Evaluating Software Architectures for Stability: A Real Options Approach

Rami Bahsoon (Student), Wolfgang Emmerich(Supervisor)
Department of Computer Science, University College London,
Gower Street, London WC1E 6BT,UK
{r.bahsoon, w. emmerich}@cs.ucl.ac.uk

1. Research problem and importance

Architectural stability refers to the extent an architecture is *flexible* to endure evolutionary changes in stakeholders' requirements and the environment, while leaving the architecture intact.

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 tension between an unstable architecture and the volatile requirements may entail large and disruptive changes for the requirements to be accommodated. 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*, where a style is a generic description of a software architecture), or even changes to the underlying architectural infrastructure (e.g. middleware). It may be expensive and difficult to change the architecture as requirements evolve [6]. Consequently, failing to accommodate the change leads ultimately to the degradation of the usefulness and the value of the system.

From an economic perspective, the volatility of requirements is a source of uncertainty that places the long-term investment in a particular architecture at risk. If the business goal that the system should be long-lived, should evolve to accommodate future changes, and should create future value, stability becomes an important architectural quality to evaluate an architecture for. The evaluation is necessary to cope with the incomplete knowledge in an evolutionary context and mitigate risks in the long-term investment in a particular architecture. The evaluation is crucial for analyzing trade-offs between two or more candidate software architectures for stability; analyzing the strategic position of the enterprise- if the enterprise is highly centered on the software architecture (as it is the case in web-based service providers companies: e.g. amzon.com); valuing the long-term investment in a particular architecture; and validating the architecture for evolution.

Our work addresses the following research question: can we use an economic approach (real-options theory) to systematically evaluate the stability of an architecture in the face of the changing requirements?

The abstract is further structured as follows. Section 2 presents our research claims. Section 3 summarizes our approach in exploiting options theory to evaluate architectural stability. Section 4 discusses related work. Section 5 summarizes our work in progress and expected contributions.

2. The major research claims

We claim that using strategic *value-based* reasoning [10] approach we can evaluate the stability of an architecture in the face of volatile requirements. We argue that *real options theory* [8] is well suited to assist in the evaluation. A stable software architecture adds to the software system and to the enterprise owing the architecture a value. The added value is attributed to *flexibility* and the *options* that flexibility creates over the evolutionary periods of the software system. The added value under the stability context is strategic in essence and not 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. exploring new markets; expanding the range of services while leaving the architecture intact; regarding an architecture as an asset and instantiating the asset to support new market products); and (iv) giving the enterprise a competitive advantage by banking the stable architecture like any other capitalized asset. An option provides the right to make an investment in the future, without a symmetric obligation to make that investment [3, 11]. If conditions favorable to investing arise, the owner can exercise the option by investing the strike price defined by an option. In the architectural context, 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. Hence, the value of the investment in an architecture may not only derive from the direct measurable cash flows of the investment, but also from the ability of an architecture to evolve, unlock future growth opportunities, and cope with uncertainties.

Classical financial techniques, such as Discounted Cash Flow (DCF) analysis and Net Present Value (NPV), fall short in dealing with flexibility and uncertainty [11]. These techniques are valid when valuing an ongoing business or an immediate investment. However, in the case of valuing the stability of software architectures in the face of evolutionary changes, the nature of the investment is long-term and strategic. *Real options theory* [8, 11] addresses the inability of these traditional budgeting techniques to address strategic value under uncertainty.

In short, to evaluate a software architecture for stability using a value-based reasoning approach, we need a technique that is suitable for strategic and long-term valuation, factors flexibility, and makes the value of the options created by flexibility tangible (as a way to make the value of stability tangible). Real options theory appears to be well suited to satisfy our needs.

3. Exploiting options theory to predict architectural stability

Approaches to evaluating software architectures for stability can be *retrospective* or *predictive* [7]. Both approaches start with the assumption that the software architecture's primary goal is to guide the system's evolution. Retrospective evaluation looks at successive releases of the software system to analyze how smoothly the evolution took place. Predictive evaluation provides "insights" on the evolution of the software system based on examining a set of *likely* changes and the extent the architecture can endure these changes. In [1] we take a predictive approach to evaluation. We use *value-based* reasoning to prediction. We exploit options theory to predict the stability of software architectures given *likely* evolutionary changes. Specifically, we derive a predictive model from Black & Scholes [2] financial options theory. Subsequent subsections draw an analogy with [2], make assumptions, formulate, and interpret the model.

3.1. Option pricing using Black & Scholes

The best-known financial option pricing method (the seminal work in the field) is that of Black and Scholes [2] (Nobel Prize winning), which is a solution to a *stochastic* calculus problem.

Under the Black and Scholes model, five parameters are needed to determine the option price. These are 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 at which the holder may exercise a contract for the purchase/sale of the underlying stock; it is also called the *exercise price*. The current stock price (S) if exercised at some time in the future, the payoff from a *call option* will be the amount by which the stock price exceeds the strike price. A *call option* gives the right to acquire an asset of uncertain future value for the strike price. The value of a call option on an asset depends on the value of the asset itself and the cost of exercising the option. Call options, therefore, become more valuable as the stock price increase and less valuable as the strike price increases. 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 expected value of a call option is given by E [max (S_t- X, 0)], where S_t denotes the stock price at time t. The call option price, C, is the value discounted at the risk-free rate of interest. It calculates to *(1)*.

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

3.2. Analogy and assumptions

A major insight behind real options theory is that flexibility in real asset is analogous to financial options: investing in flexibility is seen as buying options and exploiting flexibility is seen as exercising them [13]. Having set flexibility as an option problem, the challenge becomes valuing flexibility: we adopt [2] to valuation. We map the economic characteristics of the architecture (under development or evolution) onto the parameters of the option model (1)- as shown in Table 1. The economic characteristics include the development (evolution) effort, schedule, and budget.

Black and Scholes is an *arbitrage-based* technique; it requires knowledge of the value of the asset in question in the span of the market. Software architectures are (non-traded) real assets. Real options valuation based on arbitrage-based pricing techniques determines the value of an asset in question in the span of the market value using an equivalent *twin asset* [11]. The twin asset is an asset with the same risks characteristics as the project (or asset in question) if the option were "exercised". To facilitate valuation using the twin asset, we view the architecture as a portfolio of requirements- a portfolio of assets. In this context, we argue that the value of the architecture is in the value of the requirements it supports during the system's operation or tend to support as it evolves. The application of [2] assumes that the stock option is a function of the *stochastic* variables underlying stock's price and time. We assume that value of an evolvable architecture changes with time. It tends to change in *uncertain* ways and stochastically with the cost/value arising from changes in requirements.

Table 1. Financial/real options/software architecture analogy

Option on stock	Real option on project	Case of evaluating architectural stability
Stock Price	Value of the expected cash flows	Value of the likely change
Exercise Price	Investment cost	Estimate of the likely cost to accommodate the change
Time-to-expiration	Time until opportunity disappears	Time-to-release (and deploy) the software generation
Volatility	Uncertainty of the project value	"Fluctuation" in the value of the requirement as deemed by the stakeholders; or changes in market-value over a specified time
Risk-free interest rate	Risk-free interest rate	Interest rate relative to budget and schedule

3.3. Formulation: Constructing call options to make the value of stability tangible

Generally speaking, evolutionary changes are unanticipated. We assume that we can elicit a set of representative changes in requirements $\{i_1, i_2, \ldots, i_n\}$ that are *likely* to occur. Let assume that the value of the architecture is V, where V corresponds to current stock price S_t. As the architecture evolves, the change in i_i is assumed to enhance the architecture value by x_i % with a follow-up investment of I_{ei}, where I_{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 I_{ei} as exercise price. Thus, the investment opportunity in an architecture can be viewed as a base-scale investment in the architecture plus a *call option* on the future opportunity, where the future opportunity corresponds to the investment to accommodate the evolving requirement(s). The value of the constructed call options give an indication of the flexibility of the architecture to endure the likely changes in requirements $\{i_1, i_2, \ldots, i_n\}$. Thus, the value of the architecture materializes to (2) accounting for V and the expected value and exercise cost to accommodate i_i for $i \leq n$. We assume that the interest rate is equal to zero for the simplicity of exposition.

$$V + \sum_{i=0}^{n} E\,[\max(x_i V - I_{ei}, 0)] \quad (2)$$

3.4. Interpretation

We give a rough interpretation of (2) in the context of the evaluation for architectural stability. For a likely change in requirement i_k: (a) the option is *in the money*, if $x_k V$ exceeds the exercise cost (i.e. max $(x_k V - I_{ek}, 0) > 0$). In this case, the architecture is said to be *potentially stable* with respect to i_k. Generally speaking, the higher the value $x_k V$, the better the chances to exceed the exercise price of the option. (b) The option is *out of money*, if the value of the call option sinks to zero (i.e. max $(x_k V - I_{ek}, 0) = 0$). In this case, there is no chance that the option will ever worth something in the future. The change is said to exhibit future *threats on the stability of the architecture*; the architecture is unlikely to be stable for *this* change.

Accounting for *all* the n likely changes in $\{i_1, i_2, \ldots, i_n\}$: If the cumulative expected value of the future investments in all the change tends to zero, it is very *unlikely* for the architecture to be *stable* with respect to the likely evolutionary changes. Hence, the architecture does not tend to create any future growth opportunities. We interpret the *strategic* value of the investment in an architecture as the acquisition of a base asset that embeds growth opportunities. The values of the call options indicate the ability of an architecture to unlock future growth opportunities and enhance the upside potentials of the architecture. In case of trade-offs, we interpret the strategic value relative to other candidate architectures. The more an architecture is able to unlock future opportunities, the more stable and "evolution friendly" it is likely to be.

4. Related work

The only evaluation technique to architectural stability is the work of [7]; it takes a retrospective approach to evaluation. The approach uses simple metrics like software size metrics, coupling metrics, and color visualization to summarize the evolution pattern of the software system across its successive releases. The evaluation appears to be expensive and unpractical; it requires information to be kept for each release of the software. Yet such data is not commonly maintained, analyzed, or exploited. The evaluation assumes that the system already exist and has evolved making this approach not preventive and unsuitable for early evaluation.

Economic approaches to software design appeal to the concept of static NPV as a mechanism for estimating value [4]. These approaches, however, are not readily suitable for strategic reasoning of software development as they fail to factor flexibility. *Real options* theory has been adopted to address this problem: Sullivan [14] 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. [13] formalized that option-based analysis, focusing in particular on the flexibility to delay decisions making. Sullivan et al. [12] argued that the modularity in software design creates value in the form of real options. A module creates an option to invest in a search for a superior replacement and to replace with the best alternative discovered, or to keep if it is still the best choice. The value of such an option could be realized by the optimal experiment-and-replace policy. Knowing this value can help a designer to reason about both investment in modularity and how much to spend searching for superior alternatives.

5. Ongoing work and concluding remarks

Viewpoint-oriented framework to capture the options. The options model (2) requires the estimation of several parameters. The most important are x_iV and I_{ei} which respectively correspond to the expected value of the i^{th} change and its exercise cost, when to be realized by the subject architecture. Estimating the cost I_{ei} is a well-established component in software engineering; it is outside the scope of our work. To estimate x_iV: in financial options several proxies are available to predict the value of the financial asset - the most obvious proxy is simply the historical values of the asset. In real options such proxies rarely exist and the analyst may need to rely on experience and judgment in his/her estimations [5]. Real options valuation (based on arbitrage) focuses on market value and uses the standard deviation of the *rate of return* on the twin asset as an input to the valuation of the asset in question. If the asset value is not *directly observable*, it is reasonable to use estimates of the revenues on the asset to estimate the market value [11]. As the analyst(s) relies on experience and judgment in his/her estimation of the parameters, the estimates tend to be subjective. However, back-of-the envelope calculations, which are based on value estimates (rather on market value) are yet revealing [12]. It remains an open challenge to strongly justify precise estimates for real options in software [12].

Some aspects of follow-up investment in $\{i_1, i_2, \ldots, i_n\}$ can be justified in terms of the *directly observable* cash flows linked to future operational benefits. However, many aspects that contribute to the market value are *indirectly observable* through cash flows. Yet, their contribution to the added value is crucial for predicting the stability of the architecture. The process of valuing the expected return (value) of the follow-up investment in architectures of large complex systems necessarily involves many parties- each with their own *perspective* on the system defined by their valuation objectives, assessment regime, skills, responsibilities, knowledge, and expertise. More, each of the concerned parties might find it necessary to align the valuation with either the business, organizational, system, and/or market objectives. The problem of how to guide the valuation and introduce discipline in this setting, we term as *the multiple perspectives valuation problem*. To address this problem, we suggest a conceptual viewpoints-oriented framework- sketched in figure 1. The framework is built around four main concepts: *viewpoint*, *perspective*, *relative view*, and *view*. A viewpoint is a standing or mental position used by an individual or group of individuals when examining a universe of discourse. A *perspective* is a set of facts observed and modeled according to a particular aspect of reality. We introduce the notion of *relative view*. A relative view integrates the perspectives of *this* viewpoint. We define a *view* as an integration of the all the relative views.

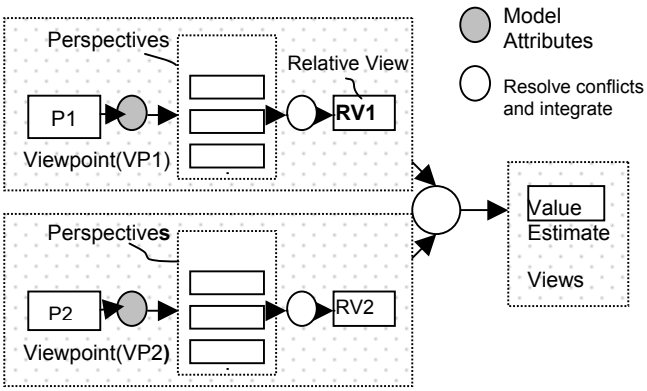

Figure 1. Viewpoints framework to capture the options

We assume that the universe and the set of discourse are: the architecture under evaluation, $\{i_1, i_2, \ldots, i_n\}$; budget and schedule; and other constraints. A viewpoint slot maintains the perspective and the relative view. We assume that there exist m *parties* $\{P_1, P_2, \ldots, P_m\}$ involved in the evaluation for stability; where P_i corresponds to an owner of the i^{th} viewpoint. We use the term party to describe a group of "actors" or "knowledge source" that advocate this viewpoint. A P_i is interested in knowing what value the architecture adds to their concerns upon realizing the change $\{i_1, i_2, \ldots, i_n\}$. The parties' concerns are described by a set of *attributes* $\{A_1p_i, A_2p_i, \ldots, A_kp_i\}$. An A_jP_i, for all $j \leq k$, is quantified and assigned an *attribute value* taking a *valuation strategy*. A valuation strategy is a method, a technique, or a regime used to assess an attribute. The valuation strategy may be aligned with the value preposition and objectives (e.g. market, customer, … etc.). Note valuing these attributes is crucial to the evaluation of software architecture for stability relative to the P_i's standing point. In software it is not usual to value using a unique criteria; the criteria vary

across viewpoints, with domains, products, and/or companies. Thus, the attributes may address different concerns. Their attribute values may be of different unit scales. However, they all pour in determining the expected return on the investment. The *relative view* integrates the attribute values and determines the value relative to this *viewpoint*. The attributes, attributes value, relative views, and valuation strategy are elements that constitute the perspective slot in the suggested framework. To obtain a value estimate from $\{P_1, P_2, \ldots P_m\}$, the relative views are integrated to form views. This can be achieved by building on multi-attribute analysis methods [15]. Viewpoints research (e.g. [9]) sets the path for dealing with various problems that may arise from the integration (e.g. conflicts).

Tuning the interpretation of the option model. The model is critical to its input $\{i_1, i_2, \ldots, i_n\}$. To "tune" the interpretation of the model, we are currently defining metrics to analyze and "profile" the input. These metrics tend to measure the extent to which an input is revealing to modification- measured in relevance to value and cost- when i_i exercises the subject architecture.

"Operationalizing" the model. We are in the process of formulating a three-phased evaluation method for evaluating software architectures for stability and evolution. The *first phase* captures the options, estimates the parameters of (2) and "profile" the input using the metrics; the *second phase* values the options; and the *third phase* assesses stability and interprets the results.

Evaluation. We will *empirically* evaluate the approach in an industrial setting with SearchSpace, one of UCL industrial partners. The evaluation aims to test the *maturity* of the approach, its *applicability*, and the *effectiveness* of the predictive model. SearchSpace is investigating changing one of its products' architectural infrastructure from CORBA to EJB. The investment in the change will increasingly be made on the basis of the stability that the architectural infrastructure creates with respect to the forward-looking strategic benefits. Roughly speaking, changing the product infrastructure from CORBA to EJB may (or may not) create growth options. These options may be exercised at a point in the future to realize certain gains. Evaluating the payoff of these options may give an indication of the stability that such change may create.

Concluding remarks. The work is expected to form a genuine effort on understating the relation between changes in requirements and the architecture through strategic value-based reasoning. It aims to assist stakeholders' in strategic *"what if"* analysis, analyzing the strategic position of the enterprise- if the enterprise is highly centered on the software architecture (as it is the case in web-based service providers companies) and evaluating trade-offs between two or more candidate software architectures for stability. The intellectual framework is most critical; it demonstrates that with *value-based* reasoning we can improve our ability to evaluate for architectural stability and develop software systems that need to adapt to the inevitable evolving requirements.

6. References

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

2. Black, F., Scholes, M.: The Pricing of Options and Corporate Liabilities. Journal of Political economy (1973)

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

4. Boehm, B.: Software Engineering Economics**.** Prentice Hall (1981)

5. Favaro, J. M., Favaro, K.R., Favaro, P. F.: Value-Based Software Reuse Investment. Annals of Software Engineering. Vol. 5. (1998) 5 – 52

6. Finkelstein, A.: Architectural Stability, Some Preliminary Comments.http://www.cs.ucl.ac.uk/staff/a.finkelstein. (2000)

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

8. Myers, S. C.: Determinants of Corporate Borrowing. Journal of Financial Economics. Vol. 5(2). (1977) 147-175

9. Nuseibeh, B., Kramer, J., Finkelstein, A.: A Framework for Expressing the Relationships between Multiple View in Requirements Specification. Transactions of Software Engineering, Vol. 20(10).(1994) 760-773

10. Proceedings of the Workshops on Economics-Driven Software Engineering Research, EDSER 1 to 5. Workshops held in conjunction with the 21st through 25 th International Conference on Software Engineering (1999 – 2003)

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

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

13. 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)

14. Sullivan, K. J.: Software Design: The Options Approach. In: 2nd International Software Architecture Workshop, San Francisco, CA (1996) 15–18

15. Yoon, Y., Paul, K., Hwang, L.: Multiple Attribute Decision Making: An Introduction, Sage Publications (1995)