring with probability 0.5. Therefore, for $n \geq 1$,

$$\mathbf{E}B_n = \mathbf{E}B_1 = pB_1^f + (1-p)B_1^u = 0.5(2500 - 100) = 1200.$$

Our model is represented by the event tree in Figure 1. There are just two possible paths in this event tree, which we denote by ω_f (favorable) and ω_u (unfavorable). Note that for $k \geq 1$ and $n \geq k$, on the favorable path,

$$\mathbf{E}(B_n|\omega_f) = B_n^f = B_1^f = 2500,$$

and on the unfavorable path,

$$\mathbf{E}(B_n|\omega_u) = B_n^u = B_1^u = -100.$$

Assuming a monthly discount factor of $R = 1.1$, we calculate the benefit of switching at

time k. For $k = 0$, we use expression (8) to compute

$$\begin{aligned} S_0 &= \sum_{n=0}^{\infty} \mathbf{E}(B_n)/R^n \\ &= B_0 + \sum_{n=1}^{\infty} \mathbf{E}(B_1)/R^n \\ &= B_0 + \mathbf{E}(B_1)/(R-1) \\ &= 1500 + 1200/0.1 \\ &= 1500 + 12000 = 13500. \end{aligned} \tag{12}$$

For $k \geq 1$, from expression (9), the benefit from switching at time k in the favorable scenario (or on the favorable path) is

$$\begin{aligned} S_k^f &= \sum_{n=k}^{\infty} \mathbf{E}[B_n|\omega_f)R^{k-n} \\ &= \sum_{n=k}^{\infty} B_1^f R^{k-n} && (13) \\ &= \sum_{n=0}^{\infty} B_1^f/R^n && (14) \\ &= B_1^f \frac{R}{R-1} && (15) \\ &= 2500(1.1)/0.1 = 27500, \end{aligned}$$

which is bigger than the direct cost $L = 10000$. Thus in the favorable scenario the expected benefit of switching is always greater than the cost. Similarly,

$$S_k^u = B_1^u \frac{R}{R-1} = -100(1.1)/0.1 = -1100, \quad k \geq 1, \tag{16}$$

which is smaller than the direct cost $L = 10000$, so in the unfavorable scenario the expected benefit of switching is smaller than the cost. Note that from the definitions of S_0, S_1^f and S_1^u it follows that

$$S_0 = B_0 + (p/R)(S_1^f + S_1^u). \tag{17}$$

Should the designer invest $L = 10000$ and restructure in order to switch to design C now, or would it be better to wait for a month and invest only if the situation favors a switch? There is no uncertainty after the first month, so these are the only strategies to consider.

We first approach this question by computing the net present value of these two strategies. The NPV of Strategy 1 is

$$NPV(1) = S_0 - L = B_0 + \mathbf{E}(B_1)/(R-1) - L = 13500 - 10000 = 3500. \tag{18}$$

Since the NPV is positive, the NPV rules indicates that the designer should invest. However, let us calculate the net present value of investing under Strategy 2: Wait one month, and invest in switching to the yellow-page agent only if the technology succeeds. Since the technology succeeds only with probability $p = 0.5$, the net present value of Strategy 2 is

$$\begin{aligned} NPV(2) &= (p/R)(S_1^f - L) \\ &= (p/R)\left(B_1^f \frac{R}{R-1} - L\right) = (0.5/1.1)(27500 - 10000) = 7954, \end{aligned} \quad (19)$$

which is significantly greater than the NPV of investing immediately. This clearly shows that it is better to wait.

We now approach the question by computing the value V_k of the investment opportunity at times $k = 0$ and $k = 1$. The payoff if we exercise our option to invest at time k is given by $G_k = (S_k - L)^+$ which is identical to the expression for the payoff from an American call option. Since there is no uncertainty after time 1, it is easy to see that $V_k = G_k = (S_k - L)^+$ for all $k \geq 1$. In particular, if the technology succeeds, the option value at time 1 is

$$V_1^f = (S_1^f - L)^+ = (27500 - 10000)^+ = 17500, \quad (20)$$

and if the technology fails,

$$V_1^u = (S_1^u - L)^+ = (-1100 - 10000)^+ = 0. \quad (21)$$

From the backward recursion (7) we conclude that

$$\begin{aligned} V_0 &= \max\{G_0,\ (1/R)\mathbf{E}(V_1)\} \\ &= \max\{G_0, (1/R)(pV_1^f + (1-p)V_1^u)\} \\ &= \max\left\{(S_0 - L)^+, (p/R)(S_1^f - L)^+\right\} \quad (22) \\ &= \max\left\{(B_0 + \mathbf{E}(B_1)/(R-1) - L)^+,\ (p/R)\left(B_1^f R/(R-1) - L\right)^+\right\} \quad (23) \\ &= \max\{3500, (1/1.1) \times 0.5 \times (17500 + 0)\} \\ &= \max\{3500, 7954\} \\ &= 7954. \quad (24) \end{aligned}$$

Notice that the values 3500 and 7954 in the $\max$ above are exactly the NPVs of strategy 1 and strategy 2, respectively. Also, $V_0 > G_0 = 3500$, so it is not optimal to invest right away. However, after 1 month, if the technology succeeds, $V_1 = G_1 = 17500$, so it is then optimal to invest at that time. Thus we have shown in two different ways that strategy 2 is optimal. In general when the uncertainty lasts several periods, computing NPVs for the exponentially many strategies is impractical. The dynamic programming approach from option pricing theory would be the method of choice.

8.4 The Abstract Algorithm

In this subsection we outline a procedure which a software manager might use to make decisions in the face of uncertainty. Assume that a software manager is interested in making a decision $\mathcal{D}$.

1. Decide on an decision tree and related probabilities. This is the manager's view of the future pertaining to the decision $\mathcal{D}$. Each node in the tree corresponds to a possible state of the world in the future.

2. For each node of the tree decide the value of making decision $\mathcal{D}$ at that node. Intuitively, the value at a node denotes the benefit achieved from making the decision at that node.

3. Decide on an interest rate R. Intuitively, the interest rate R signifies how urgent the decision is to the manager. If R is high, decision $\mathcal{D}$ will tend to happen close to the initial time. For a very high interest rate, the decision will always be made at the root of the event tree.

4. Run the dynamic programming algorithm and find the optimal decision rule for making decision $\mathcal{D}$.

In practice, estimating the parameters of the model will be hard, and will often require expert estimates. On the other hand, the designer has to make a decision. The trustworthiness of the results produced by a model obviously depend on the validity of the inputs. A designer would often explore a set of models, performing sensitivity analysis to help to assess validity. No model can give easy answers to complex design questions. The real options approach is not a silver bullet, but it does provide a technique, based on a well developed theory, that can help us to account for the value of flexibility in software products and processes, a source of value especially important for software.

9 Qualitative Design Principles

A promising aspect of the real options approach is that it both makes key variables explicit in relation to the value of flexibility and provides a basis for reasoning about the effects of changes in their values. A designer who has an intuition for the relationships in the formal theory is perhaps better equipped to reason qualitatively in practice.

In this section explore this issue in the context of the numerical example from the previous section. In particular, we study how the value of the option depends on the cost to invest, uncertainty over benefits, likelihood of a favorable outcome, and uncertainty over cost. In the following subsections, as we vary parameters, we continue to assume that the two scenarios retain their favorable/unfavorable status:

$$S_1^f > L > S_1^u.$$

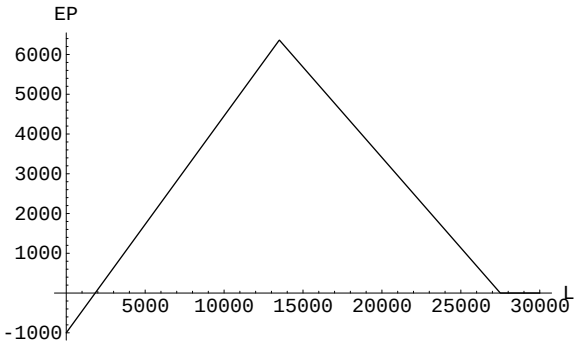

Figure 2: The benefit of waiting, EP, versus the direct cost L.

9.1 Effect of Direct Cost

From expression 22, we notice that the value V_0 of the option is the maximum of two quantities: the NPV of strategy 1, namely $(S_0 - L)^+$, and the NPV of strategy 2, $(p/R)(S_1^f - L)^+$. Note that since $p/R < 1$, as L decreases, the former NPV increases faster than the latter. Thus, if all other parameters remain the same, there is a critical value for the direct cost L below which it is optimal to restructure immediately, i.e., at time 0. Consider the quantity $EP = NPV(2) - NPV(1)$. EP represents the benefit achieved by waiting, or in other words the difference between the value of strategy 2 (where one waits until the next period) versus strategy 1 (where one makes the decision at the initial node). Larger values of EP mean higher benefits of waiting. The graph of EP with respect to the direct cost L is shown in Figure 2.

Another way to state this principle is that if the direct cost L is sufficiently low, the cost of waiting (the profit $S_0 - L$ one would forgo) outweighs the value of waiting (the value V_0 of the flexibility to reverse the decision not to invest). Since there is nothing special about time 0, this statement applies at any time. Thus we have a rigorous, options-theoretical justification for what would otherwise remain an informal software design guideline:

> *If the cost to effect a software design decision is sufficiently low, then the benefit of investing to effect it immediately outweighs the benefit of waiting, so the decision should be made immediately.*

Although this design decision-making rule of thumb seems obvious, it contradicts a heuristic that one of the authors has heard on numerous occasions: Always delay making design

decisions until you are forced to make them because they block progress on all other fronts. The plausible reasoning behind this rule is that you should wait until all possible information is in before investing. The options approach shows that this rule is wrong in general. This conclusion helps to clarify the appealing notion that by reducing costs, new technologies, such as restructuring tools [20], can toggle a situation from one in which it is best to delay to one in which it is optimal to invest immediately.

9.2 Effect of Uncertainty over Benefits B_n

In the numerical example of the previous section, the two possible values of B_n for $n \geq 1$ were $B_n^f = 2500$ in the favorable case and $B_n^u = -100$ in the unfavorable case. Now suppose we keep all parameters the same, except that we change B_n^f to 3000 and change B_n^u to -600. Notice in particular that the expectation of B_n,

$$\mathbf{E}(B_n) = 0.5 \times (3000 - 600) = 1200, \quad n \geq 1,$$

is the same as before, but that the *variance* of B_n is larger. This new parameterization models greater uncertainty about the rage of future benefits without any change in the net expected benefit, i.e., a "higher risk, higher return" project.

Since the expectation remains the same, the NPV of Strategy 1, ("restructure at time 0"), given by expression (18), is the same as before, because (see expression 12) the expected benefit S_0 of switching at time 0 depends only on the expectation of each B_n. On the other hand, if the designer waits for 1 month and switches only if the situation is favorable (Strategy 2), the net benefit S_k^f (see expression 15) is

$$S_k^f = B_1^f R/(R-1) = 3000 \times 1.1/0.1 = 33000, \quad k \geq 1,$$

which is bigger than the previous S_k^f value of 27500. Thus the NPV of Strategy 2 (see expression (19)) is bigger than before.

This shows that the incentive to delay the decision to invest in restructuring increases with project risk, manifested as uncertainty over future benefits B_n, as long as all else, notably the expected benefit, stays the same. Conversely, as uncertainty about future value diminishes, it becomes clearer whether it would pay to invest. The option value preserved by delaying diminishes. In the limiting case, you can decide immediately based on the NPV.

Consider again the quantity $EP = NPV(2) - NPV(1)$. The graph of EP against B_n^f (such that $E(B_n) = 1200$ or $B_n^u = 2400 - B_n^f$) is shown in figure 3. The options formulation makes the issue clear. The expected payoff of restructuring immediately is the same as before, since the values $\mathbf{E}(B_n)$ are the same. However, if restructuring is delayed, then one of two outcomes occurs. In the unfavorable case (see (21)) the payoff V_1^u is still zero, because the design option will not be exercised. However, in the favorable case, the payoff V_1^f (see (20)) is greater than before. Thus the option value V_0 given by (22) increases.

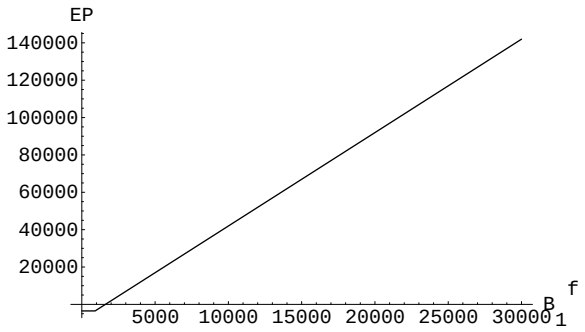

Figure 3: The benefit of waiting, EP, versus B_1^f, given that $\mathbf{E}B_1$ remains fixed at 1200.

These conclusions makes sense intuitively. Uncertainty over the value of investing in a user's manual, for example creates an incentive to wait for information. If a manual is very likely to be profitable, there is less benefit to waiting. Similarly, if its value is clearly minimal or negative, a decision not to invest can be made immediately. Thus we can conclude with the following qualitative design guideline.

> *With other factors, including the NPV, remaining the same, the incentive to wait for better information before effecting a design decision increases with risk—the spread in possible benefits.*

9.3 Effect of the Probability of a Favorable Outcome

In the example of the previous section, we assumed that at time 0 the likelihoods of favorable and unfavorable outcomes were equal, with $p = 0.5$. This probability distribution represents the risk that the favorable outcome will not be actualized. We now examine how the value V_0 of the real option depends on that probability p of a favorable outcome.

Consider the payoff $G_0 = (S_0 - L)$ from immediate exercise, i.e., the NPV of strategy 1 (see expression (17)):

$$G_0 = B_0 + (pS_1^f + (1-p)S_1^u)/R - L = (p/R)(S_1^f - S_1^u) + B_0 - S_1^u/R - L.$$

If we plot G_0 against p the slope would be $(S_1^f - S_1^u)/R$. The discounted expected value of the option V_1 is thus (expression 22)

$$\mathbf{E}V_1/R = (p/R)(S_1^f - L).$$

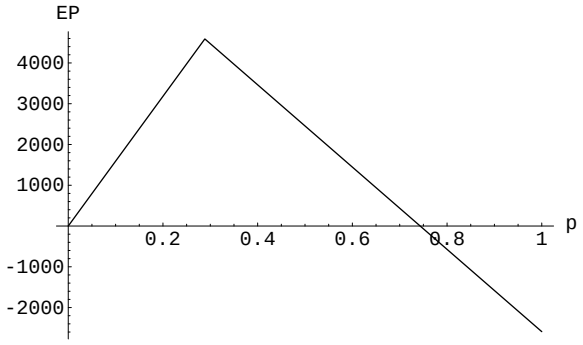

Figure 4: The benefit of waiting, EP, versus the probability p of a favorable outcome p.

This is the NPV of strategy 2. If we plot this value against p, we find the slope to be $(S_1^f - L)/R$. Since we have assumed $S_1^u < L$, we see that as we increase p, the NPV of strategy 1 grows faster than that of strategy 2. Thus as the probability p of a favorable outcome increases, at some point the strategy of investing right away becomes optimal. To put it differently, as the risk of a unfavorable future decreases, so does the incentive to wait. The graph of $EP = NPV(2) - NPV(1)$ versus p is shown in Figure 4. We have thus given a rigorous basis for the following design decision-making heuristic:

> *The incentive to wait before investing varies with the likelihood of unfavorable future events occurring.*

Notice that this decision-making heuristic addresses uncertainty and risk in a different way than the previous rule. The previous rule addresses the variance in the payoffs under different outcomes. This rule addresses variation in the uncertainty about the likelihoods of future events that influence outcomes. We have thus identified two important and orthogonal dimensions of risk and have presented heuristics with rigorous theoretical underpinnings for reasoning about and responding to them.

9.4 Effect of Uncertainty over Direct Cost

We now examine the possibility of the cost L being uncertain. This aspect of uncertainty is critical in software engineering, especially if delaying design decisions is an accepted strategy: It goes to the question of estimating project costs, which is widely known to be subject to

significant uncertainties. Uncertainty about costs might reflect uncertainty about availability of skilled labor in the future, or about changes in technology, such as the development of automated restructuring tools [20] that could significantly reduce costs.

To simplify matters, let us assume that the monthly profit B_n from restructuring is 1500 at all times $n \geq 0$ (there is no uncertainty in this regard). Thus the expected benefit of switching at time k, for any $k \geq 0$, is given by an expression analogous to expression (15) (in either scenario):

$$S_k = B_k \frac{R}{R-1} = 1500(1.1)/(0.1) = 16500, \quad k \geq 0.$$

However, now assume that the direct cost L_0 at time 0 is known to be 10000, but that it is uncertain at time 1. Let us assume that L_1 is either $L_1^f = 5000$ (a favorable situation) or $L_1^u = 20000$ (an unfavorable situation). The NPV of strategy 1, investing now, is

$$NPV(1) = S_0 - L_0 = 16500 - 10000 = 6500,$$

which is positive. The traditional NPV rule suggests switching right away. Again, this rule is faulty because it ignores the contingent, option strategy: Wait a month, and switch only if the direct cost is $L_1^f = 5000$. The NPV of this strategy is

$$NPV(2) = (p/R)(S_1 - L_1^f) = (0.5/1.1)(16500 - 5000) = 11500,$$

which considerably greater than the NPV of the first strategy. Thus it is optimal to wait a month in this case before deciding whether to invest.

Now let us go a step further, and see what happens if we keep L_0 the same and increase the uncertainty (in particular, the variance) of L_1, while keeping its expectation $\mathbf{E}L_1$ the same. This would mean L_1^f is smaller, and $NPV(2)$ larger. In this case, the value of waiting is even greater. This situation is analogous to the one in Subsection 9.2. When the uncertainty over direct costs is larger, and the expectation remains the same, the potential profit in the favorable scenario increases, while in the unfavorable scenario it remains the same at 0. Thus we have provided a rigorous theoretical justification for another heuristic:

> *All else being equal, the value of the option to delay increases with variance in future costs.*

10 An Options Interpretation of Information Hiding

In this and the next section we discuss informally how the options perspective can give us insights into key issues in software design other than timing. In this section we discuss the related work of Baldwin and Clark which seeks to explain the value of modularity with real options. In the next section, we present an informal analysis of Boehm's spiral model in real options terms.

Baldwin and Clark appeal to real options theory to explain the value of modularity, with specific reference to information hiding [1, 2, 3, 4]. Their idea is that a module creates an option to select the best implementation from a set of implementations for a given slot in a modular architecture, where variants are produced by investments in module-level "experimentation." Their aim is to develop a theory of the evolution of the computer industry, from highly centralized to decentralized. In a nutshell, they see firms as being driven by the options value of modularity to modularize designs, but that modularity also gives competitors opportunities to specialize around particular slots in the modular architecture (e.g., the disk drive).

The interesting point is that real options have already been used to explain the value of modularity to some degree. To put it simply, we understand an information hiding module as creating an option for the designer to change the hidden secret of the module if doing so becomes valuable in the future. The investment in the design of the module has an immediate benefit that can be expressed in part as the value of such a design option. More generally, modularizing a system involves the selection of a portfolio of such options. The choice should be made so as to maximize the value of that portfolio.

Of course modularity has cognitive and managerial benefits unrelated to options to change, and the options perspective has nothing to say about such benefits. The information hiding concept of modularity, however, focuses directly on encapsulation of design decisions that are "likely to change [34]." Taking the options perspective suggests ways to broaden the traditional view of information hiding. To give one example, viewed as a bet on future conditions, it might make sense to hide a decision that is unlikely to change, but where the risk is high in the sense that the event would have a high cost were it to occur in the absence of modularity. We can think about using information hiding to hedge risks of unlikely change.

Baldwin and Clark's model assumes that a given system has already selected a module implementation for each slot and that the various alternative implementations have values that are normally distributed around that of the currently selected implementation, with a variance determined by the number of design parameters that are bound within the module. They model the value of a system as a sum of the values of the modules in the system. They take value maximization as the goal of design and derive design investment policies under which such maximization is achieved. In particular, they seek optimal levels of investment in the testing and experimentation activities meant to produce and evaluate variant module implementations. They conclude that testability of modules in isolation is critical, and that firms generally under-invest in creating variant module implementations.

The work of Baldwin and Clark is encouraging. It substantiates our insight that options thinking can help to model, analyze and explain basic issues in software design, including information hiding. Their work does leave many questions unanswered, which we plan to address in future work. First, although they draw heavily from software design, appealing to information hiding and structured design, they do not try to develop an investment or options-based view of basic software design principles. Second, their model of the value of a system

is simple. It might be a reasonable model of some systems that are essentially collections of largely independent features. Some shrink-wrapped software packages have roughly this structure [12]. However, the model does not capture other systems well, in which the value of the system is not a simple sum of module values. Nor are software module implementations generated by a random walk around the current module, as reflected in the Gaussian assumption of Baldwin and Clarke. In future work, we plan to investigate the ways in which real options in real software systems interact, and to seek more convincing models of the options that modules create in software systems.

11 An Options Interpretation of The Spiral Model

The spiral model is generally regarded as a risk-based process model. We see it instead as a value-based model, in which the consideration of risk puts a premium on flexibility. It is flexibility in the face of uncertainty engineered into and encouraged by the spiral process that makes such a processes more valuable than inflexible processes, such as monolithic waterfalls.

Key elements of the model have a clear interpretation in real options terms. To the extent that we can analyze it in this way, we gain a valuable new way of understanding this process model. In this section we analyze three aspects of the model. First, it can be read as an investment-oriented, value-maximizing model. Second, developing variant approaches in each phase can be seen as creating an option at each stage—to pick the best of n approaches identified. Third its phased project structure can be seen as adding value in the form of compound real options: Each phase creates an option for the designer to invest in the next one.

11.1 The Spiral Model as an Investment Model

In describing the model Boehm states that the designer begins with a hypothesis that undertaking a project will improve the operational mission of an enterprise. The investment interpretation is that the designer begins with a hypothesis that investing will add value. This formulation is consistent with Boehm's description and has the added virtue of making the model consistent with standard corporate finance theory—invest if only if it maximizes the value added to the firm. Similarly, the designer exits a spiral when the value hypothesis is rejected. The spiral model is thus an informal model for investing under uncertainty.

11.2 Value of Flexibility versus Value of Information

Boehm emphasizes that in software design it sometimes pays to invest a little now to resolve uncertainties that might affect future outcomes. Thus, prototyping emerges as a key tactic for managing uncertainty in spiral development processes. Although prototyping seeks value maximization under uncertainty, it is perhaps not best seen in options terms. The problem is that to prototype is to resolve uncertainty by investing. Options, on the other hand, assume that

uncertainty is resolved by waiting for exogenous information. You cannot learn how much a stock is going to be worth tomorrow by investing a little in it today; you learn by waiting to see what the market decides. By contrast, when a designer invests in a prototype to resolve a risk, e.g., in interface requirements, it is investing itself that resolves the uncertainty.

The essence of prototyping is thus not in the value of flexibility (VOF), but rather in what in decision analysis and systems engineering is called value of information (VOI). The idea is that an investment in a prototype is justified when the information it reveals is worth more to the designer than its costs—e.g., when a small investment averts a costly commitment to an unworkable design. Boehm formalizes the concept in terms of statistical decision theory [21]. Hillier and Liebermann provide an introduction and references [22].

The distinction is important, suggesting novel approaches to different kinds of risk. While VOI drives the designer to invest early in risky areas (uncertainty is resolved by investing), VOF drives one to delay investing. Delaying is dual in a sense to prototyping. The reason, again, is that option values increase with risk by providing exposure to "up-side potential" with no down-side, risk thereby creating a disincentive to exercising them early.

We make the point concrete with three examples. First, waiting to see if a proposed standard is widely adopted before investing in a project that depends on it is a real-options-based approach. Second, when design is highly decentralized, e.g., in component-based development with commercial off-the-shelf (COTS) parts, uncertainties about component function and quality can be resolved by waiting to see what suppliers do. Third, an options view can be used to reason about an issue that greatly troubles many software engineering researchers: the rush to market with flawed products not engineered to the highest quality standards.

One situation leading to tradeoffs of quality for time to market is the threat of competitive entry. Exogenous information is critical, making this a VOF issue. The threat can be understood in options terms. Competitive entry reduces market share, and thus the value of the asset (cash flow stream from market share) that the designer gets by shipping a product. Such a discrete drop in asset value can be seen as a dividend—similar to the dividend that a stock pays, with an immediate, corresponding drop in share value.

The informal design rule is "be the first to market." Options theory suggests a principled explanation having to do with dividends. It is a theorem that it is optimal to delay exercising an American call option on a stock paying no dividends until expiration. There is no opportunity cost to delaying, and you know most about the value of the stock just before your right to buy it expires. However, if a stock pays a dividend, the share value drops and the option owner is not entitled to the dividend. The payoff from exercising after a dividend is thus reduced; and so it can be optimal to exericise early to capture value that would be lost otherwise.

Now consider a software designer developing a product under threat of competitive entry. The designer has an option to ship the product, i.e., to exercise an option to capture a share of the market. The entry of a competitor, reducing that share, is a dividend [48]. To capture that benefit the designer might wan to exercise the option to ship early. More generally, dividends create a countervailing incentive: not to delay, but to exercise early to capture benefits that

flow from actually owning an asset.

We see why trade-offs of quality for time to market can be optimal. The ultimate objective of design is value added. More importantly, having options concepts at hand can help us to interpret some otherwise seemingly ad hoc practices. For example, why does Microsoft use a development process that emphasizes the ability to ship a product at any time [12]? We do not know their reasoning, but an explanation in options terms is plausible. Investing to keep products shippable creates an option to ship at any time. In the face of uncertain threats of competitive entry, such an option has value. In essence, by investing in process they purchase an American option, which can be exercised at any time, rather than (say) a European option, which can be exercised only at expiration.

11.3 Generating Alternatives as Generating an Option

A key element of the spiral model is the generation of alternative solution approaches at the beginning of each project phase. We can offer a qualitative discussion of this activity in terms of real options. The insight is that generating alternatives in this context is related to the generation of alternative module implementations in Baldwin and Clark's real options analysis of the value of information hiding. Generating of a set of n solutions provides the designer with the option to select the best one. The options view suggests qualitative guidelines for deciding how much to invest in the "experimentation" that leads to new alternatives: When the risk is high, the option is worth more, so more can be spent on the option. The statement that the level of investment in producing alternatives should be linked to risk appears to be novel. We have derived this guideline from principles and have not yet tested it empirically.

11.4 Phased Projects as Compound Options

Phased project structures are an essential part of the spiral model. The goal is to keep investments small until risks are resolved. Such a structure provides an ability to manage risk by embedding options in a project to change course or even to abandon it as uncertainties are resolved over time. One way to think about phased projects that they contains embedded options to scale back or to abandon the project if conditions turn out to be unfavorable [48]. Such options are *put* as opposed to a *call* options, giving holders the right to sell assets at a specified price, e.g., to abandon a project in return for the market value of its assets). Embedding such options in the project increases its value relative to one without options.

A second interpretation is that investing in one phase of a project buys the designer an option to make a follow-on investment should the situation be favorable at that time. A designer considering such initial investment thus has an option on an option, called a compound option. Similarly, a second phase can yield an option on the third, and so on. Real options theory has been applied to value such phased projects as compound options [19].

A striking insight emerges from the options view: it can be optimal to invest in the first phase of a project even though the overall project has a negative *static* NPV [9]. The key idea

is that spending a little on one phase can yield an option to invest in the next if conditions turn out to be favorable. If the upside potential of the second phase is high, the option value today can outweigh its cost, even if the entire project currently has a negative NPV.

Extending this insight to the spiral model we infer that it can be optimal to begin a spiral not on the hypothesis that the project *will* improve the operational mission of the firm, but that it *might*. If investing in a first phase creates an option to invest in a potentially lucrative second phase, it can be optimal to invest in the first phase, even if the static NPV of the overall project is negative. In other words, we infer that it can pay to begin a spiral process as a *strategic bet* on favorable future conditions. It might be worthwhile to invest in learning a standard even before it is widely adopted, if doing so makes it possible for the designer to respond quickly if the standard does become widely adopted, for example.

11.5 A Concrete Example

We make the preceding point concrete with a simple example. Suppose that the designer can restructure in two phases. The first phase costs 1000. With probability 0.5, that is enough; but with probability 0.5 serious difficulties will be encountered, and 3000 more dollars will have to be invested to produce a satisfactory design. Perhaps the selected restructuring tool has unforeseen shortcomings that require some manual restructuring. More generally, a project might contain risks that are resolved by investing in the first phase.

Suppose that the profit from restructuring, at each time step t, is 200. Summing up the sequence, the present value of this stream, at any time, with the discount rate at 10%, turns out to be $200R/(R-1) = 2200$. NPV analysis compares the expected cost, $1000 + (0.5)(3000) = 2500$, with the present value of profits, 2200. Owing to the risk, the static NPV is -300. Not investing appears to be advised: the project does not improve the mission of the firm.

However, this analysis ignores the value of the option to cancel the project after the first phase if a second phase costing 3000 turns out to be necessary. The traditional NPV analysis ignores the contingent strategy enabled by the embedded option to cancel the project. The dynamic NPV is $(0.5)(2200) - 1000 = 100$. In other words, for 1000 the designer can buy an asset—an option on the second phase—that with even odds will be worth either 2200 or 0. It's a good bet, with an expected value of 100.

Traditional software engineering economics using static NPV overlooks the value of the options—i.e., flexibility—embedded in development projects and products. The options approach is superior to static NPV analysis for quantitative reasoning about projects and assets with embedded options. More importantly for our purposes in this paper, it provides a framework for understanding strategic design, by which value is created with intelligent bets on uncertain future outcomes.

12 Related Work

In this section we address related work in several areas: flexibility in software; options; software engineering economics; software reuse investment analysis; and real options in the analysis of information technology projects. In the following section, we discuss theoretical difficulties in taking an options approach to computer-related issues.

12.1 Flexibility in Software Product and Process Design

We are obviously not the first to notice that flexibility is critical in software design and that it has both costs and benefits that have to be weighed against each other. Information hiding [36], extension and contraction [37] and program family concepts [35] were seminal. More recently, Fayad and Cline [17], emphasize that flexibility has costs, that it should be designed in where it is economically the most effective. They see *design patterns* [15] as providing "hinges" needed for flexibility in particular dimensions. They even state that such hinges provide "opportunities" to make changes that might be necessary—reflecting an implicit options mode of thinking. However, to the best of our knowledge there have been few attempts to connect such concepts explicitly to precise notions of the value added by flexibility.

Of course, concern for value added and appeal to investment analogies in software design is not new. On the other hand, investment concepts are not yet canonical in software design. We suspect that one reason is that standard financial concepts of static NPV on which the best work is based is inadequate for modeling the value of flexibility, and that the best work thus overlooks a critical source of value at the core of modern software design. To the best of our knowledge, we were the first to use contemporary investment theories to explain important software design concepts [43]. (Baldwin and Clark relate real options to modular structures in their work on the evolution of the firm structure of the computer industry.)

12.2 Financial Options Theory

Real options theory is based on financial options theory. The options literature is immense. It is not feasible to cite all relevant work. Financial options have been studied since 1900; however the seminal modern results, which provided long-sought closed-form mathematical formulations for valuing financial options, are due to Scholes, Merton and Black [36,7]. Scholes and Merton received the 1997 Nobel Prize in economics for their work on this topic. Many other results, which are now elements of basic finance, have been produced since [10, 14, 15, 35]. For the last twenty years, researchers have been building the still emerging real options theory [10, 16, 65, 66]. The literature on real options in economics, management, and flexible manufacturing is well developed.

12.3 Software Engineering Economics

The idea of finance-based decision criteria for software design is not new. Boehm wrote on it in his Software Engineering Economics [8]. He emphasized static net present value as the basic concept of value and how it can be taken as the objective of software design.

It bears mentioning that Boehm warns of the pitfalls of taking narrow financial criteria for software design literally: focusing exclusively on quantifiable economics can compromise attention to the human issues that in the long run are critical to success. The human element is critical. Our work is meant to provide intellectual tools that can help designers to think more effectively about key design concepts and problems, not as a simplistic, narrowly economic doctrine.

Boehm's emphasis on static NPV is understandable. Static NPV is the standard measure of value added taught in business schools. Moreover, real options were hardly known when Boehm's book was published. The book mentions options colloquially, but it is does not attempt to analyze core software design concepts in terms of real or financial options.

12.4 Software Reuse Investment Analysis

The software reuse literature reports applications of finance theories to investment analysis [26]. Favaro has argued that much work in this area uses techniques known to have technical problems. He has argued NPV is the proper basis for such analysis [16]. That view is consistent with the traditional academic view, but it does not consider the value of flexibility.

Withey, of the Software Engineering Institute, considers flexibility in options terms [49] in work on investment analysis of product line architectures. He shows how economies of scope achieved through investments in product-line-oriented design can be interpreted as options on uncertain markets. Withey declined to invoke real options in his work for the Department of Defense because of skepticism about its broader validity for analyzing software design decisions (personal communication). We agree with Withey that it is critical not to take real options or other such investment concepts as silver bullets for software design decision-making.

12.5 Henderson, Kalutilaka, et al.

Henderson and Kulatilaka are developing an approach to investment analysis of information technology (IT) projects based on real options [23,28]. The proposed approach appears to be a standard use of real options for investment analysis at the project level. The work does not address software design. Flatto reports on a survey of whether managers use real option concepts in IT investment analysis, which showed that options concepts are not used [19].

13 Theoretical Difficulties

We note that in arbitrage-based approaches to financial options pricing (such as the approach embodied in the Black and Scholes equation), the "fair value", or fair trading price, of an option is defined by assuming the absence of arbitrage in a richly traded market. The value of the option is the value at which it can be traded in such a market without creating opportunities for riskless profit. More specifically, the value is defined as the value of a dynamically updated portfolio of assets traded in the market that replicates the random fluctuation of the option payoff G_k. Because the assets are traded, prices for them and thus for the portfolio are available. Because the option is traded and because it has the same payoff as the portfolio, the absence of arbitrage opportunities requires that the option have the same price as the portfolio. In this way, the price of the option is obtained in theory from the prices of assets already in the market. The option is said to be in the span of the market.

It turns out that this arbitrage-free fair value of an American call option is defined as V_k above, but under a synthetic probability measure called the risk-neutral measure and under the risk-free rate of return (i.e., the return on a risk-free bond) as a discount rate. This probability measure is in general not the real measure (e.g., the 0.5 in most of our examples), but is independent of it and based on the volatility (or spread in future uncertain value) of the asset.

However in the case of some real options—perhaps many in the case of software design—the underlying asset is not traded. In other cases, the asset exists only as a result of exercising the option and is not traded independently. Furthermore, real options themselves are not traded, although their values can be reflected in the prices of traded assets, e.g., in the stock prices of firms that hold them. Thus, the use of arbitrage pricing approaches is of questionable validity for valuing some real options, because basic assumptions underlying that pricing theory—that assets and options are traded in a rich market and that the option value are in the span of the other assets—do not always hold.

The basic question to ask is this: Is the payoff on the real option closely correlated with the payoff on an asset or set of assets (e.g., a stock) that are already priced in the market? If so, then pricing techniques based on arbitrage constructions can be used. If not, then other pricing techniques are indicated. In this paper, we employed other techniques. Namely, we use the real probability measures on our decision trees, and we use and assumed discount rate that is not the risk-free rate that would be used in risk-neutral valuation.

Strassmann has expressed reservations about Henderson's use of options in IT. He objects that in richly traded markets there is information on values and uncertainty. However, he notes that computer projects occur in low volumes, and he argues that therefore getting valid data, treating them consistently, and dealing with non-quantifiable effects makes IT project valuation different. Strassman then concludes that "... there is no true resemblance between trading in financial options and planning computer projects [41, p. 159]."

Strassman essentially dismisses the use of real options approaches to valuing options in "incomplete markets," i.e., where option prices are not in the span of the market. Strassman is both right and wrong. He is correct that the use of arbitrage-based pricing techniques

is not theoretically sanctioned. He is also correct that the problem of getting valid data is hard. However, his conclusion is somewhat too pessimistic. Dixit and Pindyck state that, "whether or not spanning holds, we can obtain a solution to the investment problem, but without spanning, the solution will be subject to an assumed discount rate [13, p. 152]." Furthermore, the discount rate is not subject to empirical justification.

In essence, when markets are incomplete, the data required for the pricing based on dynamic programming will be subjective estimates of expert analysts. The pessimist will claim that we are back to where we started, with design reasoning being based on no more than expert opinion that is not subject to dispute on empirical grounds. That is not true. Software designers do estimate the "likelihood of change" and other such probabilities, in any case. The options approach to reasoning about design principles helps us to understand them in a deeper way, because it gives us insight into what the whole range of parameters is in the investment problem, and it tells us how they are related to overall value. In other words, the options approach gives us a well developed and quite sophisticated intellectual framework within which the reason about the implications of our estimates.

Second, in a forward-looking dimension, the identification of the parameters of importance suggests that we begin to collect data relevant to better estimating the data required for options-based valuation techniques. Perhaps most of all, at this time, the options framework gives us a powerful new language in which to think about and discuss key issues in software design, in theory and in practical situations.

Third, the options approach provides a suggestive intellectual basis on which to develop new hypotheses about better software development techniques. One of the most intriguing is that we begin to manage software projects as dynamically changing, actively managed portfolios of assets including real options. The deck of cards provides a visual metaphor, which we juxtapose with Boehm's spiral. As uncertainty is resolved over time, investment decisions are made on the basis of the real options held at the time: cards are bought, monitored and played in a dynamic game against nature and against adversaries. We have yet to make this idea precise and to evaluate its potential to help in practice.

14 Conclusion

Existing concepts are very effective and largely correct. The problem that we identify in this paper is that current doctrine, cast in terms such as information hiding, delaying of design decision, and spiral process models, is ad hoc, idiosyncratic and not explained or unified by any real theory or mathematical models. This makes the ideas hard to teach and to learn, hard to connect to the decision-making criteria that enterprises use in deciding how best to invest scarce resources, and unnecessarily hard to use most effectively in practice. What is the underlying common rationality that ties together such apparently unrelated concepts as information hiding, delaying design decisions, and spiral processes?

In this paper we propose that one answer can be found in the view of software design

as an investment activity. Viewing design this way immediately suggests that current design concepts might be serving as (perhaps imperfect) proxies for explicit policies for investing under uncertainty. If that is the case, then we should be able to analyze existing policies in investment terms. Doing this might reveal both the deeper structures underlying current doctrines, as well as lead the way to refinements and improvements in such doctrines.

We have taken this approach in this paper. The basic claim that we have made and defended on theoretical (not empirical) grounds is that a diverse range of existing design concepts can be analyzed in terms of a particular investment concept, that of real options. We believe that investment theory more promises to help us to build more solid foundations for software design concepts. We have appealed to real options theory in particular to try to explain concepts related to an especially critical issue in software product and process design: the value of flexibility to optimize for value added under uncertainty.

Uncertainty is especially pervasive in software design, for reasons familiar to any designer: complexity, the evolution of the external fields determining both costs and what is valuable, competition, and so forth. In the face of uncertainty, there is a demanding premium on flexibility, because flexibility gives the designer the fundamental means to maximize value dynamically as uncertainty is resolved over time. For this reason we have focused on an investment theory that aims in particular to model the value of flexibility in the face of uncertainty, namely real options theory.

This perspective appears to be useful for helping to explain and to understand a diverse range of issues in software design, including timing of design commitments, information hiding and spiral process models, as well as such issues as rushing of imperfect products to market and Microsoft's always-ready-to-ship product development model.

Moreover, the options view suggests that in some cases, the accepted wisdom admits some improvement. For example, we argue on theoretical grounds that it can be optimal to begin a spiral early even if the project viewed as a whole does not promise to improve the operational mission of the enterprise (i.e., to add value). We also provide an argument against simplistic timing heurisms, such as always delay a design decision until it blocks all progress. This heurism is valid in the absence of dividend phenomena, but is obviously no valid. The options interpretation provides a cogent explanation about tradeoffs between opportunity costs of committing to and delaying investments.

One of the most valuable kinds of results in the mathematical field is the establishment of a bridge between previously disconnected areas of reasoning. Such a bridge allows the structures and insights from one field to be exploited in another. We have tried to construct such a bridge in this paper, between the very informal realm of contemporary software design thinking and the formal realm of modern finance, with a particular focus on options valuation concepts.

On theoretical grounds alone in this paper we have argued that this bridge provides intellectual leverage. Much future work remains to be done. Drawing connections to other investment subdisciplines, such as capital budgeting, is an obvious next step.

Another dimension to explore is trying to explain other troubling issues in software engineering, with the objective of deriving new approaches to handling them. One area of immense uncertainty, for example, is cost estimation. Perhaps estimates are inherently uncertain owing to the many random events that interact in real development processes. In this case, rather than trying for more accurate estimation models, perhaps one should try to structure projects to optimize their value in the face of such uncertainty. For example, by using only standard hardware and software that can be sold on the market, one might maximizing the value of an embedded option to abandon should early cost estimates prove too optimistic.

A third direction for future work is experimental research in which the investment view of software design is operationalized, with tool and management support. We are considering in particular developing a process model based on the view of projects are portfolios of assets that include real options. Such projects must be managed actively as investments over time. Resources must be invested in both producing assets and options in such a way as to optimize for the value added at each step. Software development comes to be seen as active investment management (AIM) as uncertainties are resolved over time. Using such an approach in real-world development and assessing its strengths and weaknesses as well as practical impediments to using it is an important topic for future work.

Acknowledgements. This work was supported in part by the National Science Foundation (NSF) under grant numbers CCR-9502029 and CCR-9506779, and by the Defense Advanced Research Projects Agency and Rome Air Labs under grant number F30603-96-1-0314.

References

[1] C. Baldwin and K. Clark. Modularity-in-design: An analysis based on the theory of real options. Technical Report Working Paper, Harvard Business School, 1992, revised 1997.

[2] C. Baldwin and K. Clark. Design options and design evolution. Technical Report Working Paper 97-038, Harvard Business School, 1997.

[3] C. Baldwin and K. Clark. The microstructure of designs. Technical Report Working Paper 97-031, Harvard Business School, 1997.

[4] C. Baldwin and K. Clark. Value, contract structure and organizations. Technical Report Working Paper 97-085, Harvard Business School, 1997.

[5] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–655, May/June 1973.

[6] B. W. Boehm. *Software Engineering Economics*. Advances in Computing Science and Technology. Prentice Hall, 1981.

[7] B. W. Boehm. Software engineering economics. *Transactions on Software Engineering (SE)*, 10(1), Jan 1984.

[8] B. W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–73, May 1988.

[9] R. Brealey and S. Myers. *Principles of Corporate Finance*. (New York: McGraw-Hill), fifth edition edition, 1996.

[10] M. Brennan and E. Schwartz. Evaluating natural resource investments. *Journal of Business*, 58(2):135–157, 1985.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *The Design and Analysis of Computer Algorithms*. Cambridge, Mass.: MIT Press, 1990.

[12] M. Cusumano and R. Selby. *Microsoft Secrets*. (New York: Free Press), 1995.

[13] A. Dixit and R. Pindyck. *Investment under uncertainty*. Princeton Univ. Press, 1994.

[14] A. Dixit and R. Pindyck. The options approach to capital invesment. *Harvard Business Review*, pages 105–115, May-June 1995.

[15] R. J. E. Gamma, R. Helm and J. Vlissides. *Design Patterns*. (Reading, Massachusetts: Addison-Wesley), 1995.

[16] J. Favaro. A comparison of approaches to reuse investment analysis. In *Proceedings of the Fourth International Conference on Software Reuse*, 4 1996.

[17] M. Fayad and M. Cline. Aspects of software adaptability. *Communications of the ACM*, 39(10):58–59, October 1996.

[18] J. Flatto. *The application of real options to the information technology valuation process: A benchmark study*. PhD thesis, Univ. of New Haven, 1996.

[19] R. Geske. The valuation of compound options. *Journal of Financial Economics*, 7:63–81, 1979.

[20] W. Griswold and D. Notkin. Automated assistance for program restructuring. *Transactions on Software Engineering and Methodology*, 2(3), July 1993.

[21] A. Habermann, L. Flon, and L. Cooprider. Modularization and hierarchy in a family of operating systems. *Comm. of the ACM*, 19(5):266–272, May 1976.

[22] F. Hillier and G. Liebermann. *Introduction to Operations Research*. McGraw Hill, 6th edition edition, 1995.

[23] J. Hull. *Options, Futures, and Other Derivative Securities*. Prentice Hall, 2nd edition, 1993.

[24] C. Kester. Today's options for tomorrow's growth. *Harvard Business Review*, pages 153–160, March/April 1984.

[25] N. Kulatilaka and A. Marcus. Project valuation under uncertainty: when does dcf fail? *Journal of Applied Corporate Finance*.

[26] W. Lim. Reuse economics: A comparison of seventeen models and directions for future research. In *Fourth International Conference on Software Reuse*, 1996.

[27] D. Luenberger. *Investment Science*. (New York: Oxford University Press), 1998.

[28] S. Madj and R. Pindyck. Time to build, option value, and investment decisions. *Journal of Financial Economics*, 18(1):7–27, 1997.

[29] P. Maes. Agents that reduce work and information overload. *Comm. of the ACM*, 37(7), July 1994.

[30] R. McDonald and D. Siegel. The value of waiting to invest. *Quarterly Journal of Economics*, 101(4):707–727, 1986.

[31] T. Mitchell, R. Caruana, D. Freitag, and J. McDermott. Experience with a learning personal assistant. *Comm. of the ACM*, 37(7), July 1994.

[32] S. Myers. Determinants of corporate borrowing. *Journal of Financial Economics*, 5:147–175, 1977.

[33] S. Myers and S. Majd. Abandonment value and project life. *Advances in Futures and Options Research*, 4:1–21, 1990.

[34] D. Parnas. The modular structure of complex systems. Technical report.

[35] D. Parnas. Program families.

[36] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the Association of Computing Machinery*, 15(12):1053–1058, Dec. 1972.

[37] D. L. Parnas. Designing software for ease of extension and contraction. In *Proceedings of the Third International Conference on Software Engineering*, pages 264–277. IEEE, 1978.

[38] R. Pindyck. Irreversible investment, capacity choice, and the value of the firm. *American Economic Review*, 78(5):969–985, 1988.

[39] S. Ross. Uses, abuses, and alternatives to the net-present-value rule. *Financial Management*, 24(3):96–102, Autumn 1995.

[40] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. (Upper Saddle River, NJ: Prentice-Hall), 1996.

[41] P. Strassmann. *The Squandered Computer: Evaluating the Business Alignment of Information Technologies*. (New Canaan, Connecticut: The Information Economics Press), 1997.

[42] R. Stultz. Options on the minimum or the maximum of two risky assets: Analysis and applications. *Journal of Financial Economics*, 10:161–185, 1982.

[43] K. Sullivan. Software design: the options approach. In *Proc. Software Architectures Workshop*, 1996.

[44] K. Sycara, K. Decker, A. Pannu, and M. Williamson. Distributed intelligent agents. Technical report, Carnegie Mellon University, 1996.

[45] A. Triantis and J. Hodder. Valuing flexibility as a complex option. *The Journal of Finance*, XLV(2):549–565, June 1990.

[46] L. Trigeorgis. The nature of option interactions and the valuation of investments with multiple real options. *Journal of Financial and Quantitative Analysis*, 28(1):1–20, 1993.

[47] L. Trigeorgis, editor. *Real Options in Capital Investments: Models, Strategies and Applications*. (Westport, Connecticut: Praeger), 1995.

[48] L. Trigeorgis. *Real Options: Managerial Flexibility and Strategy in Resource Allocation*. (Cambridge, Massachusetts: MIT Press), 1996.

[49] J. Withey. Investment analysis of software assets for product lines. Technical Report CMU/SEI-96-TR-010, Carnegie Mellon University – Software Engineering Institute, Nov. 1996.