Evaluating Architectural Stability with Real Options Theory

Rami Bahsoon and Wolfgang Emmerich
Dept. of Computer Science, University College London
Gower Street, WC1E 6BT, London, UK
{r.bahsoon | w.emmerich} @cs.ucl.ac.uk

Abstract

Architectural stability refers to the extent to which a software architecture is flexible enough to respond to changes in stakeholders' requirements and the environment. We contribute to a novel model that exploits options theory to evaluate architectural stability. We describe how we have derived the model: the analogy and assumptions made; its formulation and possible interpretations. We use a refactoring case study to empirically evaluate the model. The results show that the model can provide insights into architectural stability and investment decisions related to the evolution of software systems.

Keywords. Economics-driven software engineering; relationship between requirements and software architecture.

1. Introduction

Architectural stability refers to the extent to which an architecture is *flexible* enough to endure evolutionary changes in stakeholders' requirements and the environment, while leaving the architecture intact and adding to the system and/or the enterprise a value [1, 2]. In an evolutionary context, there is a pressing need for stable software architectures. In this context, requirements are generally *volatile*; they are *likely* to change and evolve over time. The change is inevitable as it reflects changes in stakeholders' needs and the environment in which the software system works. The change may "break" the architecture necessitating changes to the architectural structure (e.g., changes to components and interfaces), architectural topology (e.g., architectural style), or even changes to the underlying architectural infrastructure (e.g., middleware). It may be expensive and difficult to change the architecture as requirements evolve. Consequently, failing to accommodate the change leads ultimately to the degradation of the usefulness of the system.

From an economic perspective, the volatility of requirements may be regarded as a major source of *uncertainty* that confront an architecture during evolution. It places the investment in a particular architecture at *risk*. To cope with uncertainties and mitigate risks in the investment, there is a critical need for evaluating the stability of software architectures. Such evaluation is necessary for valuing the long-term investment in a particular architecture; analysing trade-offs between two or more candidate software architectures; analysing the strategic position of the enterprise- if the enterprise is highly centred on the software architecture (as it is the case in web-based companies); and validating the architecture for evolution.

The novel contribution of this paper is an approach for evaluating the stability of software architectures with real options theory [8,9]. The approach assumes that the software architecture's goal is to guide the system's evolution. The approach views software evolution as a process in which a software system is undergoing a change (an incremental) and seeking a value. The approach attributes the added value to the flexibility of the architecture to respond to the change(s). Means for achieving flexibility are typical architectural mechanisms or strategies that are built-in or adapted into the architecture with the objective of facilitating evolution and future growth. For example, consider a functionality that is likely to change and evolve over time: "componentizing" the functionality and hiding it behind negotiable and configurable interfaces is a simple example of such a mechanism. In this context, the *flexibility* of an architecture to endure changes in stakeholders' requirements and the environment has a value that can *assist* in predicting the stability of software architectures. More specifically, flexibility adds to the architecture values in the form of *real options,* that give the right but not a symmetric obligation- to evolve the software system and enhance the opportunities for strategic growth by making future follow-on investments. The added value is strategic in essence and may not be immediate. It takes the form of (i) accumulated savings through enduring the change without "breaking" the architecture; (ii) supporting reuse; (iii) enhancing the opportunities for strategic "growth" (e.g. regarding an architecture as an asset and instantiating the asset to support new market products); and (iv) the ability to respond to competitive forces and changing market conditions. Specifically, we contribute to a model that exploits Black and Scholes options theory (Nobel Prize winning) [3] to assess flexibility- as a way to predict architectural stability. The model provides "insights" into architectural stability and investment decisions related to the evolution of software systems. This is based on examining a set of *likely* changes and how valuable is the embedded or adapted flexibility in responding to the changes. We describe how we have derived the model: the analogy and assumptions made with [3]; its formulation; and possible interpretations. We refer to this model as ArchOptions. To evaluate the model, we apply it to a refactoring example. Our observations verify that the theory, the model, and its interpretations are sensible.

The paper is further structured as follows. Section 2 shows how we exploit options theory to predict architectural stability. Section 3 evaluates the model. Section 4 previews closest related work. Section 5 concludes.

2. Evaluating Architectural Stability with an Options Analogy

Real options theory, an emerging financial theory, is well suited to address many Software Engineering problems from a value-based engineering perspective [5, 6]. Real options theory provides an analysis paradigm that emphasizes the value-generating power of flexibility under uncertainty [6]. An option is an asset that provides its owner the right without a symmetric obligation to make an investment decision under given terms for a period of time into the future ending with an expiration date [9]. If conditions favorable to investing arise, the owner can exercise the option by investing the strike price defined by the option. A *call* option gives the right to acquire an asset of uncertain future value for the strike price. A real option is an option on non-financial (real) asset, such as a parcel of land or a new product design. In traditional applications, real options analysis recognizes that the value of the capital investment lies not only in the amount of direct revenues that the investment is expected to generate, but also in the future opportunities flexibility creates. These include, abandonment or exit, delay, exploration, learning, and growth options. We view stability as a *strategic* architectural quality that adds to the architecture values in the form of *growth options.* A growth option is a real option to expand with strategic importance [8]. Growth options are common in all infrastructure-based (as it is the case of software architectures) or strategic industries with multiple-product generations or applications [8]. As many early investments can be prerequisites or links in chain of interrelated projects, growth options set the path for the future opportunities [8]. In the architectural context, growth opportunities are linked to the flexibility of the architecture to respond to future changes. Since the future changes are generally unanticipated, the value of the growth options lies in the enhanced flexibility of the architecture to cope with uncertainty; otherwise, the change may be expensive to pursue and opportunities may be lost.

The search for a potentially stable architecture requires finding an architecture that maximizes the yield in the added value, relative to some future changes in requirements. As we are assuming that the added value is attributed to flexibility, the problem becomes maximizing the yield in the embedded or adapted flexibility in a software architecture relative to these changes. We derive a predictive model from Black & Scholes [3] options theory. A likely future change in requirement is said to be analogous to buying an option on an asset, with an exercise price corresponding to the cost of implementing the change. For a likely change in requirements, the model constructs a call option to value the flexibility of the architecture to accommodate the change- as a way to make the value of stability tangible. The value of the call option indicates the ability of an architecture to unlock future growth opportunities and enhance the upside potentials of the architecture.

Under the Black and Scholes model, five parameters are needed to determine the option price: the current stock price (*S*), the strike price (*X*), the time to expiration (*T*), the volatility of the stock price (σ), and the free-risk interest rate(*r*). The price of the stock option is a function of the stochastic variables underlying stock's price and time. The strike price (*X*) is the price for which the holder may exercise a contract for the purchase/sale of the underlying stock; also referred to as the *exercise price*. The volatility of the stock price (σ) is a statistical measure of the stock price fluctuation over a specific period of time; it is a measure of how uncertain we are about the future of the stock price movements. The value of a call option on an asset depends on the value of the asset itself and the cost of exercising the option. The payoff from a call option is the amount by which the stock price exceeds the strike price, if *S* exercised at some time in the future. Call options, therefore, become more valuable as the stock price increase and less valuable as the strike price increases. The expected value of a *European call option* is given by $E\,[\max(S_t - X, 0)]$, where E denotes the expected value of a European call option and S_t denotes the stock price at time *t*. The European call option price, *C*, is the value discounted at the risk-free rate of interest. It calculates to equation *(1)*:

$$C = e^{-r\,(T-t)} E\,[\max(S_t - X, 0)] \qquad (1)$$

In a risk-neutral world, ln S_t has the following probability distribution given by *(2)*,

$$\ln S_t \sim \phi\,[\ln S + (r - \sigma^2/2)(T-t),\ \sigma(T-t)^{1/2}] \qquad (2)$$

where $\phi\,[m, s]$ denotes a normal distribution with mean *m*, and standard deviation *S*. Evaluating the right-hand side of *(1)*- in application of integral calculus- results in Black and Scholes valuation of a call option.

$$C = S\,N(d_1) - Xe^{-r\,(T-t)}\,N(d_2) \qquad (3)$$

where,

$$d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)(T-t)}{\sigma(T-t)^{1/2}}$$

$$d_2 = \frac{\ln(S/X) + (r - \sigma^2/2)(T-t)}{\sigma(T-t)^{1/2}} = d_1 - \sigma(T-t)^{1/2}$$

N (x) is the cumulative probability distribution function for a standardized normal variable (i.e., it is the probability that such a variable will be less than *x*).

To assess flexibility, we map the economic characteristics of the architecture (under development or evolution) onto the parameters of the option model of *(1)*- as shown in Table 1. The economic characteristics include the development (evolution) effort, schedule, and budget. We exploit (2) and *(3)* to valuation.

Table 1. Financial/real options/ArchOptions analogy

Option on stock	Real option on a project	ArchOptions
Stock Price	Value of the expected cash flows	Value of the architectural potential of the change (*x_iV*)
Exercise Price	Investment cost	Estimate of the likely cost to accommodate the change *(C_{ei})*
Time-to-expiration	Time until opportunity disappears	Time indicating the decision to implement the change (*t*)
Volatility	Uncertainty of the project value	"Fluctuation" in the return of value of V over a specified period of time (σ)
Risk-free interest rate	Risk-free interest rate	Interest rate relative to budget and schedule (*r*)

Let us assume that the value of the system is V. As the software evolves, a change in future requirement i_i is assumed to enhance the system value by $x_i\%$ with a follow-on investment of C_{ei}, where C_{ei} corresponds to an estimate of the likely cost to accommodate the change. This is similar to a call option to buy (x_i%) of the base project, paying C_{ei} as exercise price. Thus, the investment opportunity in the system can be viewed as a base-scale investment plus call options on the future opportunities, where a future opportunity corresponds to the investment to accommodate some future requirement(s). The payoff of the constructed call option gives an indication of how valuable the flexibility of an architecture to endure some likely changes in requirements. The value of the system having a particular architecture, materializes to ArchOptions- given by *(4)*- accounting for V and both the expected value and exercise cost to accommodate i_i, for i ≤ n. The expectation E is valued using *(2)* and *(3)*. We assume that the interest rate is zero for the simplicity of exposition.

$$V + \sum_{i=0}^{n} \mathrm{E}\left[\max\left(x_iV - C_{ei},\, 0\right)\right] \qquad (4)$$

For a likely change in requirement k, we interpret *(4)*:

(a) The option is *in the money*: if x_kV exceeds the exercise cost (i.e. max (x_kV - C_{ek}, 0) >0), then the architecture is said to be *potentially stable* with respect to i_k. The more the option is in the money, the more valuable is the embedded flexibility; hence, the better are the potentials for the stability of the architecture with respect to the change. In real situations, the architect/analyst is interested in selecting an architecture that maximizes the yield in options relative to some likely changes. An optimal selection could be when the option value approaches the maximum, indicating an optimal payoff in an investment in flexibility. The analyst may perform sensitivity analysis and analyze when such a situation is likely to occur.

(b) The option is *out of money*: if the value of the call option sinks to zero (i.e. max (x_kV - C_{ek}, 0) =0), then the flexibility of the architecture in response to the change is not likely to add a value. The architecture is likely to be unstable for this change (i.e., stability is when flexibility adds value). Two interpretations might be possible: (i) the architecture is overly flexible in the sense that its response to the change has not "pulled" the options. This implies that the embedded flexibility (or the resources invested in implementing flexibility) are wasted and unutilized to reveal the options relative to *this* change. In other words, the degree of flexibility provided is much more than the flexibility demanded for this change. This case has the prospect in providing an insight on how much do we need to invest in flexibility to achieve stability relative to the likely future changes, while not sacrificing much of the resources; (ii) the other case is when the architecture is inflexible relative to the change. This is when the cost of accommodating the change is much more than the cumulative expected value of the architecture responsiveness.

Estimating C_{ei}. Estimating cost is a well-established component in software engineering; it is outside the scope of our work. For example, it is feasible to use existing metrics to cost estimation (e.g., COCOMO-II [7]). Another approach is to build on architectural level dependency analysis (e.g., [13]) research to extract cost estimates of accommodating i_i, guided by some structural criteria.

Estimating x_iV. Black and Scholes is an *arbitrage-based* technique. The technique requires knowledge of the value of the asset in question in span of the market. Software architectures, however, are (non-traded) real assets. Real options may be valued similarly to financial options, though they are not traded [9]. Real options valuation based on arbitrage determines the value of an asset in question in span of the market value using a correlated *twin asset* [9]. The twin asset is an asset that has the same risks the asset in question will have when the investment has been completed [9]. We calibrate changes in requirements with their market value, when available. We argue that valuation based on man-month does implicitly hold market-based data and is still done in relation with the market. We adopt a viewpoint-oriented framework for making such calibration implicit or explicit, as detailed in [2].

The application of [3] assumes that the stock option is a function of the stochastic variables underlying stock's price and time. We assume that V moves stochastically bounded to two extreme values: optimistic and pessimistic. This assumption appears to be plausible: (i) it tends to account for all possible values within the bound, yielding to a better approximation when opposed to an ad-hoc type of estimation; (ii) the value of an (evolvable) system changes over time; it tends to change in uncertain way due to changes in requirements.

3. Case Study

We empirically simulate the proposed theory, demonstrate the applicability of the model, and validate its interpretations. We summarize the simulation rationale as follows: (a) embed flexibility in the software structure (b) observe the responsiveness of the structure to some random changes in requirements following action (a); (c) quantify flexibility and interpret it relative to likely future changes as a way for understanding stability. To achieve the simulation rationale, we use refactoring, a preventive change. Refactoring can be seen as an upfront investment to embed flexibility, given by I_e. The objective is to "clear up" much of the system degraded structure and enhance its upside potentials by making it more accommodating for likely future changes. Refactoring may add value to the structure. The added value could take the form of growth options, enhancing the flexibility of the structure with respect to future changes. Future changes following refactoring will tell us how valuable these options are. But, the benefits due to refactoring are uncertain as the demand on future changes are uncertain. This makes refactoring a good candidate to reason using option "thinking". We use the refactoring case study of a traffic light system published in [10], which proposes a framework to predict the return on investment (ROI) for a planned refactoring using cost-benefit analysis. We recast the problem into an option problem: we consider the benefits of refactoring to be uncertain as the demand for future changes -following refactoring- is uncertain. We restrict architectural information to data and control dependency for this example. Table 2 summarizes the structural changes upon evolving S_0 (the initial structure) to S_1 (the refactored structure) of the traffic light system. Table 2 shows that refactoring has transformed the structure into a more flexible state through the decrease of both control and data dependencies. The decrease in dependencies in S_1 means less complexity and better prospects for accommodating future changes.

Table 2. Aggregate results: the change (%)- evolving S_0 to S_1

	S_0	S_1	Change (%)
Size in SLOC	740	602	-19%
No. of Modules	29	38	31%
Avg. SLOC Per Module	26	16	-38%
Data Dependency	147	112	-23.60%
Control Dependency	101	73	-19.40%

We apply the model: we construct a call option for the likely changes following refactoring, accounting for I_e. The model materializes to $- I_e + \sum_{i=1\ldots n} E\ [\max (x_iV - C_{ei}, 0)]$ [2]. To estimate x_iV, we restrict the valuation to the development perspective. We use the expected savings in development effort for likely futures changes due to refactoring. We use \$2000 for man-month to cast the effort into cost.

Estimating (I_e). Table 3 reports the refactoring effort (man-month), cost (\$), and schedule (month) based on the refactoring plan presented in [10]. Table 3 provides three values: optimistic, likely, and pessimistic for each parameter. All are calculated using COCOMO II.

Estimating (x_iV). To value the architectural potential of S_1 due to refactoring, we use twenty random changes to stress S_1 with cost given as C_{ei}. The twenty changes are of an adaptive nature; they are generated based on percentage estimates of design, integration, and code to be modified per change. The same likely changes were used to stress S_0. The objective is to calculate the difference (i.e., savings-if any) in effort/cost of S_1 over S_0. The aim is to quantify the responsiveness of the structure due to the embedded flexibility, from the development perspective. We use COCOMO II to estimate the effort/cost for the twenty changes on each structure. x_iV corresponds to the difference- as reported in Table 4. Expected savings, due to refactoring, are in the range of \$12806 (optimistic) to \$7433 (pessimistic) for the twenty changes.

Estimating volatility (σ), Exercise time (t) and free risk interest rate(r). We take the percentage of the standard deviation of the three x_iVs estimates-the optimistic, likely, and pessimistic values- to calculate σ. As a simulation assumption, we set the exercise time to three years. We set the free risk interest rate to zero (i.e., assuming that the value of money today is equal to that in three years time).

Table 3. Refactoring effort, schedule, and cost

	Effort			Schedule			I_{ei}		
	Op	Lik	Pes	Op	Lik	Pes	Op	Lik	Pes
Refactoring	0.9	1.2	1.5	3.6	3.9	4.2	1893	2366	2958

Observation 1. Flexibility creates options: S_1 is more flexible than S_0 (due to decrease in dependencies as a result of refactoring); S_1 has created more options when compared to S_0.

Table 5 shows that S_1 is in the money in response to the twenty random changes, relative to the development perspective. The results read that refactoring (i.e. as the embedded flexibility in S_1) is likely to enhance the option value by an excess of \$5979 (pessimistic) to \$10593 (optimistic) over S_0, if the twenty changes need to be exercised following refactoring. Thus, as flexibility is improved, S_1 is likely to add value in the form of options in response to the twenty changes.

Table 4. Options on S_1 relative to S_0 (\$) for the twenty changes

	Pessimistic			Likely			Optimistic		
	C_{ei}	T	x_iV	C_{ei}	T	x_iV	C_{ei}	T	x_iV
	1454	3	7433	1817	3	9292	2212	3	12806
Option	5979.09			7474.6			10593		

Observation 2. How worthwhile is it "buying" flexibility to achieve architectural stability?

Let us take the average value of the twenty changes. The objective is to simulate the responsiveness of S_1 to one likely average change. The result of table 5 implies that though S_1 is flexible, one change has not "pulled" the options. S_1 is said to be out of the money for this change. This implies that the embedded flexibility (or the resources invested in implementing flexibility) are wasted and unutilized to reveal the options relative to this change. In other words, the degree of flexibility provided is much more than the flexibility demanded for this change. S_1 is unstable for one average change, for flexibility does not add a value, if this change needs to be exercised. Intuitively, if an architecture does not reveal an added value upon the evolution of the software, it could be far from being considered stable. We repeat the above experiment, but stressing S_1 with two, three, four, and then ten average changes at a time. Using two average likely changes, the options reported zero values. Again, two likely average changes have not "pulled" the options. Interestingly, S_1 has just about pulled the options for three changes. For four, five, and nine changes, S_1 reveals the options; however, flexibility is not likely to payoff as $(- I_e + \sum_{i=1 \ldots n} E\ [\max (x_i V - C_{ei}, 0)]_{S1} < 0)$. For ten changes, refactoring is expected to payoff as $(- I_e + \sum_{i=1 \ldots n} E\ [\max (x_i V - C_{ei}, 0)]_{S1} > 0)$. Thus, flexibility is likely to add to the system a value, if ten or more changes need to be exercised during the next three years. Hence, S_1 is likely to reveal stability, as the embedded flexibility is likely to add value for these changes.

Table 5. Options on S_1 for one to ten changes at a time

Changes	σ	x_iV			Options		
		Pes.	Lik.	Op.	Pes.	Lik.	Op.
1Req.Ch.	1.4	371.7	464.6	640.3	0	0	0
2 Req.Ch.	2.7	743.3	929.2	1280.6	0	0	0
3 Req.Ch.	4.1	1115.0	1393.8	1920.9	0+	0+	1.2
4 Req.Ch.	5.5	1486.6	1858.4	2561.2	73.6	92.45	334.9
5 Req.Ch.	6.8	1858.3	2323.0	3201.5	405.6	507.6	989.07
9 Req. Ch.	12.2	3339	4181.4	5760	1885	2364	3547
10 Req. Ch.	13.6	3717	4640	6400	2263	2823	4188

This case has the prospect of providing an insight into how much we have to invest in flexibility to achieve stability relative to the likely future changes, while not sacrificing much of our resources. In real situations, an optimal stability could be when the option value approaches the maximum, indicating an optimal payoff in an investment in flexibility. The analyst may conduct sensitivity analysis to manipulate the model variables and analyze when such a situation is likely to occur.

4. Related Work

Economics approaches to software design appeal to the concept of static Net Present Value (NPV) as a mechanism for estimating value [5]. These techniques, however, are not readily suitable for strategic reasoning of software development as they fail to factor flexibility [5,6]. Real options theory has been adopted to address this problem. Sullivan [12] suggested that real options analysis can provide insights concerning modularity, phased projects structures, delaying of decisions and other dynamic software design strategies. Sullivan et al. [11] argued that the structure and value of modularity in software design creates value in the form of real options. The value of such an option could be realized by the module optimal experiment-and-replace policy.

In [7], Jazayeri takes a retrospective approach to evaluating architectural stability using simple metrics such as size, coupling, and color visualization to summarize the evolution pattern of the software system across its successive releases. The evaluation requires information to be kept for each release of the software. Yet, such data is not commonly maintained, analyzed, or exploited.

5. Conclusions and Future Work

To sum up, following the argument of [11], such models need not be perfect: what is essential is that they capture the most important terms; their assumptions and operation must be known and understood so that the analyst can evaluate their predictions. The observations verify that the theory and the model interpretations are reasonable. In real situations, the changes may be exemplified using change scenarios captured from stakeholders. Our future work entails further evaluation of the model. The objectives are to test the model on real scale projects and scenarios, explore its potentials and limitations, observe, learn, and tune the interpretations (if necessary).

6. References

[1] Bahsoon, R., Emmerich, W.: ArchOptions: A Real Options-Based Model for Predicting the Stability of Software Architecture. In: Proceedings of the Fifth ICSE Workshop on Economics-Driven Software Engineering Research (2003)

[2] Bahsoon, R., Emmerich, W.: Applying ArchOptions to Value the Payoff of Refactoring. In: Proceedings of the Sixth ICSE Workshop on Economics-Driven Software Engineering Research (2004)

[3] Black, F., Scholes, M.: The Pricing of Options and Corporate Liabilities. Journal of Political Economy (1973)

[4] Boehm, B., Clark, B., Horowitz, E., Madachy,R., Shelby, R., Westland, C.: The COCOMO 2.0 Software Cost Estimation Model. In: International Society of Parametric Analysts (1995)

[5] Boehm, B., Sullivan, K. J.: Software Economics: A Roadmap. In: Finkelstein, A. (ed.): The Future of Software Engineering (2000)

[6] Erdogmus, H., Boehm, B., Harriosn, W., Reifer, D. J., and Sullivan, K. J.: Software Engineering Economics: Background, Current Practices, and Future Directions. In: Proceeding of 24th International Conference on Software Engineering, Orlando, FL. (2002)

[7] Jazayeri, M.: On Architectural Stability and Evolution. Lecture Notes in Computer Science, Springer Verlag, Berlin (2002)

[8] Myers, S. C.: Finance Theory and Financial Strategy. Corporate Finance Journal. Vol. 5(1).(1987) 6-13

[9] Schwartz, S., Trigeorgis, L.: Real options and Investment Under Uncertainty: Classical Readings and Recent Contributions. MIT Press Cambridge, Massachusetts (2000)

[10] Stroulia, E., Leitch R.: Understanding the Economics of Refactoring. In: Proceedings of the Fifth ICSE Workshop on Economics-Driven Software Engineering Research (2003)

[11] Sullivan, K. J., Griswold, W., Cai, Y., Hallen, B.: The Structure and Value of Modularity in Software Design. In: Proceedings of ESEC/FSE-9, Vienna, Austria (2001) 99-108

[12] Sullivan, K. J.: Chalasani, P., Jha, S., Sazawal, V.: Software Design as an Investment Activity: A Real Options Perspective. In: Real Options and Business Strategy: Applications to Decision-Making. Trigeorgis L.(ed.) Risk Books (1999)

[13] Zhao, J.: Using Dependence Analysis to Support Software Architecture Understanding: In M. Li (Ed.): New Technologies on Computer Software International Academic Publishers, September (1997) 135-142