Valuation of Complex Options in Software Development

Hakan Erdogmus

Software Engineering Group
National Research Council of Canada
Montreal Road, Bldg. M-50
Ottawa, Ontario, Canada K1A 0R6
Hakan.Erdogmus@nrc.ca

March 14, 1999[1]

Abstract – Embedded strategic flexibility may have a significant impact on the net worth of a software development project. Disregarding the additional value of flexibility may make a project look less attractive than it actually is. In this position paper we show how strategic flexibility in software projects can be valued in a practical and methodical manner using the concept of real options. The valuation method relies on the identification of two basic kinds of constructs for composing options: staggering and nesting. These two constructs allow the formulation and valuation of a complex strategic scenario as a portfolio of real options. We motivate the work with several examples of options found in software development involving COTS components.

Keywords – *software engineering economics, software investment analysis, Net Present Value, options, real options, strategic flexibility, valuation techniques, option pricing techniques, volatility, discount rate*

Introduction

The role of software as an important source of wealth creation for many businesses drives the push toward a value-based approach to the evaluation of software investments (Favaro and Pfleeger, 1998). The highly uncertain nature of software development makes strategic flexibility one of the focal points of this movement. Embedded strategic flexibility can increase the net worth of a software project significantly. For example, the ability to defer a design decision until further information becomes available, to abandon development if a prototype is unsuccessful, to add and replace software components, to change the maintenance schedule, to reuse software or design artifacts, to modify functionality, and to migrate to a different architecture are all forms of strategic flexibility that may generate additional value for a software project.

The recognition that flexibility has real economic value under uncertainty has led to the emergence of a sub-discipline of corporate finance: *real options*. The central idea is that strategic flexibility in capital investment decisions can be valued as a portfolio of options on real assets, much akin to options on financial securities (Luehrman, 1998; Dixit and Pindyck, 1995; Trigeorgis, 1994).

Sullivan (1996) and Favaro et al. (1998) have advocated the application of real options in the context of software engineering economics. In this proposal, we build upon the existing body of knowledge (Sullivan et al., 1998; Chalasani et al., 1997; Favaro et al., 1998) to make the real options approach a practical and methodical valuation tool for software investments.

An option gives its holder the right, *without the obligation*, to acquire or dispose of a risky asset at a set *strike price* within a specified time period. If the market conditions are favorable before the option expires, the holder exercises this right, thus making a profit. Otherwise, the holder lets the option expire. This asymmetric nature of options gives them real economic value.

Real options in software development have a distinct combination of characteristics that make their analysis difficult:

- Mutiplicity – Several options exist simultaneously.

[1] Revised March 24, 1999; April 10, 1999.

- Complexity – Valuation of certain options that are conceptually regarded as a single unit requires their decomposition to several, more elementary options.
- Interaction – Options often share underlying assets, and therefore affect each other's value.
- Time dependence – The underlying asset of an option is given by the expected benefit or penalty of its exercise. The value of this asset (a) depends on the remaining value of the project at the time of exercise of the option, and (b) is subject to discounting to the present time.
- Discreteness – Options are often exercised at discrete intervals, such as at the beginning or end of release cycles.

Here we propose a valuation technique that addresses these issues. The technique is based on the identification of two types of interactions between options: *staggering* and *nesting*. Staggering captures the ability of the exercise of an option to kill a subsequent option that would have continued to exist otherwise. Nesting captures the ability of the exercise of an option to create a subsequent option that would have ceased to exist otherwise. Using these two ways of composing options, almost any project scenario can be valued. A third kind of interaction concerning multiple options with the same expiry date is also possible, but not discussed here.

Unlike some other approaches, we consider *put options* (the right to dispose of a risky asset) in addition to *call options* (the right to acquire a risky asset). A duality exists between calls and puts. We offer some insight into choosing the value of a key option parameter, interest rate. The choice of this parameter is linked to the call-put duality. We briefly touch upon another interesting topic, volatility, in the discussion section. First, examples of real options in software development are provided for motivation.

Examples of Complex Options in Software Development

Software development based on large-scale use of Commercial Off-The-Shelf, or COTS, components is attracting considerable attention as more and more businesses investigate this approach as an alternative to traditional development (Dean and Vigder, 1997; Carney, 1997). The use of COTS components presents many forms of strategic flexibility that has to be considered during project evaluation, giving rise to a rich context to apply a real options perspective. Here are some examples:

Option to replace components

A developer is looking into using a COTS component to implement a critical subsystem of a new application. None of the currently available products support an emerging standard that is expected to become important in the future. A new COTS product that supports this standard will be on the market some time after the application has been deployed. The application will be designed to accept COTS components. Thus the developer will have the option to replace that subsystem at a cost within a given window when the new COTS product becomes available. In return, the new COTS product promises an increase in the remaining asset value of the project by a certain percentage. The increase is attributed both to the potential impact of the emerging standard on the market's acceptance of the application and to the expectation that the COTS product will be competitively priced. However, the net payoff from the replacement is subject to uncertainty. If the conditions are favorable in the future, the developer will exercise the replacement option within the given window when it seems likely that the net payoff from the benefit will exceed the cost of replacement. Otherwise, the developer will forego the replacement idea altogether. This scenario is an example of leverage through third-party innovations.

Assuming that replacement can occur only at the beginning of a new release cycle, the flexibility to replace in this scenario is akin to a series of staggered call options. The underlying risky real asset here is the net payoff of replacement and the strike price is the expected replacement cost. Since the net payoff is linked to the remaining asset value of the project at any given point when the option to replace presents itself, the underlying asset value varies for each individual option in the series while the strike price remains constant. The value of this options portfolio increases along with the underlying flexibility,

which in turn is proportional to the width of the exercise window. The overall portfolio resembles an American option, but cannot be valued as such due to the time-dependence of the underlying asset value.

Phased migration option

The functionality and quality of systems assembled from COTS components may be difficult to control, with a potential negative impact on the net worth of the project. When time to market for the COTS strategy is expected to be relatively short, a reasonable approach is to initially develop a COTS-based system as a software prototype (Chalasani et al., 1997). The resulting system can then be slowly migrated into a custom system by gradually replacing the unsatisfactory COTS components by their proprietary counterparts. The component-based organization of such a system is inherently conducive for the migration to take place in multiple phases. This strategy permits reclaiming the functionality and quality disadvantage of the COTS components while sustaining a market presence from an early stage.

The migration strategy is not a time-zero decision. Migration proceeds only as long as the market remains receptive to the product, the revenues are likely to increase by using further proprietary components, and the development cost of the proprietary components remains inferior to the expected benefit of migration. Thus executing a migration phase creates a further opportunity to execute a subsequent phase. This scenario is akin to a series of nested call options, where each call represents a single phase. The value of the migration option may rise or fall as the number of phases increases, depending on the discount rate, the total benefit and cost of the migration, and the distribution of the total cost and benefit among the individual phases.

Option to delay upgrades

Upgrade refers to the regular maintenance activity of replacing one or more COTS components by their more recent releases. A major problem with upgrades is uncertainty as upgrade costs may vary significantly from one release to another (Vigder and Dean, 1998).

While upgrade uncertainty increases the total risk, paradoxically, it may also increase the value of a project when it is considered as a semi-discretionary activity. Since maintenance does not revolve around critical bug fixes and essential functional improvements, at the beginning of a new release cycle the developer may choose not to upgrade for that cycle if the associated cost is likely to be higher than the benefit. This decision has an uncertain penalty on the expected future cash inflows, which can be expressed in terms of a temporary drop in the remaining asset value of the project. However, the number of consecutive no-upgrade cycles must have an upper limit to prevent product obsolescence. The penalty of delaying an upgrade is compounded for each consecutive no-upgrade period. Therefore, the developer cannot delay upgrades indefinitely. In other words, exercising the delay option repeatedly will eventually kill the option temporarily. Immediately following an upgrade cycle the option to delay is effectively reinstated.

Each instance of the opportunity to upgrade is akin to a put option. The overall upgrade scenario can be thought of as a complex portfolio of nested and staggered put options. The options interact in the sense that the exercise of an early option may kill existing subsequent options while creating new ones.

Real Option Valuation Model

The valuation model that we propose has two inherent assumptions: (1) uncertainty of an option's underlying asset is resolved gradually and continuously over time; and (2) events can happen only at discrete time intervals. Assumption 1 implies the use of a continuous formulation as suggested by Black and Scholes (1973). Assumption 2 implies that all options have a discrete exercise scenario, and thus are similar to European options.

As customary, we formulate a capital investment project is as a portfolio of real options. Every option in a portfolio is identified by its e*xercise time* (T), *underlying asset value* (U), and *strike price* (P). For a

call option, U is the present value of the expected benefit of exercise and P is a cash outflow representing the amount to be paid to receive this benefit. For a put option, U is the present value of the expected penalty of exercise and P is a cash inflow representing the amount to be saved by accepting to incur this penalty. Often, a put option is associated with foregoing a planned expense. In both cases the strike price is incurred immediately upon the exercise of the option, at time T.

Two additional variables affect the value of an option: the *interest rate* at which the strike price is to be discounted, and the per-period *volatility* of the underlying asset. We will discuss these variables later.

Each option of a portfolio decomposes that portfolio into *stationary* and *discretionary* parts. An option is called *complex* if it precedes other options in the portfolio. When the stationary part of the portfolio with respect to an option A contains a subsequent option B, the exercise of B is contingent upon the expiry of A. In this case, B is said to be *staggered* within A. If the discretionary part a portfolio with respect to an option A contains a subsequent option B, the exercise of B is contingent upon the exercise of A. Then option B is said to be *nested* within option A.

The value of a portfolio is defined by its Net Present Value (Ross et al., 1996).[2] The portfolio NPV is given by the following equation:

$$NPV = NPV_0 + \Delta_0(U_0, P_0; T_0)$$

where NPV_0 is the NPV of the stationary part of the portfolio with respect to the earliest option and Δ_0 is this option's premium. The option premium is always calculated relative to the stationary part of the project with respect to that option. The underlying asset value U_0 is calculated using the present value of the stationary (S) and discretionary (D) parts: for a call option,

$$U_0 = \max(0, PV[S^{(T0)}] - PV[D]);$$

and for a put option,

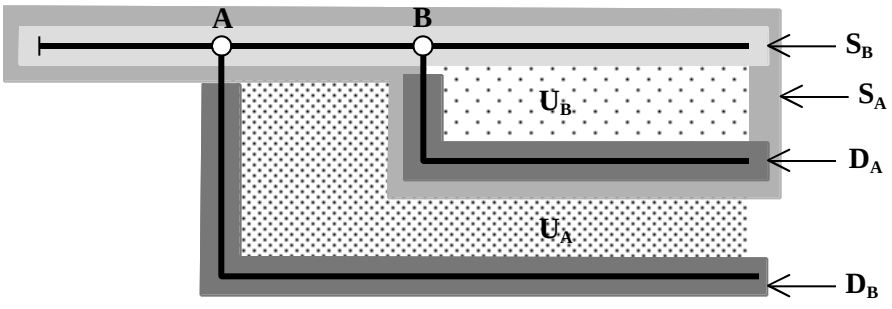
$$U_0 = \max(0, PV[D] - PV[S^{(T0)}]).$$

Here, $PV[S^{(T0)}]$ and PV[D] represent the present value of the project's remaining life beyond the exercise time T_0 for the stationary part and the discretionary part, respectively.

If the option is simple, the option premium is calculated in a straightforward manner using the corresponding Black and Scholes formula for the European call or put.

Options That Kill Options: The Effect of Staggering

Consider the portfolio of options illustrated in the diagram, where option B is staggered within option A. The diagram identifies the stationary (S) and discretionary (D) parts of the portfolio with respect to each option.



[2] The use of NPV in software engineering economics was first discussed by Boehm (1981), and more recently by Favaro (1996).

Option A is complex due to staggering, whereas option B may be simple or complex. Since A is the earliest option, $NPV = NPV_A + \Delta_A$, where $NPV_A = NPV_B + \Delta_B$. The calculation of Δ_A must take into account Δ_B. The underlying asset value of option A (U_A) always includes the asset value of D_A relative to S_B ($U[D_A|S_B]$). If A is a call option, the asset value excludes the option premium of B, hence:

$$U_A = U[D_A|S_B] - \Delta_B.$$

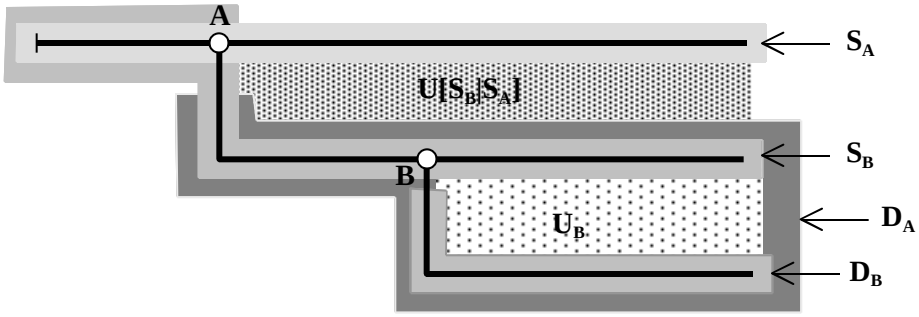
If A is a put option, the asset value also includes the option premium of B, hence:

$$U_A = U[D_A|S_B] + \Delta_B.$$

$U[D_A|S_B]$ is easily calculated by disregarding B, as if A was defined directly on S_B.

Options That Create Options: The Effect of Nesting

Consider now the portfolio of options illustrated below, where option B is nested within option A.



Option A is complex due to nesting, whereas option B may be simple or complex. Again, since A is the earliest option, the portfolio $NPV = NPV_A + \Delta_A$. This time we need to concentrate only the discretionary part with respect to A, where B occurs. The underlying asset value of A, U_A, can be composed into two parts: the asset value of S_A relative to S_B ($U[S_B|S_A]$) and the option premium of B (Δ_B). If A is a call option, the sign of the latter component is positive, hence:

$$U_A = U[S_B|S_A] + \Delta_B.$$

If A is a put option, it is negative, hence:

$$U_A = U[S_B|S_A] - \Delta_B.$$

The quantity $U[S_B|S_A]$ is calculated easily by disregarding option B.

Choice of Interest Rate & Duality of Calls and Puts

In financial option valuation, Black and Scholes model requires the use of the risk-free interest rate. However, this may not be appropriate in the case of real options. In a financial option the strike price is certain, and hence it should be discounted using the risk-free rate. In a real option, the strike price is often subject to a similar systematic risk as the rest of the cash flows. This situation particularly applies to options that exist in software development, owing to the highly uncertain nature of future expenditures. In such cases it is more appropriate to use the same rate both in option value and present value calculations.

We can justify this argument from a different perspective by examining the relationship between put and call options. It is possible to convert a put option into a call option, and vice versa, by switching the stationary and discretionary parts of the portfolio with respect to that option. The resulting portfolio is called the *dual* of the original.[3] Intuitively dual portfolios should have the same NPV, as the net worth of

[3] We can show that the dual of a portfolio consisting of a series of *nested call options* is a portfolio consisting of a series of *staggered put options*.

a project should not depend on a particular way of interpreting its embedded flexibility. Indeed, dual portfolios have the same NPV precisely when the interest rate used in the Black and Scholes option calculations equals the discount rate used in the present value calculations of the project.

Discussion

This position paper proposed a method suitable for analyzing real options found in software development projects. We have identified two basic ways of composing options: staggering and nesting. These constructs allow the formulation and valuation of complex scenarios in a systematic way. We are currently in the process of analyzing several example scenarios, including the ones discussed here, using mathematical software.

We employ the Black and Scholes (1973) formulas to value simple real options. The Black and Scholes option pricing theory is founded on the concept of gradual resolution of uncertainty over time and continuous hedging against this uncertainty (Cox et al., 1979). Although, the theory makes questionable assumptions that do not necessarily apply to real assets that are not traded in efficient markets, the simplicity and practicality of the resulting equations make this approach attractive. The alternative is to use a discrete technique, founded on dynamic decision tree analysis and Bayesian principles (Chalasani et al., 1997; Sullivan et al., 1998; Dixit and Pindyck, 1995; Trigeorgis, 1994; Favaro et al., 1998). However, discrete techniques require the probabilities of the possible states of future as well as the conditional distributions of the underlying asset value to capture uncertainty. We find it more convenient to use a continuous technique that represents uncertainty through a single time-dependent volatility measure.

The existing real options literature unfortunately fails to explain in a satisfactory manner the impact of this volatility measure on the distribution of the underlying asset value. As such, volatility remains a mysterious notion in real option valuation. The expectation of the compounded per-period return rate of the underlying asset equals the discount rate used in the present value calculations. The Black and Scholes volatility measure is precisely the spread, or standard deviation, of this return rate. At the end of any given finite period the underlying asset value is assumed to have a lognormal distribution. This makes it possible to determine numerically the value of the Black and Scholes volatility measure without appealing to the concept of historical volatility, provided that the expectation of the asset value at the exercise time of the option and the discount rate are given.

Finally, we stress on the qualitative aspect of valuation techniques over the quantitative aspect. Valuation techniques should be viewed as a way of thinking about software investments rather than as tools for obtaining hard estimates.

References

Black, F. and Scholes, M., "The pricing of options and corporate liabilities," *Journal of Political economy*, 1973

Boehm, B., *Software Engineering Economics*, Prentice Hall, 1981.

Carney, D., "Assembling large scale systems from COTS components: opportunities, cautions, and complexities," SEI Monographs on Use of Commercial Software in Government Systems, Software Engineering Institute, Pittsburgh, PA, June 1997.

Chalasani, P., Jha, S., and Sullivan, K., "The options approach to software prototyping decisions," Carnegie-Mellon University Department of Computer Science, Technical Report, August 1997.

Cox, J.C., Ross, S.A., Rubinstein, M., "Option pricing: a simplified approach," *Journal of Financial Economics*, 7(3), 1979.

Dean, J.C. and Vigder, M.R., "System implementation using COTS software," *Proceedings of the 1997 Software Technology Conference*, Salt Lake City, Utah, 1997.

Dixit, A.K. and Pindyck, R.S., "The options approach to capital investment," *Harvard Business Review*, May-June 1995.

Favaro, J.M. and Pfleeger, S.L., "Making software development investment decisions," *Software Engineering Notes*, 23(5), 1998.
Favaro, J.M., "A comparison of approaches to reuse investment analysis," in *Proceedings of the 4th International Conference on Software Reuse*, 1996.
Favaro, J.M., Favaro, K.R., and Favaro, P.F., "Value based software reuse investment," *Annals of Software Engineering*, vol. 5, 1998.
Luehrman, T.A., "Investment opportunities as a portfolio of real options," Harvard Business Review, July-August 1998.
Ross, S.A. et al., *Fundamentals of Corporate Finance*, Irwin, 1996.
Sullivan, K.J. et al., "Software design as an investment activity: a real options perspective," submitted for publication,1998.
Sullivan, K.J., "Software design: the options approach," *Proceedings of SIGSOFT Software Architectures Workshop*, San Francisco, CA, 1996.
Trigeorgis, L., *Real Options: Managerial Flexibility and Strategy in Resource Allocation*, MIT Press, 1994.
Vigder, M.R. and Dean, J.C., "Building maintainable COTS-based systems," *Proceedings of the International Conference on Software Maintenance*, Washington, DC, 1998.