Applying ArchOptions to Value the Payoff of Refactoring

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

Abstract

ArchOptions is a real-options based model that we have proposed [1] to value the flexibility of software architectures in response to future changes in requirements. In this paper, we build on ArchOptions to devise an options-based model, which values the architectural flexibility that results from a refactoring exercise. This value assists in understanding the payoff of investing in refactoring: if the refactored system results in an architecture that is more flexible, such that the expected added value (in the form of options) due to the enhanced flexibility outweighs the cost of investing in this exercise, then refactoring is said to payoff. We apply our model to a refactoring case study from the literature.

1. Introduction

As software is enhanced, modified, or adapted to new requirements, the software becomes more complex and drifts away from its original design. To reduce complexity, there is a need for techniques that incrementally improves the internal software quality. The research domain that addresses this problem is referred to as restructuring, or in the case of object-oriented and agile development, as refactoring [12]. In the context of software evolution, restructuring and refactoring are used to improve the quality of the software such as extensibility, modularity, reusability, complexity, and efficiency. In refactoring, the key idea is to redistribute classes, variables, and methods across the class hierarchy in order to facilitate future adaptations and extensions. This in turn will result in a modified structure (compared to the original) with different qualitative measures and value potentials.

Numerical measures can be used before applying a refactoring, to measure the (internal or external) quality of software, or after the refactoring, to measure improvements of the quality. For example, Simon et al. use distance-based cohesion metrics to detect where in a given piece of software there is a need for refactoring [16]. Kataoka et al. use coupling metrics to evaluate the effect of refactoring on maintainability [11]. Coleman et al. use a polynomial of multiple measures to define a maintainability index by which the effect of refactoring can be evaluated [5]. However, little has been done on understanding the economics of refactoring. For example, when is it cost-effective to invest in a refactoring exercise? How can we value the payoff due to refactoring, prior to investing in such an exercise? How can we reason about this payoff in connection with changes in the structure and at correspondingly higher level of abstractions than code? These questions translate into a need for economic models that quantify the payoffs of refactoring. Such models inform the decision in investing in refactoring through a tradeoff between the up-front cost and the expected added value to the system as a result. The added value may be strategic or operational; it may take the form of expected savings in maintenance and/or returns due to the enhancement of some qualities such as maintainability, extensibility, modularity, reusability, or efficiency. A characteristic of these benefits, whether strategic or operational, is that their payoffs are uncertain and may not be immediate.

Little work has been done to understand the economics of restructuring and refactoring. Notable effort includes [18, 20]. Leitch and Stroulia [18] have proposed a framework for predicting the return on investment (ROI) for a planned refactoring using cost-benefit analysis. In their seminal work, Sullivan et al. [20] have shown how options thinking can be used to value software design decisions including restructuring. They have developed an option model that borrows from decision analysis to value the payoff of the decision to restructure legacy systems and its optimal exercise time.

In this paper, we propose an options-based approach to value the architectural flexibility that results from a refactoring exercise. We build on the ArchOptions model [1], which values the flexibility of software architectures, relative to likely changes in requirements. We assume that refactoring a system could enhance the flexibility of the system's structure/architecture. This incurs an upfront cost to investment. It is worthwhile investing in refactoring, if the refactored system could lead to an architecture/structure that is flexible enough and adds a value to system following this exercise. We use the expected benefits due to changes in the structure, as a way to value the payoff of refactoring. As we assume that the added value is attributed to flexibility, the decision to refactor is driven by the motivation to maximize the payoffs in the adapted architectural flexibility that results from refactoring. We use savings in maintenance cost relative to some likely future changes as a way to quantity the added value. We apply the model to a refactoring case study from the literature.

The use of strategic *flexibility* to value software design decisions is not new. It has been explored in, for example, [2,7,8,14,19,20]. However, the use of the resulting architectural flexibility and its value as metric to inform the decision of investing in refactoring is in the scope of the work. The paper is further structured as follows. Section 2

presents an option model to value the payoff of refactoring. Section 3 evaluates the model. Section 4 concludes the paper and indicates further work.

2. Valuing the Payoff of Refactoring

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 that flexibility creates [7]. These include growth, abandonment or exit, delay, and learning options. An option is an asset that provides it 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 [15]. If conditions favourable 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.

We derive a real option-based model from [3], referred to as ArchOptions. In ArchOptions, we value the growth options of an architecture relative to some future changes, as a way for understanding the architectural flexibility/stability. A growth option is a real option to expand with strategic importance [13]. Growth options are common in all infrastructure-based (as it is the case with software architectures) or strategic industries with multiple-product generations or applications [15]. In the architectural context, growth options 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 too expensive to pursue and opportunities may be lost.

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 *(1)* accounting for V and both the expected value and exercise cost to accommodate i_i, for $i \leq n$. Valuing the expectation E of expression *(1)* uses the assumptions of [3] and detailed in [1]. We assume that the interest rate is zero for the simplicity of exposition.

$$V + \sum_{i=0}^{n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)] \qquad (1)$$

The model has the prospect of valuing the architectural flexibility and its value potentials due to various types of changes. These may be preventive, adaptive, or perfective [10]. Refactoring, a preventive change, can be seen as an investment to embed flexibility. The objective is to "clear up" much of the system degraded structure and enhance its upside potentials by making it more accommodating for future changes. In this context, refactoring can be seen as an investment to purchase growth options that enhance the upside potentials of the structure, paying an upfront cost I_e, which corresponds to the cost of refactoring. We build on the ArchOptions model to value whether it is worthwhile to invest into refactoring, as shown in *(2):*

$$V - I_e + \sum_{i=0}^{n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)] \qquad (2)$$

Let us assume that S_1 is a structure of the software obtained by refactoring S_0. We assume that refactoring is an economical choice, if it adds value to S_1 relative to S_0. We attribute the added value to the enhanced flexibility of S_1 over S_0. If we are considering savings in maintenance as a criteria for understanding the value added to the system, then future changes in requirements following refactoring will tell us how valuable S_1 is relative to S_0. But the added value due to refactoring is uncertain, as the demand on future changes are uncertain. This makes refactoring a good candidate to reason using option "thinking".

The decision to refactor has to be guided by the expected payoff in $(- I_e + \sum_{i=1\ldots n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0])_{S1}$ relative to that of S_0. That is, if $(- I_e + \sum_{i=1\ldots n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)]_{S1} > \sum_{i=1\ldots n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)]_{S0})$ for some likely changes, then it is worth investing in such an exercise, as the investment in refactoring is likely to generate more growth options for S_1 than for S_0. As we assume that $x_i V$ is the expected saving in S_1 over S_0 due to refactoring, it is reasonable to consider that if $(- I_e + \sum_{i=1\ldots n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)]_{S1} >= 0)$, then investing in refactoring is said to payoff. An optimal payoff could be when the option value (i.e., $\sum_{i=1\ldots n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)]$ approaches the maximum relative to some changes in requirements, indicating an optimal payoff in an investment in flexibility provided that $(- I_e + \sum_{i=1\ldots n} \mathrm{E}\,[\max\,(x_i V - C_{ei}, 0)]_{S1} >= 0)$. The analyst may conduct sensitivity analysis to manipulate the model variables and analyze when such a state is likely to occur.

For a change in requirement k, if the $(- I_e + \mathrm{E}\,[\max\,(x_k V - C_{ek}, 0)]) < 0$, then refactoring is not likely to payoff as the flexibility of the architecture in response to the change is not likely to add a value if the change need to be exercised. Two interpretations might be possible: (i) the architecture is overly flexible in the sense that its response to the change(s) 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 the changes. In other words, the degree of flexibility provided is much more than the flexibil-

ity demanded for the change. This case has the prospect in providing an insight on how much do we need to invest in refactoring 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 to the changes.

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)

The options model *(2)* requires the estimation of several parameters. Most importantly are x_iV, I_e, and C_{ei}.

Estimating C_{ei}, I_e. 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 [4]). Another approach is to build on architectural level dependency analysis (e.g., [14]) research to extract cost estimates of accommodating i_i, guided by some structural criteria.

Capturing and estimating x_iV. 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.

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 [15]. Real options valuation based on arbitrage-based pricing techniques determines the value of an asset in question in span of the market value using a correlated *twin asset* [15]. The twin asset is an asset that has the same risks the asset in question will have when the investment has been completed [15]. 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 [15]. Real options valuation (based on arbitrage) focuses on market value and uses 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 [15]. For example, some aspects of the architectural responsiveness to the change can be justified in terms of the directly observable cash flows linked to future operational benefits or the market- making it easy to use the rate of return to value the options. However, many others aspects may not be directly observable through cash flows. Yet, their contribution to the added value is crucial. If the analyst(s) relies on experience and judgment in his/her estimation, the estimates tend to be subjective but could make an implicit use of market information. However, back-of-the-envelope calculations, which are based on value estimates (rather than on market value) are yet revealing [19]. We note that it remains an open challenge to strongly justify precise estimates for real options in software [20]. As a compromise, estimating x_iV requires a comprehensive solution that is flexible to incorporate multiple valuation techniques; some with subjective estimates and others based on market data, when available. The problem of how to guide the valuation and introduce discipline in this setting, we term as the *multiple perspectives valuation problem*. As the added value may be relative to the market; on one or more technical aspects of the system; and/or relative to the organization, the solution may be through a valuation framework that captures the added value - of the architectural potential of the change- from different perspectives. The purpose is to reach a comprehensive value of options from the different perspectives. Also, the aim is to promote flexibility through incorporating both subjective estimates (may implicitly use market information) and/or explicit market value (when available). As the architecture is the artefact that facilitates both technical and market reasoning, such an approach seems to be viable. Addressing this problem and its solution is outside the scope of this paper.

3. Case Study

The objective of the study is to empirically simulate the applicability of the model, and validate its interpretations. We summarize the simulation rationale as follows: (a) refactor and observe its effect on the flexibility of the structure (b) observe the responsiveness of the structure to some random changes in requirements following action (a); (c) quantify flexibility relative to likely future changes as a way for understanding the payoff of refactoring. Particularly, we seek an understanding for the following: Are the model interpretations valid? When does refactoring, as an adapted flexibility, add to the system a value? How worth-

while is it investing in such an exercise while not sacrificing much of our resources?

To achieve the simulation rationale, we use the refactoring case study of a traffic light system published in [18], 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- are 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. To capture and estimate x_iV, we restrict the valuation to the development perspective for space limitation. We use the expected savings in development effort for likely futures changes due to refactoring. When necessary, we use \$2000 for man-month to cast the effort into cost. We show how we have estimated the parameters:

Estimating (I_e). Table 3 reports the refactoring effort (man-month), cost (\$), and schedule (month) based on the refactoring plan presented in [18]. Table 3 provides three values: optimistic, likely, and pessimistic for each parameter. All are calculated using COCOMO II.

Capturing and 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.

Calculating the volatility (σ). 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. Volatility stands for the "fluctuation" in the value of the estimated x_iV. Intuitively, it "aggregates" the "potential" values of the structure in response to the change(s). We take the percentage of the standard deviation of the three x_iVs estimates-the optimistic, likely, and pessimistic values- to calculate σ.

Exercise time (t) and free risk interest rate(r). 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 the same as 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 random likely changes (Development Perspective)

	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 investing in refactoring, while not sacrificing much of our resources?

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, refactoring has not "pulled" the options for one change. 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 unutilised 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. We repeat the above experiment, but stressing S_1 with two, three, four, and then five 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, refactoring 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, refactoring is likely to add to the system a value, if ten or more changes need to be exercised during the next three years.

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

Changes	σ	$x_i V$			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

4. Conclusions and Future Work

The observations verify that the model interpretations are reasonable. We have appealed to the use of maintenance savings as a way to value the options due to refactoring. Needless to say, the valuation could have incorporated other valuation points of view (e.g., extensibility, reusability, efficiency etc.) to value the options due to refactoring on other qualities and/or the market (if relevant). The aim is to have a comprehensive value of options from different perspectives. Future work entails detailing how such a valuation points of view can be used to capture, value, and reconcile the options from different perspectives.

Experts may question our use of [3] to options valuation, as the satisfaction of the spanning condition may be doubtful. We argue that valuation based on man-month does implicitly hold market-based data and is done in relation with the market. Alternatively, we could have cast the options model to use different options valuation (e.g., [6]). However, the application of [3] offers a closed and an easy-to-compute solution, for it assumes that $x_i V$ is lognormaly distributed, not requiring $x_i V$ to be probability-adjusted for rise and drop in value, as when compared to [6]. We have not explicitly modeled the uncertainty of future changes and their corresponding time value. We will investigate this in the future. Following the argument of [19], 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.

5. Acknowledgements

The authors would like to thank Hakan Erdogmus, Kevin Sullivan, and other anonymous reviewers for their very valuable feedback that has significantly improved the content and the presentation of the paper. All errors of fact or interpretation remain the sole responsibility of the authors.

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] Baldwin, C. Y., Clark, K. B.: Design Rules - The Power of Modularity. MIT Press (2001)

[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] Coleman, D., Arnold, P., Bdoff, S., Gilchrist, H., Hayes, F. and Jeremaes P., Object-Oriented Development: The Fusion Method. Prentice Hall (1994)

[6] Cox, J., Ross, S., Rubinstein, M.: Option Pricing: A Simplified Approach. Journal of Financial Economics. Vol.7 (3). (1979) 229-264

[7] 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)

[8] Erdogmus, H.: Value of Commercial Software Development under Technology Risk. The Financier, vol. 7. (2000)

[9] Hull, J. C.: Options, Futures, and Other Derivative Security. Third edition, Prentice-Hall (1997)

[10] IEEE Standard 610.12: Glossary of Software Engineering Terminology. In: Software Engineering Standards Collection, IEEE CS Press (1993)

[11] Kataoka, Y., Imai, T., Andou, H., and Fukaya, T.: A Quantitative Evaluation of Maintainability Enhancement by Refactoring. In: Proc. Int'l Conf. Software Maintenance, pp. 576-585, (2002)

[12] Mens, T., Tourwe, T.: A Survey of Software Refactoring. In: IEEE Transactions on Software Engineering. Vol. 30(2)(2004)

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

[14] Port, D., Huang, L., and Boehm, B: Strategic Architectural Flexibility. In: 4th International Workshop on Economics-Driven Software Engineering Research (EDSER), (2002), 32-37

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

[16] Simon, F. Steinbru, F. ckner, and Lewerentz, C. : Metrics Based Refactoring, Proc. European Conf. Software Maintenance and Reeng. pp. 30-38 (2001)

[17] Stafford, J. A., Wolf, A. W.: Architecture-Level Dependence Analysis for Software System. International Journal of Software Engineering and Knowledge Engineering. Vol. 11(4) (2001) 431-453

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

[19] 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

[20] 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)