Towards an Economics Model for Software Development in Support of B2C Services

Brendan Tansey[1] and Eleni Stroulia[1]

Computer Science Department, University of Alberta, Edmonton, AB, T6G 2E8, Canada {tansey,stroulia}@cs.ualberta.ca

Abstract. Economic models are critical for business success because they can be used to determine the costs and benefits associated with making a business decision. As the economy moves away from traditional goods and towards delivering services, the need for an accurate economic model for evaluating software-development decisions of service-delivering businesses is paramount. More specifically, the research problem is to develop an economic model for the development, maintenance, and evolution of software systems in support of B2C services. In this paper, we describe our work on an initial economics-modeling framework for applications, following the service-oriented architecture style. We present methods for calculating both the cost of developing a new service and the value generated by introducing the service to the market. The conjunction of these two concepts enables the estimation of a business's ROI when new software development is required to deliver a new service. An example study demonstrates the concepts described herein.

1 Introduction

The primary goal of any business is to make money [23]. To that end, the business must plan its future actions based on all relevant information. If this information is not readily available, it may potentially be inferred or predicted based on models. Among the most important models used for this purpose are economics models, which can help the company predict costs and revenues and can, in turn, predict the future financial state of the company. Having the ability to make accurate financial predictions is crucial during certain points in the lifecycle of the company, such as during the inevitable phases of expansion; predicting the costs and benefits of various expansion options can help determine which option to pursue.

Consider, for example, the fictional online book seller Amazin.com that is looking forward to an expansion of their services. Based on their available resources, they have determined that they can expand in only one direction at the current time, and their expansion options are limited to two strategies: they can either enter into the online music market by selling CDs along with books, or they can partner with another company and sell their products for a per-transaction fee. Each of these strategies has potential benefits, but at what cost? Can the additional revenue created by each option exceed the cost of implementing the option? Which option provides the highest return-on-investment (ROI)? An economic model can help answer these questions[1].

Traditionally, economic models for software development have focused on the prediction and analysis of the costs associated with the development of an application [7]. The implicit assumption is that the value of a software product is known from the outset, based on a contract between the software company and the client. As the value is constant, profit can be maximized simply by reducing operating and development costs. However, much development is currently evolutionary, based on a common framework or on pre-built commercial off-the-shelf (COTS) components [9]. This includes service-oriented applications[2] that are comprised of a collection of simpler services communicating together to create a larger, more functional, service. As such, development of service-oriented applications (SOAs) falls outside the scope of conventional software economic models. This increasingly prevalent software-development lifecycle model necessitates the creation of an appropriate economic model.

An economic model for SOAs should help address two basic questions: 1) What is the cost of the SOA project? –and– 2) What is its value? Both of these questions have complex and multifaceted answers, and there are many contributors to both cost and value. This paper covers only the tip of the proverbial iceberg, in that it concentrates on only the basic cost of creating a service, and a major contributor to value that is seldom taken into account. In effect, this paper will serve as a basis for further cost/value research in the area of service-oriented computing. This in mind, based on the answers to the above questions, the ROI of the project can be determined, which is at the crux of making strategic decisions, such as those detailed in the Amazin expansion decision above.

Traditional software cost models such as COCOMOII [7] can be used to address the first question. There exists, however, an interesting difference between standard use of these models and what is required in this case: since SOAs generally utilize a reuse-based development process, the portion of the system that has already been developed must be taken into consideration. In effect, the cost of creating the system must be a function of the complexity of the transformation from a collection of pre-built components to a final product.

The answer to the question of value estimation is more elusive. There is value in the immediate financial gains produced by marketing a service; in addition, a non-tangible future value also exists because of the potential services that could "easily" be built using the constituents of the current service. The flexibility and reusability of a service composition [17] can be modeled with *real option theory* [9]. An option, from a business perspective, gives its owner the right to

[1] For the curious, this motivation is inspired by Amazon.com's actual expansion into the CD market in June 1998 [2].

[2] The term "service" in this paper is used in two senses: it denotes (a) a business interaction between an organization and a customer and (b) a software component. Note that the two terms are related: business that deliver services to customers through web-based applications are increasingly likely to adopt the service-oriented architecture style for the development of their systems since that enables the flexible adaptation of these systems in response to their strategic decisions.

buy or sell an asset at a fixed price on a variable market for a limited time. In software-engineering terms, when systems are composed of small interlocking components (as is the case with SOAs), there exists the option to replace any component with an upgrade, which would incur some cost and result in the benefit of an increase in the value of the service delivered by the system.

Real-option analysis, in conjunction with traditional service valuation market research, enables the determination of the value of an SOA project. Combined with the cost analysis of developing and maintaining the system, a complete service-based economic model can be developed. The following sections will further expand on these points. The Methodology section will detail the methodology used in determining the costs and values associated with the economic model, an Illustrative Example will demonstrate the usefulness of the model, and the Related Research section will discuss the related work in the fields of cost and value determination.

2 Methodology

The methodology of this study is composed of two parts: the determination of cost using COCOMOII, and the determination of value using the Net-Present Value model integrated with Real-Options Theory.

2.1 Cost Determination

A number of COCOMOII parameters must be calibrated for a specific software development environment. COCOMOII.2000 comes with a set of defaults in place, determined through testing the model on 161 sample projects. While these parameters are adequate as a starting point, they will likely produce less accurate results than if the model were calibrated to match a specific development environment.

COCOMOII is essentially a collection of models that can be used for predicting the effort involved in multiple stages of the product lifecycle using various approaches to development. One of the included models deals with a software project based on the reuse of previously created code, which fits well with the reuse-based development environment of web services. The reuse model determines the difference between the existing reusable code and the envisioned system in terms of logical lines of code to be written. This difference can then be converted to a measure of effort required to make this alteration. The intuition behind this model is that the complexity of the change is a function of the distance from the old code pieces to the new service, and the cost of performing the change is a function of this complexity.

In this section, we briefly present the equations, parameters, and variables of COCOMOII.2000 that are relevant to our work. The details of the model, its parameters, their default values, and how they should be calibrated can be found in the COCOMOII.2000 model manual [5].

The steps involved in the effort calculation are shown in (1), (2), and (3).

$$Equivalent\_KSLOC = Adapted\_KSLOC * \left(1 - \frac{AT}{100}\right) * AAM \tag{1}$$

$$\text{Where: } AAM = \begin{cases} \frac{[AA+AAF(1+(0.02*SU*UNFM))]}{100}, & \text{for } AAF \leq 50 \\ \frac{[AA+AAF+(SU*UNFM)]}{100}, & \text{for } AAF > 50 \end{cases} \tag{2}$$

$$AAF = (0.4 * DM) + (0.3 * CM) + (0.3 * IM) \tag{3}$$

The model parameters, mentioned in the above equations, are as follows. AAM is the Adaptation Adjustment Modifier, and AAF is the Adaptation Adjustment Factor; they represent the effort of fitting the adapted code to an existing product, and the relative size of the modification, respectively. $Equivalent\_KSLOC$ ($EKSLOC$) is the number of *logical* source lines of code (SLOC), divided by 1000, required to create the new service. This parameter is a measure of effort, not an actual count of code lines to be written. It reflects the effort that the "translation" of the existing service to the new one will require. AT is the percentage of code that can be translated using automatic translation tools from the existing system to the new one, thereby not requiring any effort. AA is the assessment and assimilation increment, which is a measure of effort required to determine if the old code base is suitable for the new project. SU is the level of software understanding increment that models the quality and complexity of the old code base. $UNFM$ is the familiarity of the programming team with the code that they are modifying. AA, SU, and $UNFM$ are project-specific parameters that need to be set using the calibration guidelines in the COCOMOII.2000 model manual [5].

This leaves four parameters that need to be calculated before the model can be used to predict effort: $Adapted\_KSLOC$, DM, CM, and IM. These parameters are based directly on the project in question. $Adapted\_KSLOC$ is the number of logical source lines of code that need to be modified in order to adapt the existing project to the new one. Physical lines of code can be translated to logical lines of code (*i.e.,* the number of code statements) through the use of an application such as CodeCount [26] or LOCC [8]. DM represents the percent of the original design that is modified and is a fairly subjective value. CM is the percent of the code base that is modified. IM is the percent of integration into the new code base that is required required (*i.e.,* through automatically translated, manually translated and newly written code), and measures the percentage of the code that must be retested to ensure proper integration of the reused code. If CM and DM are both equal to zero, then SU is also set to zero. This is because if no changes to the code are necessary, then programmer understanding of the code is irrelevant.

In order to translate $EKSLOC$ into a useful effort measure, the COCOMOII Post-Architecture Model can be used, which is shown in (4), (5), and (6).

$$PM = A * Size^{E} * \prod_{i=1}^{17} EM_i \tag{4}$$

$$E = B + 0.01 * \sum_{j=1}^{5} SF_j \tag{5}$$

$$Size = \left(1 + \frac{REVL}{100}\right) * (New\_KSLOC + EKSLOC) \tag{6}$$

In this model, PM is the amount of effort required for the entire translation process. It is measured in Person-Months of programmer time required to make the alteration. A is the effort coefficient, which should be calibrated to reflect the development environment. There are 17 EM_is, Effort Multipliers, to control the rate at which various factors of the software development process affect the effort required to perform a change. There are 5 SF_js, Scale Factors, which determine the effect of economies or diseconomies of scale on the software project. B is the scaling base-exponent, which should be calibrated based on the specific development environment. $REVL$ is the requirements evolution and volatility level, which is the percentage of code that is discarded from the original module during the modification. New_KSLOC is the amount of new code that is required for the incorporation of the reusable code into the final product.

The time required to perform a change can also be calculated using COCOMOII. The equation for this is as follows in (7) and (8).

$$TDEV = \left[C * (PM_{NS})^F\right] * \frac{SCED\%}{100} \tag{7}$$

$$F = (D + 0.2 * [E - B]) \tag{8}$$

$TDEV$ is the time required to perform the required changes, measured in calendar months. B, C, D, and E are environment-specific coefficients. E corresponds to the value for E from the effort calculation; B, C, and D are unique to the $TDEV$ equation and should be calibrated. PM_{NS} is the nominal effort involved in making the required changed; it is measured in person-months, and does not take the $SCED$ parameter into account. $SCED$ is the required schedule compression: it ranges from 1.43 for an accelerated schedule to 1.00 for a drawn-out schedule.

Once the effort, measured in person-months, has been determined, it can be directly translated into a monetary amount by simply multiplying it by the monthly salary of the programmers involved in the project. This identical procedure can be used to determine the cost of maintaining a service as well, as the pre-maintenance version of the service can correspond to the reusable version, and the post-maintenance version can correspond to the final product in the above model. The difference between an initial reuse model and the maintenance model is that the scope of the change is usually fairly limited during maintenance, but this is not necessarily true for strict reuse; this implies that the values for $Adapted\_KSLOC$, DM, CM, and IM will be lower during maintenance.

There is one point of contention to the use of this model for web services: the usage of SLOCs as a relevant metric when computing effort involved in a change has recently been questioned. Basing an effort calculation on the change in an interface specification may be more relevant than differencing SLOC counts for environments where web service specification languages such as WSDL and BPEL are used. This alternate approach can be undertaken by basing the effort calculations on a measure of the change in Function Points (FPs), which are the inputs and outputs to a system, shared files or interfaces, queries, and logical collections of user data. COCOMOII supports the use of FPs, in that the FP values can be converted to an equivalent number of SLOCs for use in the prediction models. Such conversion ratios do not yet exist for WSDL and BPEL. In principle, one should calibrate this FP-to-SLOC ratio in the context their own service-oriented development environment to obtain an accurate estimate of effort. This can be done by using the effort and lines of code written in past projects as training points for linear regression, formulated in such a way that the weight on the model corresponds to the desired FP-to-SLOC ratio.

2.2 Value Determination

To estimate the value of an envisioned service, developed as an extension of a current one, the standard corporate valuation tool of Net Present Value (NPV) [28] can be utilized. NPV determines the future value of a product or service expressed using the current value of money, which makes analysis of projects with different time frames directly comparable. In order to express future money in current terms, the future value is 'discounted' by a certain rate, which models the opportunity cost of the investment in the product [10]. The formula for determining NPV is as given in (9).

$$NPV = (C - M)/(1 + d)^T - I + \Omega \tag{9}$$

In the NPV equation, C represents the value of the asset, M represents the continued cost of operating the service, d is the discount rate or risk-free interest rate, T is the initial development time, I is the initial development cost, and Ω is the value of flexibility related to the service.

I and T are determined using the cost-analysis techniques presented in the Cost Determination section. Additional contributors to cost, such as infrastructure or hardware costs, which are beyond the scope of this paper, can be directly added to I. M is determined by adding the normal maintenance costs to the non-software-related costs, such as those incurred through a license agreement with another company to use or incorporate a service of theirs. The discount rate d accounts for the opportunity cost involved in the investment, and is set to the rate of growth the money invested in the project would have obtained had it not been used to develop this service. If no other risk-free projects are available to compare against, this variable can be set to the growth rate of a 'tracking portfolio', which is a portfolio composed of projects related to the project to be undertaken [12]. This leaves the value-related variables C and Ω to be determined.

The value of the service, C, is determined by market forces and can be ascertained through traditional high-level market research. This variable simply models the amount of income the owner can expect to receive related to the operation of the service. The flexibility premium Ω, on the other hand, is more difficult to model. To determine a value for Ω, we can relate the flexibility inherent in services to real option theory, which allows us to use the Black-Scholes formula described below [13]. To do this, we must first identify the manner in which flexible areas of a service can be related to real options.

Flexibility and Real Option Theory An option is the right to purchase or sell an asset for a fixed price for a limited time [11]. This fixed price is referred to as the 'strike price', while the current market value of the asset is referred to as the 'spot price'. A 'real option' is a refinement of this definition that only deals with real assets, not financial derivatives. This changes the definitions related to the option slightly in that the strike price is no longer the exercise price for the option, but is the investment cost instead. Similarly, the spot price is not the current value of the stock, but the gross value of expected cash flow [15].

There are two types of options: 'call' and 'put' options. With a call option, the owner has the right to acquire an asset for a fixed price, and contrarily a put option gives the owner the right to sell an asset for a fixed price. To relate service flexibility to real options, the various flexible aspects must be placed in one of these two categories.

In service-oriented development, a service is often created as the composition of other services, some of them delivered by other businesses. For example, imagine a concierge service that, among other services, books travel arrangements and accommodations. This service will be composed of connections to scheduling, travel, and booking services, all of which are likely managed by separate corporate entities. When creating the concierge service the owners will have many choices regarding which company to use for their services; should they use travel agent A or travel agent B to manage airline ticket bookings? While their initial decision can be based on a simple cost/benefit analysis of each of the choices, they also have the option to change, or replace, service providers in the future. This replacement is akin to a call option; they have the option to purchase another service for a fixed price (*i.e.,* the price the provider regularly charges), yet they will get a variable return from their investment based on the quality improvement of their product due to the provider switch. Also similar to a call option is the ability to add new services in the future. If the market demands that the concierge service add a car rental service to their service portfolio, the company has the option to do so. Generally new services will only be added in this fashion if the predicted benefit of providing the service is greater than the cost of incorporating and maintaining the service. Conversely, if the market no longer desires a service, the company has the option to drop that service from their portfolio; this is analogous to a put option. If the cost of providing go-kart bookings is greater than the demand from the concierge service's customers, then it would be prudent for the company to abandon this service. In this case the spot price would be the cost of abandoning the service, such as early contract termination fees from the provider, and the strike price would be the 'termination value', or the value of the salvaged resources from the discontinued service [9]. Finally, there is the option to upgrade a service. This can also be classified as a call option, where the strike price is the cost associated with upgrading, and the spot price is the benefit gained through the upgrade, which is the scenario demonstrated in the above example.

Valuating Options To determine the actual economic value related to the ability to exercise an option, the Black-Scholes formula [3] can be used. This formula is different for call and put options, as is shown in (10), (11), and (12).

$$\Omega_{call}(U,S,T,r,\sigma) = U \cdot N(d) - S \cdot e^{-rT} \cdot N\left(d - \sigma\sqrt{T}\right) \tag{10}$$

$$\Omega_{put}(U,S,T,r,\sigma) = S \cdot e^{-rT} \cdot N\left(-d + \sigma\sqrt{T}\right) - U \cdot N(-d) \tag{11}$$

$$d = \frac{\ln\left(U/S \cdot e^{-rT}\right)}{\sigma\sqrt{T}} + \frac{1}{2}\sigma\sqrt{T} \tag{12}$$

The parameters of this model are as follows: U is the spot price, S is the strike price, T is the number of periods until the option expires, r is the constant risk-free interest rate (d in the NPV formula), σ is the constant volatility of the asset, and $N(x)$ is the cumulative standard normal distribution function. Once the Ω variable for the option has been determined, it can be added to the NPV formula to determine the overall value of the service. If the resulting number is positive, the service will create a positive cash flow for the company. Consequently, if NPV is negative, the company should use their resources on another project instead. In the case where multiple projects are possible and all have positive NPVs, the project with the greatest NPV should be chosen.

3 An Illustrative Example

To assist in the understanding of the concepts developed above, we illustrate the model we have developed with our motivating example of the Amazin.com case study. Note that all cost figures, value amounts, time frames, and situations are fictional; they simply serve to illustrate the various stages of applying our economics model to estimate ROI.

3.1 Initial State

In the beginning, when they are considering a potential expansion, Amazin.com is a fairly new online retailer; their sole market is books and all services are offered over the Internet, using a software system built using web services. They are doing fairly well financially, want to expand their business.

The current financial condition of the company is as follows: the value of the service is \$500K; the cost of operating the service is \$400K; there are no options held. This places the current NPV at $NPV_{initial} = 500K - 400K = \$100K$.

3.2 Cost

Amazin, while successful in the book-selling market, would like to apply their existing web-service architecture to a new domain in an attempt to expand. To do this, they can modify their systems to sell CDs as well as books. To modify the existing system to additionally manage CD sales, the following amounts of alterations must be made: 4% of the original design needs to be modified, 6% of the original code needs to be modified, 20% of the original system needs to be retested, 23 function points are to be altered, and 10 new function points are to be added. Additionally, none of the code is being automatically translated, as all of the modifications are being performed manually.

The following development environment factors are hypothesized: the FP to SLOC ratio is 1:27; Software Understanding (SU) is nominal; the Assessment and Assimilation Increment (AA) is high, as the programmers are already intimate with the code base, and similarly, Programmer Unfamiliarity ($UNFM$) is low.

This will yield the following equivalent SLOC calculation to determine the extent of the development feat:
$AAF = (0.4 * 0.04) + (0.3 * 0.06) + (0.3 * 0.2) = 0.094$
$AAM = [8 + 0.094 * (1 + (0.02 + 30 + 0.0))]/100 = 0.1092$
$EKSLOC = 23 * 27/1000 * (1 - 0/100) * 0.1092 = 0.6781$

With $EKSLOC$ determined, an actual effort value can be calculated, with the help of a few more assumptions about the development environment: all Effort Multipliers are set to their nominal values; all Scale Factors are set to their nominal values; all constants are left at their default COCOMOII.2000 value; the Requirements Evolution and Volatility Level ($REVL$) is set at 5, indicating that 5% of the translated code was discarded.
$Size = (1 + 5/100) * (10 * 27/1000 + 0.6781) = 0.9955$
$E = 0.91 + 0.01 * (3.72 + 3.04 + 4.24 + 3.29 + 4.68) = 1.0997$
$PM = 2.94 * 0.9955^{1.0997} * 1 = 2.925$

The time to develop the CD aspect of Amazin, including planning and preparation time, can also be calculated:
$F = 0.28 + 0.2 * (1.0997 - 0.91) = 0.31794$
$TDEV = 3.67 * 2.925^{0.31794} * 100/100 = 5.163$ Months

Assuming that a programmer is paid \$50/hour, and there are 152 work-hours in a month, the dollar amount associated with this production cost would be:
$Cost = 5.163 * 50 * 152 =$ \$39238.80

The entire set of modifications to the system will require roughly \$40 000 to perform, with a time frame of 5.163 months, assuming only one programmer is working on the expansion. More programmers can be assigned to the task to expedite the development process. We can assume there are 5 programmers adding CD sale functionality, for a total of just over 1 month development time.

For the sake of this example, assume that each CD costs Amazin \$7 to purchase and handle; this \$7 per CD is the ongoing cost of the service.

3.3 Value

With the cost of the new CD service determined, the value that can be achieved needs to be established. This is done though market analysis. For the sake of simplicity, we assume that all CDs are sold for the same price. The supply and demand of CDs at different prices is shown in Table 1. The supply column states the amount of resources Amazin is willing to redirect from selling books (*i.e.,* the number of CDs Amazin is willing to sell) at each price point, and the demand column states the number of CDs customers are willing to purchase at each price point. The optimal price point in this case is the intersection point, at \$16 per CD. The value of this is:
$Value = 16 * 10000 =$ \$160000

Table 1. CD Supply and Demand

Price	**Supply**	**Demand**
\$ 13	7000	13000
\$ 14	8000	12000
\$ 15	9000	11000
\$ 16	10000	10000
\$ 17	11000	9000
\$ 18	12000	8000

Taking into account the ongoing cost of purchasing and handling CDs, the value of the CD service is effectively:
$EffectiveValue = 160000 - 7 * 10000 =$ \$90000

Since the entry into the CD market is a new endeavor, it is expected to grow in value by 50% annually.

3.4 Option Theory Analysis

Both pieces required for analysis of the expansion strategy are now in place; the cost of developing the option and the value of executing the option are established. The value of the option can then be calculated using the following variables, gathered above: the spot price (U) is 90000, the strike price (S) is 40000, the risk-free interest rate (r) is 0.05, periods until expiration (T) equals 0.43 years (5.163 months), and the expected growth (σ) is 0.5. This yields:
$d = \frac{ln(90000/40000*e^{-(0.05)(0.43)})}{0.5\sqrt{0.43}} + \frac{1}{2}0.5\sqrt{0.43} = 2.5717$
$\Omega = 90000 * N(2.5717) - 40000 * e^{-(0.05)*(0.43)} * N(2.5717 - 0.5 * \sqrt{0.43})$
$=$ \$49761.92

This option value can then be added to the initial NPV calculation as the Ω variable, yielding:
$NPV_{new} = 100K + 49.76K =$ \$$149.76K$

By entering in to the CD market with their new web service, Amazin will be able to increase their NPV by nearly 50%. If Amazin is considering other expansion strategies as well, similar calculations can be performed on them and the highest-valued option can be chosen as the one to implement.

4 Related Research

During the 1970s and early 1980s numerous software cost models were proposed [7], including COCOMO [4], SLIM [27], PRICE S [16], SEER [20], Estimacs [29], and SPQR/Checkpoint [21]. Since their conception, most of these models have become antiquated, are no longer widely used, or have become proprietary.

COCOMO is an exception: it has been extensively published, and it evolved into COCOMOII [6] and later into COCOMOII.2000 [5]. Presently, nearly all commercial software cost estimation products are based on COCOMOII [31, 1, 18, 19], with the exception of some models tailored specifically to web hypermedia applications [24]. COCOMOII is easily accessible, and can be fine-tuned to suit almost any development environment and project size, which is the reason we chose it as the basis of our work.

Value estimation has not been equally extensively researched. As Sari Kalin's article in CIO magazine [22] mentions, integrating applications and services can either significantly improve the efficiency of a company, or it can lead to the waste of significant quantities of resources if performed improperly. Kalin proceeds to state that an accurate ROI calculation is crucial to the integration decision. One of the example corporations represented in the article deals with integrating web services with their company in order to streamline communication with suppliers. Though the article establishes the need for a viable ROI calculation, no suggestions for such a calculation are provided. IBM, on the other hand, discusses methods in which ROI can be calculated in their discussion of web services and enterprise-wide SOAs [25]. The two methods that they describe are Book-value ROI and Net Present Value (NPV). As NPV takes into account the time-value of money, where Book-value ROI does not, all value calculations in this paper will use NPV for the ROI calculations. NPV has been used successfully in the past for modeling the value of commercial-off-the-shelf (COTS) based development [10, 13], and has been proven to work well in environments where component reuse is prevalent [14], as is the case with web services. Gunjan Samtani of UBS PaineWebber and Dimple Sadhwani of Island ECN extend the ROI calculation for web services by listing the multiple aspects of a project that should be examined in order to account for all possible sources of costs and benefits [30]. It is with these sources in mind that we propose our economic model.

5 Conclusion

In this paper, we presented our work modeling the software economics of service-oriented development. Service-oriented development is fundamentally incremental in nature: services are reused and composed in different ways to deliver new, more complex, value-adding services. When business that deliver B2C services through service-oriented applications make decisions on how to evolve their systems in order to deliver more to their customers, they have to consider the alternatives and evaluate the cost of the software development and the potential value of the envisioned service. In our work we have adopted COCOMOII.2000 as the basis for estimating effort and Net-Present Value integrated with Real-Options Theory for value estimation. In our example case study, we have shown how our model can be used to estimate the ROI of a considered expansion decision, by estimating the cost of the necessary software development and the expected value of the expanded service.

This work is by no means mature. The COCOMOII.2000 model needs to be calibrated for this particular software-development lifecycle model. Market research ideas and work on customer expectations of service quality should be integrated in order to better estimate the potential value of a new service. We consider the work we reported here as a first step in this challenging and exciting research field, with the calculation of the numerous other contributors to cost and value, and their incorporation into this model, to be our future research projects.

Acknowledgements

This research was supported by a grant from the Alberta IBM CAS. Thanks to Hakan Erdogmus for his excellent feedback on sections of this paper.

References

[1] U.S. Air Force Analysis Agency. Revic. http://sepo.spawar.navy.mil/Estimation.html.

[2] Amazon.com. 1998 annual report. http://media.corporate-ir.net/media_files/irol/97/97664/reports/123198_10k.pdf, March 1999.

[3] Fischer Black and Myron S. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–54, May-June 1973. Available at http://ideas.repec.org/a/ucp/jpolec/v81y1973i3p637-54.html.

[4] Barry Boehm. *Software Engineering Economics*. Prentice Hall PTR, October 1981.

[5] Barry Boehm. *COCOMO II Model Definition Manual*. University of Southern California, 2000.

[6] Barry Boehm, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, and Richard Selby. Cost models for future software life cycle processes: Cocomo 2.0. *Annals of Software Engineering*, 1(1):57–94, December 1995.

[7] Barry W. Boehm and Kevin J. Sullivan. Software economics: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 319–343, New York, NY, USA, 2000. ACM Press.

[8] Joseph A. Dane. Modular program size counting. Master's thesis, University of Hawaii, December 1999.

[9] Hakan Erdogmus. Building a business case for cots-centric development: an investment analysis perspective. In *Proceedings of the ICSE '99 Workshop on Ensuring Successful COTS Development*, May 1999.

[10] Hakan Erdogmus. Comparative evaluation of software development strategies based on net present value. In *ICSE'99 International Workshop on Economics-Driven Software Engineering Research (EDSER1)*, May 1999.

[11] Hakan Erdogmus. Valuation of complex options in software development. In *ICSE'99 International Workshop on Economics-Driven Software Engineering Research (EDSER1)*, May 1999.

[12] Hakan Erdogmus. Value of commercial software development under technology risk. *The Financier*, 7, 2000.

[13] Hakan Erdogmus and Jennifer Vandergraaf. Quantitative approaches for assessing the value of cots-centric development. In *METRICS '99: Proceedings of the 6th International Symposium on Software Metrics*, page 279, Washington, DC, USA, 1999. IEEE Computer Society.

[14] John Favaro. A comparison of approaches to reuse investment analysis. In *ICSR '96: Proceedings of the 4th International Conference on Software Reuse*, page 136, Washington, DC, USA, 1996. IEEE Computer Society.

[15] John M. Favaro, Kenneth R. Favaro, and Paul F. Favaro. Value based software reuse investment. *Annals of Software Engineering*, 5:5–52, 1998.

[16] F.R. Freiman and R.E. Park. Price software model-version 3: An overview. In *Proceedings, IEEE/PINY Workshop on Quantitative Software Models, IEEE Catalog No. TH0067-9*, pages 32–44, October 1979.

[17] John Hagel. Winning at soa (editorial). Sand Hill Group. http://www.sandhill.com/opinion/editorial.php?id=13, 2005.

[18] Cost Xpert Group Inc. Cost xpert. http://www.costxpert.com/.

[19] Software Productivity Center Inc. Estimate professional. http://www.spc.ca/resources/estimate/index.htm.

[20] R.W. Jensen. An improved macrolevel software development resource estimation model. In *Proceedings, ISPA 1983*, pages 88–92, April 1983.

[21] C. Jones. *Programming productivity.* McGraw Hill, 1986.

[22] Sari Kalin. Return on investment - calculating roi. *CIO Magazine*, August 2002. Available at http://www.cio.com/archive/081502/roi.html.

[23] Mitchell McInnes, Ian Kerr, J. Anthony VanDuzer, and Chi Carmody. *Managing the Law: The Legal Aspects of Doing Business.* Prentice Hall, 2003.

[24] E. Mendes et al. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[25] Judith M. Myerson. Work with web services in enterprise-wide soas: Part 13. http://www-128.ibm.com/developerworks/webservices/library/ws-soa-enter13/. Last accessed on June 6^{th}, 2006.

[26] University of Southern California Center for Software Engineering. Codecount. http://sunset.usc.edu/research/CODECOUNT/.

[27] L.H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, pages 345–361, July 1978.

[28] S.A. Ross et al. *Fundamentals of Corporate Finance*. Irwin, 1996.

[29] H.A. Rubin. A comparison of cost estimation tools. In *Proceedings, ICSE 8*, pages 174–180, August 1985.

[30] Gunjan Samtani and Dimple Sadhwani. Web services return on investment - working out what you're getting out of web services. In *Web Services Business Strategies and Architectures*. Wrox Press, August 2002. Article available at http://www.webservicesarchitect.com/content/articles/samtani07.asp.

[31] Softstar Systems. Costar. http://www.softstarsystems.com/.