Valuation and Selection of OSS with Real Options

Androklis Mavridis

Product Manager, b-Open S.A
andy@b-open.gr

Abstract. The selection of Open Source Software (OSS) applications is a complex and difficult task. The evolving nature of OSS with constant updates, as well as the vast number of available projects hampers the selection process. Advancements in evaluation methods offer assistance in measuring various quality aspects, but do not examine the financial implications of risks and uncertainties imposed by the frequent updates/modifications and by the dynamics of the OSS communities. We perceive the OSS applications as assets capable of generating value upon selection. The objective is to discover the uncertainty factors affecting the overall value, to measure the quality evolution and finally to quantify the expected generated utility value of the OSS candidates.

Keywords: Open Source Selection, Real options analysis.

1 Introduction

In the Open Source Software (OSS) realm, the enormous number of available projects and applications hinders the selection process. The available quality evaluation approaches provide a time static quality assessment and thus, are not able to handle the evolving nature of open source projects, the resulting changes in quality and the risks associated with these changes. Moreover, these approaches, do not examine the financial implications of risks and uncertainties imposed by the frequent updates/modifications and by the dynamics of the OSS communities.

Quality-based economic valuation approaches dealing with design rational, have been introduced recently providing reasoning in IT investments where uncertainty and changes are difficult to predict [1]. The value-oriented notion of software artefacts has been treated with the Real Options Analysis (ROA) perceiving these artefacts as assets capable of generating value when properly managing the inherent risks, as "exercise options". Inspired by previous endeavours of valuations [2], [3], [4], [5], in this paper we propose a method assisting the OSS applications selection, by managing the applications' quality evolution, and valuate the selection as option, in a similar approach to financial options.

The basic concept of our work is that a software development decision under uncertainty, such as the selection of an OSS application is comparable to a financial derivative. In financial markets a derivative is a financial instrument whose value depends on, or derives from, the values of basic underlying assets [6]. In our context

L. Iliadis, M. Papazoglou, and K. Pohl (Eds.): CAiSE 2014 Workshops, LNBIP 178, pp. 44–52, 2014.

these assets are the candidate applications OSS applications. In its classic application, ROA uses the expected revenue notion as a foundation [7] and perceives the assets under examination as revenue generators based on the initial investment/acquiring cost and on the expected profits volatility.

Our method proposes the employment of Real Options Analysis to exploit the OSS quality evolution and calculate the system's generated value, aiming to maximise the benefits of the selection decision. Doing so, we inject extra intelligence to OSS selection process.

The paper structure follows. In section two we provide the related research work followed by a detailed presentation of our method in section three. Finally in section four we discuss the threats to validity followed by the concluded remarks in section five.

2 Background

The OSS application selection process includes searching, locating, and evaluating processes based on pre-defined criteria, and deciding upon applications [8]. However, only a limited number of empirical studies on selection of OSS applications have been performed [9]. Even if successful applications of normative selection methods for both COTS and OSS exist, such methods are rarely applied [10]. In [11] authors attributed the situational nature of OSS, where project specific properties constrain the selection outcome, as one of the suspending factors limiting the adoption of normative selection methods.

The limitation of the OSS evaluation can be justified partially to the subjective measurements provided by the communities and to the absence of well-structured and standardized quality models [12] [13] [14]. To overcome this obstacle, various research efforts have been introduced aiming to measure OSS quality employing different methods and focusing on different perspectives, ranging from source code quality analysis to community based metrics analysis, often with impressive results. Initial cumulative efforts resulted in a number of quality models, the most indicative and well known are summarised below.

2.1 Software Quality Evaluation

In our work we focus on the OSS quality and its evolution over project's generations. We take advantage of the aforementioned efforts and we argue that quality evolution is generated and affected by the dynamic eco-system nature of OSS. In order to account for the uncertainties imposed by the dynamics inherent in OSS projects we perceive the OSS selection process as an investment under uncertainty that can be approached and formulated with different options.

We make the hypothesis that OSS quality carries economic value in the form of real options, expressed through the right, but not the obligation, to select an OSS in the future, where the OSS to be selected is treated as a real asset. Real options analysis can help in discovering how OSS value changes over time, as quality changes through

project's releases. Our aim is to provide OSS stakeholders with a method able to translate in financial terms the impact of quality evolution to the OSS selection with respect to potentially uncertain future conditions.

2.2 Real Options

Real Options Analysis (ROA) is based on the analogy between investment opportunities and financial options. A real option is a right, but not an obligation, to make a decision for a certain cost within a specific time frame. A project is perceived as an option on the underlying cash flows with multiple associated investment strategies to be exercised if conditions are favourable. The big advancement is that ROA accommodates not only the value of the investment's expected revenues but also the future opportunities that flexibility creates. The inherent ability to react to market conditions increases the expected value of the investment by maintaining or improving the upside potential and limiting the downside loss. ROA overcomes the limitations of the traditional Discounted Cash Flow (DCF) method as it considers all possible price paths for the underlying project value and assumes a distribution for the underlying prices rather than a deterministic price assumption.

As option is an asset that provides its owner the right without a symmetric obligation to make an investment decision such as growth, exit, wait, and learning etc. If conditions 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. There are two option mechanisms, namely the *call* and *put*.

During the last decade Real Options theory found to be very attractive in IT investments [15], perceiving mainly the IT project from a holistic point of view, while in the same time efforts going a step further dealing with uncertainties inherent in software engineering practices were also introduced, such as in [16].

Based on these foundations, the central idea of our work is that a selection decision, such as *when to select a candidate OSS*, is analogous to a financial derivative expressed as a *call option,* where the owner (the person in charge of the OSS selection) has the right to decide when it is preferable to select the OSS candidate application. In financial markets a derivative is a financial instrument whose value depends on, or derives from, the values of basic underlying assets [17].

3 Proposed Method

Initially inspired by the work [18] where the author applies Real Options Analysis in COTS (commercial off-the- shelf) based software development scenario and encouraged by [19], highlighting the similarities of COTS and OSS in the context of OTS (Off-the-shelf) software development, we approach the application of Real Options in the context of OSS application selection/integration in a similar fashion to this analytically presented in [3].

Our method exploits the results of OSS quality assessments, as a basis for options based analysis. We keep the mechanisms as broad and flexible as possible in order to assure its applicability through various existing state of the art OSS quality assessment tools/frameworks and through various situational contexts. In authors' previous work [20] emphasis was given on the expected revenues each project could generate, while the application of Real Options was based on the simulated volatility of the most sensitive (the most affecting the Net Present Value) quality attributes. In this extended work we split the OSS applications selection process in three consecutive steps. The first step commences with the quality assessment of the candidate applications against the selected attributes. In the second we calculate the volatility of the scores of quality attributes in the form of (%) standard deviation, and finally in the third step we calculate the call options for each OSS candidate and we compare the results. In the following sections we analytically present these steps.

3.1 Quality Assessment

The majority of the available state of the art quality assessment methods can be used, for evaluating the desired quality attributes, as long as they provide weighted measurable metrics. The requirement in this step is to assess each candidate application against the same list of attributes. In this method we focus on fairly measurable attributes such as ***number of downloads per week***, ***number of major releases***, ***project maturity***, ***Community adoption***, etc. As the selection process of a candidate OSS can involve different stakeholders ranging from managers to software developers, different views of quality might be of concern. For example for an IT manager, factors influencing the evolution of the project, such as project downloads, From the other hand, for a software developer wishing to integrate an OSS to an existing system, the evolution of community adoption would provide more insights. For this reason above all, the proposed method should be constructed in a manner able to cope with different scenarios of use depending on the quality characteristics chosen.

3.2 Calculate Quality Volatilities

To perform the quality assessment and make inferences about the quality attributes, we employ available hierarchical quality models like the ones proposed in [21], [22], [23], [24]. The aim is to provide as much of automation as possible, through the analysis of source code and associated community metrics over the project's releases, and calculate the evolution of quality in the next future release in the form of standard deviation. We cope with situations where the information available for making judgement concerning quality evolution is collected by activities initiated/performed by the project's developer team and the involved community.

This information, like number of downloads or number of reported/fixed bugs can be extracted through the statistics services offered by OSS repositories (i.e. Sourceforge[1]) or more accurately through automatically extracting data from the project's web pages and especially the source code control system, in the form of CVS [23]. The aim is to analyse the offered statistical data, and provide experts' judgements about the evolution of the quality attributes [25]. For example by calculating the standard deviation –over the project's releases- a judgement on several quality attributes could be made. Though we acknowledge the fact that the analysis is characterised by subjectivity due to its highly dependability on experts opinion [13] [26], we believe that as long as it provides better understanding to stakeholders involved [27], and is able to infuse experts' tacit knowledge into measures associated with the achievement level of software quality attributes [28], it is useful and can be used supportively to provide extra reasoning in the context of OSS selection.

3.3 Calculating the Value of a Call Option

The investment opportunity can often be seen as a call option on the present value of the expected cash flows from the investment. The value of an option of the underlying asset depends on a number of variables:

a) Current Value of the Underlying Asset: As option generates value from the underlying asset, changes in the value of the underlying asset affects the value of the option, an increase in the value of the asset will increase the value of the call option.
b) Volatility of the Underlying Asset Value: The higher the variance in the value of the underlying asset, the greater the value of the option.
c) Dividends on the Underlying Asset: The value of the underlying asset can be expected to decrease if dividend payments are made on the asset during the life of the option. Hence, the value of a call option decreases when the size of the dividend payments increases.
d) Strike Price of Option: The value of the call option will decline as the strike price increases.
e) Time to Expiration: Options tend to become more valuable as towards expiration time, as the longer the time to expiration the more the value of the asset to move.
f) Riskless Interest Rate: As the buyer of an option pays the price of the option up front, an amortization of the cost is involved depending on the level of interest rate and the expiration time. Increases in the interest rate will increase the value of a call option.

We extend the Option valuation mechanism to accommodate the specificities and constraints of the OSS realm. To proceed with the Options analysis we need the following input:

Stock Price: The Stock Price for a candidate is expressed as the expected total cash flows or the utility value resulted from the successful adoption of the OSS for a given time frame.

[1] www.Sourceforge.net

Strike Price: This is the accumulation of the costs Total Cost of Ownership (TCO) for the transition to the OSS application for the given period until option expiration date [29]. Training/learning, software and Support costs are included here.

Time interval: This is the time until the opportunity disappears for making the selection.

Volatility: Represents the quality fluctuation – evolution over project releases.

Risk Free Rate: Assumed to be a known market value specific to the market domain [30].

Evolution Steps: The number of binomial steps to be calculated.

For the calculation of the Stock Price we adopt the notion of *utility* from the works of [31]. In a similar approach to this described in [3], we assume that this value is totally due to the utility provided by each attribute. Thus, the total Stock Price (Value) for a given candidate (*i*) is given by the equation (1):

$$\mathrm{SP}_i(\mathrm{t}) = \sum_j Vj(t) \tag{1}$$

Where V*j* is the utility attributed to the j^{th} quality attribute.

The objective here is to calculate the *Total Utility Value* (the total utility expected to be obtained until the expiration date) and the *Call Option Value* (the value of the option). In our context this translates to: *What is the total expected utility gain in the given time interval until selection and what is the option value in waiting until the time for making the selection expires?* To calculate the option value for each candidate we apply binomial options pricing model [32].

We build for each candidate application two binomial lattices based on the American call option fashion, which dictates that the option (the selection) can be exercised at any given time until the expiration. The first lattice calculates the expected total OSS value, while the second calculates the option value, the amount by which a call option is in the money (utility), in other words "how far" it is profitable to wait until making the selection of an OSS candidate.

To clarify the mechanisms of the binomial pricing model, lets consider an application whose value is initially So and an option on the application's selection whose current value is f. Suppose that the option lasts for time T (time to select). During the life of the option, the application value (due to its quality) can either move up i.e. the project community enhances the quality of the given application, from So to a new level, Su or down, i.e the community stops supporting the application and thus its quality value decreases, to a new level Sd.

The proportional increment in the application's value when there is an up movement is u-1; the proportional decrement when there is a down movement is 1-d. If the application value moves up to Su, the payoff from the option is assumed to be fu; if the stock price moves down to Sd, the payoff from the option is assumed to be fd.

From this point we calculate the binomial options value by applying the following equation (2):

$$\text{Value} = ([p * \text{Option up} + (1-p) * \text{Option down}] * \exp(-r * \Delta t)), \quad (2)$$

where r is the risk free rate corresponding to the life of the option, p is the probability of an up movement factor,

$$p = \frac{e^{(r-q)\Delta t} - d}{u - d} \quad (3), \text{ with } u = e^{\sigma\sqrt{t}} \quad (4)$$

the up movement factor, d=1/u the down movement factor, and σ expressing in % the percentage of the quality volatility. Hence, similarly to equation (1), the total option value TOV for a given candidate *(i)* application will be given by the following equation (5):

$$\text{TOV}_i(t) = \sum_j OVj(t) \quad (5)$$

Where *OVj* is the option value attributed to the j^{th} quality attribute.

It is to be noted here that we limit our focus on a simple selection process and do not consider in this work staged developments, upgrades and migrations of OSS applications which require a more in depth analysis. These cannot be dealt adequately with the simple call option mechanism but rather with combined option schemes. Nevertheless the binomial pricing model can effectively handle the simple selection process, framed as a call option regardless of the number of candidates.

4 Applicability

Several factors need to be taken into consideration when applying the proposed quality-valuation method to realistic settings. Our method is based on the premise that calculation of volatility is always possible regardless of the scenario of use and the information at hand, as it is suggested that a fair amount of releases should be examined in order to obtain enough insights of quality's evolution over the project's life time. This requirement limits the method's applicability only to mature and established projects.

Another limitation is the assumption that it is feasible to calculate the expected revenues and the costs attributed to qualities examined for the OSS application under analysis. Even though, the employment of the notion of "utility" provides a handy workaround to contexts with inadequate financial historical data, it remains a difficult task and should be handled by experienced project managers.

The calculated volatility strongly depends on the hierarchical quality model used. Different models could produce different estimates of volatility. Additionally, it is not always possible to efficiently perform structural analysis on the source code mainly due to tools' constraints to code size.

Our approach currently does not perform any kind of analysis on the correlations that naturally exist between quality attributes. The addition of a mechanism addressing the tradeoffs at quality level would increase the validity of the measured volatility limiting at the same the experts' subjectivity.

Lastly even though the mechanisms and the variables required performing the binomial lattice practices are simple enough, clear instructions on the methodology and the associated tools should be given prior to analysis.

5 Discussion - Conclusions

We have presented a method assisting the OSS applications selection process. Assumptions were made in order to simplify our approach and to increase its applicability. We proposed a blend of evaluation methodologies and valuation analysis able to give insights to risks anticipated due to future uncertainties on the qualities of OSS applications.

Our method provides an alternative view to the selection process in the uncertain Open Source Software realm. It is capable of shedding light to the mechanisms introducing uncertainty and manages this uncertainty to maximise the profits of our decisions. Nevertheless, we do not suggest that this proposed OSS application selection approach should be treated as panacea. Intuition and other factors should be taken always into account in the OSS selection process.

Perceiving the selection decision as call option bearing monetary value introduces a link between the worlds of software technology and finance and thus, the results can easily be exploited by both developers and managers.

References

[1] Shaw, M., Arora, A., Butler, S., Poladian, V., Scaffidi, C.: In search of a unified theory for early pre-dictive design evaluation for software, Technical Reports CMU-ISRI-05-114, Carnegie Mellon University (2005)

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

[3] Ozkaya, I., Kazman, R., Klein, M.: Quality-Attribute Based Economic Valuation of Architectural Patterns, Technical Report CMU/SEI CMU/SEI-2007-TR-003 (2007)

[4] Meinhausen, N., Hambly, B.M.: Monte Carlo Methods for the Valuation of Multiple-Exercise Options. Mathematical Finance 14(4), 557–583 (2004)

[5] Erdogmus, H.: Valuation of Learning Options in Software Development under Private and Market Risk. The Engineering Economist 47(3), 308–353 (2002)

[6] Hull, J.C.: Options, Futures, and Other Derivatives. Pearson Prentice Hall (2006)

[7] Amram, M., Kulatilaka, N.: Real Options: Managing Strategic Investment in an Uncertain World. Harvard Business School Press, Boston (1999)

[8] Mandanmohan, T.M., De' Rahul: Open Source Reuse in Commercial Firms. IEEE Software, No-vember-December 21(6), 62–69 (2004)

[9] Li, J., Conradi, R., Bunse, C., Torchiano, M., Slyngstad, O.P.N., Morisio, M.: Development with off-the-shelf applications: 10 facts. IEEE Software 26(2), 2–9 (2009)

[10] Torchiano, M., Morisio, M.: Overlooked Aspects of COTS-Based Development. IEEE Soft-ware 21(2), 88–93 (2004)

[11] Hauge, O.: An Empirical Study on Selection of Open Source Software - Preliminary Results. In: FLOSS 2009, Vancouver, Canada, May 18 (2009)

[12] Sung, W.J., Kim, J.H., Rhew, S.Y.: A Quality Model for Open Source Software Selection. In: Sixth International Conference on Advanced Language Processing and Web Information Technology, ALPIT 2007 (August 2007)

[13] del Bianco, V., Lavazza, L., Morasca, S., Taibi, D., Tosi, D.: An investigation of the users' perception of OSS quality. In: Ågerfalk, P., Boldyreff, C., González-Barahona, J.M., Madey, G.R., Noll, J. (eds.) OSS 2010. IFIP AICT, vol. 319, pp. 15–28. Springer, Heidelberg (2010)
[14] Lincke, R., Lundberg, J., Löwe, W.: Comparing Software Metrics Tools. In: ISSTA 2008, Seattle, Washington, USA, July 20-24 (2008)
[15] Schwartz, S., Trigeorgis, L.: Real options and Investment Under Uncertainty: Classical Readings and Recent Contributions. MIT Press, Cambridge (2000)
[16] Erdogmus, H.: Valuation of Complex Options in Software Development. In: First International Workshop on Economics-Driven Software Engineering Research, EDSER-1, Los Angeles, May 17 (1999)
[17] Hull, J.C.: Options, Futures, and Other Derivatives. Pearson Prentice Hall (2006)
[18] Erdogmus, H.: Management of License Cost Uncertainty in Software Development: A Real Options Approach. In: 5th Annual Conference on Real Options: Theory Meets Practice, Los Angeles (2001)
[19] Jingyue, L., Conradi, R., Bunse, C., Torchiano, M., Slyngstad, O., Morisio, M.: Development with Off-the-Shelf Components: 10 Facts. IEEE Software 26(2), 80–87 (2009), doi:10.1109/MS.2009.33
[20] Mavridis, A., Stamelos, I.: Options as Tool Enhancing Rationale of OSS Components Selection. In: IEEE-DEST 2009, Istanbul, Turkey, May 31-June 2 (2009)
[21] ISO/IEC TR 9126-(1-4). Software engineering – Product quality. International Standardisation Organisation (2003-2004)
[22] McCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality, Nat'l Tech. Information Service, volumes 1,2 and 3, AS/A-049-014/015/055. Springfield, Vancouver (1977)
[23] Dormey, G.R.: A Model for Software Product Quality. IEEE Transactions on Software Engineering 21(2), 146–162 (1995)
[24] Bansiya, J., Davis, C.: A Hierarchical Model for Object-Oriented Design Quality Assessment. IEEE Transaction on Software Engineering 28(1), 4–17 (2002)
[25] Moses, J.: Benchmarking quality measurement. Software Quality Journal 15(4), 449–462, doi: 10.1007/s11219-007-9025-4
[26] Moses, J., Farrow, M.: Tests for consistent measurement of external subjective software quality attributes. Empirical Software Engineering 13(3), 261–287, doi:10.1007/s10664-007-9058-0
[27] Wohlin, C., Andrews, A.A.: Assessing Project Success Using Subjective Evaluation Factors. Software Quality Journal 9(1), 43–70, doi:10.1023/A:1016673203332
[28] Rosqvist, T., Koskela, M., Harju, H.: Software Quality Evaluation Based on Expert Judgement. Software Quality Journal 11(1), 39–55, doi:10.1023/A:1023741528816
[29] Russo, B., Braghin, C., Gasperi, P., Sillitti, A., Succi, G.: Defining the Total Cost of Ownership for the Transition to Open Source Systems. In: Proceedings of the First International Conference on Open Source Systems, Genova, July 11-15 (2005)
[30] Hull, J.C.: Options, Futures, and Other Derivatives, 6th edn. Prentice Hall, Upper Saddle River (2006)
[31] Kazman, R., Asundi, J., Klein, M.: Making Architecture Design Decisions: An Economic Approach Technical Report CMU/SEI-2002-TR-035 ESC-TR-2002-035
[32] Copeland, T., Antikarov, V.: Real Options: A Practitioner's Guide. TEXERE, New York (2001)