Quantifying Value in Software Product Line Design

John D. McGregor
School of Computing
Clemson University
Clemson, SC USA
johnmc@cs.clemson.edu

J. Yates Monteith
School of Computing
Clemson University
Clemson, SC USA
lrddpy@gmail.com

Jie Zhang
School of Computing
Clemson University
Clemson, SC USA
zhangjie190@gmail.com

ABSTRACT

A software product line is a strategic investment for an organization. Besides the initial decision to use a product line approach other strategic decisions are made, including which variations to accommodate. In this paper we present an adaptation of an equation for computing option values. The equation can be used to understand the economic impact of adding a variation point to the product line architecture. The equation was exercised on multiple sets of hypothetical data and and produced the expected changes from one data set to another. In the future the equation will be validated with data from real projects. We describe some practical sources of values for the parameters of the equation.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*variation management, real options*

General Terms

Software Engineering,Economics

Keywords

strategic software design

1. INTRODUCTION

Software product line strategies realize tremendous savings due to reuse of assets across multiple products; however, this savings can be diluted by decisions that seem valid but which are based on uncertain information. Not only is some of the data uncertain, but also the degree of uncertainty changes over time. In some cases time allows the uncertainty to be eliminated, in others the uncertainty can only be reduced through improved estimation techniques. Additionally, a software product line encounters uncertain forces because of changes over its life time. Many software reuse efforts have failed to realize significant reductions in costs, and in some cases have raised costs, because, after extra resources were expended to make assets reusable, anticipated future profitable uses of those assets failed to materialize.

A software product line is a set of products we intend to develop from a common set of assets over some time [4]. The use of "intend" is deliberate because whether each of those products is actually produced is uncertain and is more uncertain the further into the future the product is scheduled to be produced. Each product will include some of the features provided by the product line and it is this ability to pick and choose features that provides the product developer with options by providing the opportunity to select from among design variants.

A software product line includes products that differ from one another in specific ways. The template specification of a product in the product line contains variation points, those design points at which some of the available design variants may or may not be used. Each variation point represents a decision about what to include and what not to include in a product. These variation points and the accompanying variants provide the product line organization with options for future products, but there is always some uncertainty associated with the "future."

An option is the right, but not the obligation, to perform some action in the future. In our case each variation point represents an option to vary the behavior of a piece of software from one product to another. This is a "real" option since it deals with a product rather than an abstract financial instrument. A variation point is implemented by a variation mechanism that requires resources to design and implement and that may well impact the quality attributes of the products in the product line.

The option approach has two costs. "purchasing an option" is spending money now with the possibility of a payoff in the future. "exercising the option" is spending money at that future time to take advantage of the opportunity presented by the option. The cost of purchasing the option is the cost of implementing the variation point. The cost of exercising the option, which we are not obligated to do, is the cost of creating the necessary variant and the cost of inerting it at the variation point.

The revenue received from the products made possible by those variation points is the return from exercising the option. Therefore we are actually interested in two decisions.

First is the decision of whether or not to create a variation point, which is of strategic importance and will be treated as a real option, and second are the decisions of which variants to have available for use in a product. Treating these two ideas separately is a strategic advantage for the product line organization [5].

Adding another variant to the set of variants that can be chosen at a variation point is usually straightforward. Adding a new variation point to an existing asset is often the equivalent of making a fundamental architectural change to an existing product - expensive and time consuming. Therefore the initial design that inserts a variation point is a strategic decision that commits resources against the anticipation of future income.

The contribution of this paper is an investigation into the applicability of real options to decision making regarding uncertainty, risk, and value in a software product line organization as they relate to variation points. We provide a preliminary view of a framework based on real options within which approaches to valuing actions such as creating a variation point can be exercised. We apply previous work on real options to the problem of computing the value of a variation point.

In the following sections we first identify the sources of uncertainty in a product line context. Then we discuss how uncertainty is handled and we relate the handling of uncertainty to the concept of value and real options. We introduce previous application of real options and then use their approach to define a framework for computing the value of options. We apply the framework to a hypothetical scenario and discuss the results.

2. SOURCES OF UNCERTAINTY

Uncertainty in a product line arises from many sources. Olagbemiro, Mun, and Shing list Heisenberg, Gödel, and pragmatic types of uncertainty [6]. The Heisenberg type of uncertainty occurs because of changes made during implementation resulting in failure to satisfy requirements after implementation. The Gödel type of uncertainty arises because of differences between a model representation and the real-world artifact. Pragmatic uncertainty is related to the accuracy of human actions such as various types of estimates.

For our purposes we use the categories from the SEI's framework for Product Line Development: software engineering, technical management, and organizational management to further classify types of uncertainty. In Table 1 we give an example of each type of uncertainty.

Uncertainty relates to (1) how accurate the information is at the moment it is used, and (2) over time how likely is that information to change so that the information supporting a decision is less accurate. Our model must address all these types of uncertainty in order to be a faithful representation of the decision making environment.

3. HANDLING UNCERTAINTY

Uncertainty is handled by first trying to reduce the degree of uncertainty and second by reducing the impact of that uncertainty. A software product line incurs some amount of uncertainty by planning for multiple products into the future. The uncertainty about whether a product will actually be developed as planned is reduced by initially conducting a thorough analysis of market trends to more accurately predict demand and developing clear specifications of customer needs.

Table 1: Examples of uncertainty

	Software engineering	Technical management	Organizational management
Heisenberg	Requirements change and new architectures come into vogue	The scope of a product line often changes so decisions made early carry uncertainty	The business case is sensitive to changes in the business climate over time.
Gödel	When an architecture barely meets a quality attribute, the uncertainty in those estimates make it impossible	Technical risks will be directly affected by uncertainty about the information used to establish the risks.	Technologies may not be as compatible in actuality as they appeared to be in the forecast.
Pragmatic	Individual estimates of performance for modules can be off if a new implementation is used	The probabilities used to evaluate risk may not be known exactly.	Market projections may be optimistic or a disruptive event may change the trajectory.

A software product line uses variation points to manage the differences among products and reduce the impact of uncertainty. Inserting variation points into early designs mitigates the late discovery of a need. We use a variation mechanism that allows as late a binding as is practical depending on what the variation is; however, often later binding also means increased cost either because more complex technology is needed for a late binding or because test costs have increased since the binding will happen outside the manufacturing facility.

4. VALUE

Value is money and we work to create or keep that value using strategy [1]. Designs create value [1] by providing options. The potential impact of a design decision is magnified in a software product line since it impacts more products than most design decisions and affects the organization over a longer time period than most design decisions. In this paper we will present a method for quantifying the value of these options. This approach allows designers to compare designs, which differ by the variation points included in each design.

Value is often taken to be synonymous with cost in software engineering analyses; however, there are several facets to value in a software product line. First, the core assets represent a special kind of value similar to a professional's expertise because software is a non-rival good. That is, the use of the asset in one product does not prevent it from being used in additional products, unlike a physical piece of hardware that can only be placed in one product. Second, an asset may represent a competitive advantage due to an unusually efficient or comprehensive algorithm to which no competitor has access. The value is the ability to attract buyers who might have purchased other software. Third, there is value in having mitigated uncertainty by having a variation point available even if the needed variant does not yet exist. The new feature will be provided much faster and at much less cost than if the variation point had not been implemented. It is this type of value that we will address in

this paper.

Value changes over time. It changes in response to changes in the market and changes in technology. If one of several products in which an asset is to be used is cancelled before being manufactured, the value, to the organization, of those assets that were to be used in the product is reduced. If a supplier raises prices on a piece that is required to make an asset the value of that asset is reduced. The first type of change is difficult to mitigate but the second is mitigated by modularity that makes it easier to find alternative implementations.

As uncertainty increases, risk increases but so does the value of options that can mitigate some of that risk. Choosing the set of options to have available so as to produce the greatest value is a challenging task. It is challenging not only because of the uncertainty but because in many cases we have no easily available means of comparing the value produced by different options in the form of software assets. The product line designer needs tools to aid in making these decisions.

There are many types of design options possible in software development. The most commonly referred to type of option is providing multiple implementations of an interface. In fact, being able to swap one implementation for another without sending ripples throughout the rest of the system is widely viewed as a powerful tool for creating value [2]. In the software product line literature the inclusion/exclusion type of design option is also widely discussed. A product is defined by exercising options to include or exclude certain features. This capability can be implemented in several ways including switches that simply enable features that are always in the code or by a plug-in mechanism that allows a feature to be added or removed even while the program is running.

An organization's ecosystem is constantly changing and this results in changes in value. In this environment, the decision to create a variation point that is a definite "go" at a point in time is accompanied by some uncertainty about how long that decision will be valid. The focus of this paper is not on the variants that can be bound at a variation point but on the variation points themselves. The question being, "how can we provide quantitative information to guide the decision to create a particular variation point?" Sullivan et al [8] previously explored the value of modularity to a product using the Net Options Value formulation defined by Baldwin and Clark [2], but they did not consider the valuation of individual variation points.

5. MAPPING OPTION CONCEPTS TO SOFTWARE PRODUCT LINE CONCEPTS

The notion of a real option has been studied in a number of contexts. There are several theories that impose constraints on the computation of the value represented by an option. We will point out some of these as we proceed. First we relate the basic concepts of options to the basics of software product lines. The value v of a real (non-income producing) option that pays off W(T) at future time T is given by the general formula 1:

$$v(t,T) = e^{(-r(T-t))} E\left[max(0, W(T))\right] (1)$$

, where t is current time, E denotes the risk-neutral expected value, and r is the riskless discount rate [7]. Briefly, risk-neutral refers to a measure which is equally balanced between no risk and high-risk. The expected value is the average of a set of random values.

- Cost - While there are many costs in a software product line organization we focus on the cost of a single variation point. The investment cost includes the specification, design, and implementation of the variation mechanism and similar costs for the variants that can be selected at the point.
- Value - The value of a variation point is directly related to the sales that will result from having that feature. Some features will be required for a customer to even consider purchasing the product. Those are classified as mandatory features and can be ignored in a discussion on variation points. Two sources for this information on value are marketing, which is a specific product line practice, and reviews of comparable competitive products.
- Risk - One risk of including a variation point in a product line architecture is the possibility that the variation point is never used. That is, an option is created but overall the products in the product line that option is never exercised. Resources have been expended without any return.
- Flexibility - One issue is the amount of flexibility provided by the variation point. This relates to the range of variants that can be offered through the variation point. A Bluetooth connection allows a number of devices to be attached but there are limits on bandwidth that preclude certain applications.

A software product line can be viewed as a portfolio of options; however, each of the options can be exercised at multiple times. Every real option must be accompanied by an underlying asset that provides a basis for the value of the option. In our case, the asset underlying each option is the product revenue that can be attributed to the presence of a particular variation point. The revenue can sometimes be approximated by comparing the prices of products that do not have the feature enabled by the variation point to those that do have it. It can also be approximated by dividing the revenue by the number of optional features.

6. OUR COMPUTATIONS

We follow the lead of NASA's use of real options to define our approach[7]. NASA defined a model in which a set of technologies are matured to be ready for use on missions and decisions are made as to which technologies to use. Whether a technology would have reached maturity by the time of a specific mission was uncertain as was which missions were actually to be flown.

Our portfolio of options is a portfolio of variation points. We have two types of uncertainty that must be addressed. First there is uncertainty about which variants will be needed at each variation point and when they will be needed. Second,

there is uncertainty about which products will actually be built. The initial scope gives a best estimate at a point in time but later in the product line life cycle the business climate may prevent buyers from supporting a product or new technologies may have obviated the need for the product.

Equation 2 aids the product line manager in integrating information from the software architect, the core asset development lead, product managers to make decisions about variation points using information about several aspects of the product line. The product release schedule is used to determine when revenue begins and when assets must be ready. The spread of personnel over the development schedule will help estimate the cost per period per asset.

Equation 2 gives the expected present value of a resource (in our case a variation point) at time T given a current time t. We use the present value form of the equation because we are interested in making initial decisions about including, or not, each variation point in the product line architecture.

$$v_i(t,T) = \max\left(0, -E\left[\sum_{\tau=t}^{T} c_i(\tau) e^{-r(\tau-t)}\right] + p_{i,T} E\left[\sum_{k} \max\left(0, \sum_{\tau=T}^{T^*} X_{i,k}(\tau)\, e^{-r(\tau-t)}\right)\right]\right) \quad (2)$$

where

$$X_{i,k}(\tau) = VMP_{i,k}(\tau) - MC_{i,k}(\tau)$$

The components of the equation are as follows: E is the expected value function and is used to reflect the two kinds of uncertainty. The first addresses the cost of implementing the variation points and the second addresses the net value of the products being produced.

r is the risk-free discount rate. This is the rate of return someone would expect from a very low risk investment such as a US government bond.

$c_i(\tau)$ is the cost of developing the variation point "i" in period τ for an exercise date of T; this allows for the incremental development of i.

$p_{i,T}$ is the probability that the implementation of variation point "i" will be successful by the exercise date T. $X_{i,k}(\tau)$ is the net marginal value of variation point i in product k in period τ given a successful implementation of i; essentially $X_{i,k}(\tau) = VMP_{i,k}(\tau) - MC_{i,k}(\tau)$ where

$VMP_{i,k}(\tau)$ is the marginal value of the contribution of variation point i in product k in period τ given a successful implementation of i, and

$MC_{i,k}(\tau)$ is the marginal cost of "productizing" variation point i in product k in period τ given a successful implementation of i. At the initial product there may be a cost to tailor the asset to fit the exact need in that product. This marginal cost will be essentially the same as C_{reuse} [3] in succeeding products after the variation point is used in one product. C_{reuse} is one cost function in the Structured Intuitive Model for Product Line Economics (SIMPLE). It is the cost of reusing an asset, which was made to be reusable, in the implementation of a product.

T^* is the time horizon of the product line ($T^* >= T >= t$) which we will take to be the time at which all of the hypothesized products are scheduled to have been released (We view the value as being accrued when the product is built, but the analysis might be extended to include further versions of the product.).

Equation 2 is a standard formulation for computing an option value and does not yet reflect the special circumstances of a software product line. It is how the terms in the equation are defined that embodies the product line flavor. The reuse of an asset across multiple products enters into the computation in two ways. First, the total cost of an asset is amortized over the set of products which use that asset. Our planning includes developing a table which associates each product with the assets that will be integrated to form that product. This allows us to compute scenarios in which products are dropped from the product line and compare that result to the original value with all products created. This is useful to model scenarios in which products further into the future are less likely to ever be produced. Second, the total revenue produced by a product is allocated to its constituent assets using the product/asset mapping. The assets could be weighted to reflect their relative improtance. The time by which a variation point is needed is determined by the release date of the first product that uses the asset that includes the variation point.

Given the preceding formulation of the options view, the configuration of a product can be seen as the exercise of multiple options while at the same time deciding not to exercise certain other options. Selecting a variant to use at a variation point is an exercise of the option represented by that variation point. Early in the life of a product line configuring a product may result in the selection of a variant that has not yet been implemented. The cost of that implementation would be incurred at that point. In our initial exercise of the equation we will assume that a variation point and all variants are ready at the same time.

7. AN ILLUSTRATIVE SCENARIO

The general option equation 2 can be used in many ways. By varying the contents of the product/asset mapping, product release schedule, and the parameters such as $p_{i,T}$ we can consider different impacts of the option value. In this paper we report on only one instantiation of the basic equation.

We begin experimenting with the following use scenario. A product line manager is planning a new software product line. We have a product line of 9 products that are scheduled to be released, one every 6 months (every period), beginning with the second 6 months, covering 5 years. We will consider that the products are identified as products 0 - 8 and will be released in numerical order. There will be 11 core assets. The cost of each asset, which includes the cost

Table 2: Asset/Product Mapping

Asset/ Product	Product 0	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8
Asset 0	1	1	1	1	1	1	1	1	1
Asset 1	1	1	1			1			
Asset 2	1			1		1	1	1	1
Asset 3				1	1			1	1
Asset 4		1	1		1		1		1
Asset 5	1	1			1	1		1	
Asset 6						1	1	1	1
Asset 7	1	1	1	1		1	1		
Asset 8		1	1		1		1	1	
Asset 9				1		1	1	1	
Asset 10			1		1		1		1

Table 3: Asset cost per period

	Period 0	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6	Period 7	Period 8	Period 9
Asset 0	100	150	0	0	0	0	0	0	0	0
Asset 1	150	200	0	0	0	0	0	0	0	0
Asset 2	100	150		0		0	0	0	0	0
Asset 3	100	100	100	50	50	0	0	0	0	0
Asset 4	300	200	150	0	0		0	0	0	0
Asset 5	100	150	0	0	0	0	0	0	0	0
Asset 6	0	50	50	0	100	300	150	0	0	0
Asset 7	150	200	0	0	0	0	0	0	0	0
Asset 8	100	200	250	0	0	0	0	0	0	0
Asset 9	200	0		150	100	500	0	0	0	0
Asset 10	50	100	150		250	0	0	0	0	0

of the variation point, is estimated for each time period up to the time the asset is released. In this scenario we have included a variety of patterns of expenditures including one in which construction on an asset is halted for a time. The architect has allocated assets to products so that we know the frequency of use of each asset and we know when the asset must be ready by the release date for the earliest product release in which it participates. This is a best-case scenario because every asset is complete on time and every planned product is produced.

The individual product managers have already made revenue estimates as part of the business case for each product. The value of an asset is computed by allocating the total estimated revenue of each product to the assets used in that product. The value of the asset is the total of all of its revenue allocations.

The detailed data for this scenario are given in Tables 2, 3, 4.

We further assume that only asset exclusion/inclusion operations are available as variation mechanisms. We assume

Table 4: Product revenue

Product 0	1200
Product 1	1500
Product 2	900
Product 3	1000
Product 4	1600
Product 5	800
Product 6	2100
Product 7	1400
Product 8	2800

that value is not accrued until the asset and/or the product are released for their purpose.

Other parameter values are:

- r = .05/year
- $p_{i,T}$ = .85 Our initial iteration assumed this fixed value but during a second iteration we steadily decreased the likelihood of being ready after an intial period.
- i = designation for a variation point; initially we assume 1 variation point per asset
- k = product number - $1 < k < 10$; in our initial experiments we produced one product every six months but our formulation allows for products to be released in any period. The total value of the asset or product accrues immediately upon release.
- t = current time - 0 years initially
- T = exercise date = varies from 0 to T*; initially in steps of 6 months; this is the date by which the variation point is inserted into the asset or not (not when the variation point is actually used in a product)
- T* = time horizon - 5 years = 10 time periods

As stated in the previous section, Equation 2 is a basic model that can be applied to many situations. It is the model inputs that specialize the scenario. For this initial scenario the data for Tables 2, 3, 4 was created using relative sizes and patterns that are representative of expected product line situations rather than actual data from a specific project. The assumptions and constraints used to determine the costs for core asset construction and the value of each asset in each period characterize the scenario. For the initial scenario the value of each asset was determined by dividing the revenue from the product across the set of assets used to implement that product. No adjustment was made for hypothesized differences in complexity or other asset characteristics. The cost for an asset was allocated to the time periods leading up to the time period in which an asset is to first be used in a product. Different patterns were used for the assets to illustrate how typical patterns of development might affect the valuation of the variation point. The resulting $v_i(t, T)$ is a vector of the values of the individual variation points stated in current dollars.

Table 5: Initial results

Asset	Number of Products	Cost	Value with constant p	Value with decreasing p
0	9	298	6960	6955
1	4	415	3252	3253
2	5	298	4076	4074
3	4	525	2673	2313
4	5	785	3761	3772
5	5	298	4061	4077
6	4	999	1596	1089
7	6	415	4510	4515
8	5	716	3314	3322
9	4	1340	1031	806
10	4	765	2602	2420

The deterministic equation 2 was turned into a stochastic model. Each of the parameters to the model was represented by random variable with a normal distribution. We applied a Monte Carlo simulation and ran 200,000 trials with the value of p held constant for the entire time T^* and an additional 200,000 trials with the value of p decreasing by .05 in each time period after an initial 2 years of contant value for p. This simulates that estimates are less accurate the further in the future the event is. The values in Table 5 are the asset values for the completed product line at time T^*.

8. DISCUSSION

Several observations are possible from these initial computations. The model produces realistic results. Asset 9 has a much smaller value than others but analysis shows that the asset is not completed until far into the life cycle and is only used in a few products. Tweaking parameters for other assets produces the expected changes in the computed values. Most of the parameters to this model are usual for planning a software product line and are realistically available.

We have made assumptions that simplified some of our computations. One is that there is only one variation point per asset. In our future work we will investigate the effect of coupling of multiple variation points on the computation. Another restriction is that all activity happens instantaneously at the beginning of a period. The impact of this restriction can be mitigated to a degree by creating smaller intervals to more closely model the continuous nature of time.

9. FUTURE WORK

Since this is preliminary work there is much left to be done. Foremost is to use actual data from an operating product line to determine the accuracy of the decisions. We are looking for collaborators to provide data.

The equation contains several parameters that must be known for an actual scenario to be complete. We have some thoughts on how to estimate some of these.

This framework can be used with many different scenarios. For example, in the initial scenario we computed only one way to implement each variation point. We could compute the values for two alternative means of implementing the same variation point. Rather than sum up the v_i to get the total value for the product line we would compare the values and select the implementation that provides the greatest value.

The value of each asset can be modeled in several ways. First, how is value accrued? As the asset is constructed is there a gradual increase in value or does it achieve all of its value at once when the asset is available for use? Second, once it has attained full value does it retain that value in succeeding time periods or does it only have that full value in a time period when it is used in a product?

Currently our view of value is primarily $revenue - cost$. A particularly difficult aspect to represent in the model is the value of having an asset that provides certain behavior so that a quick response is possible. We will eventually consider how marketing plans and technology forecasts could be used with our computation to predict the assets to "bet on" in terms of the likelihood of future usefulness. Also we do not currently have a term that allows the model to capture the value of a cutting-edge proprietary algorithm except through increased sales revenue.

This framework will need to be incorporated into a larger context to compute the full cost/benefit of a particular software product line configuration. The more complete context is given in [3].

10. CONCLUSIONS

We conclude by first providing a short summary of the process around this technique and then we have a few words of conclusion.

The equation is intended to be used during planning for a software product line and provides input to the product line business case. The product line manager cooperates with the product line architect, core asset lead, and product managers or marketing people. The product line manager estimates when each product will be released, the architect determines which assets will be used in which products, and product managers estimate revenue per product. The core asset lead estimates the hours needed to create each asset and gives a confidence interval on those estimates. The tables shown in section 7 are completed by these numbers. The stochatic model is evaluated using Monte Carlo simulation and the product line manager has a profile of asset values. We envision this being iterative with the scope of the product line as well as the estimates being manipulated to produce a profitable product line definition.

In this initial work we have dealt with a number of issues. The way we stated the option - whether to insert the variation point or not - has eliminated the issue of exercising the option multiple times. The main issues that must be addressed are values for parameters to the model. The framework has been constructed in such a way that most parameters can be estimated from earlier previous development experience or from the data used in developing the business case or the product line scope. There are still several models of development to choose among and we will investigate these as the work proceeds but we expect that most will remain viable and will be selected by the individual product line engineer. The framework we have developed supports scoping a software product line. The results can be used to

Table 6: Parameter value sources

r	riskless discount rate	use government bond interest rate
$p_{i,T}$	the probability that development will be completed by time T	expert project manager judgement
$VMP_{i,k}(\tau)$	Value added by the asset to the product during a time interval	could use an earned value calculation
$MC_{i,k}(\tau)$	Cost of work on an asset in a time period	estimated work schedule

identify products or assets that will be less profitable and the analysis rerun without the less profitable elements. The development model which the farmework describes makes it easy to see which remaining assets and products would be affected. This iterative approach supports finding the optimal scope. The real options model of valuation requires a large number of sometimes difficult to estimate parameters, but it is a versatile model that can support a range of analyses.

The model we have developed supports scoping a software product line. The results can be used to identify products or assets that will be less profitable and the analysis rerun without the less profitable elements. The model makes it easy to see which remaining assets and products would be affected. This iterative approach supports finding the optimal scope. The real options model requires a large number of sometimes difficult to estimate parameters, but it is a versatile model that can support a range of analyses.

11. REFERENCES

[1] C. Baldwin. Keynote presentation: The power of modularity: The financial consequences of computer and code architecture, 2006.

[2] C. Y. Baldwin and K. B. Clark. *Design Rules: The power of modularity*. The MIT Press, 2000.

[3] G. Boeckle, P. Clements, J. D. McGregor, D. Muthig, and K. Schmid. A cost model for software product lines. In *Proceedings of the Product Family Engineering Conference 5*, volume 3014/2004, pages 310–316. Springer Lecture Notes in Computer Science, 2004.

[4] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[5] J. Hunt and J. D. McGregor. When less is more: Implementing optional features. In *Proceedings of the ACM Southeast Conference 2007*, pages 30–35, March 2007.

[6] A. Olagbemiro, J. Mun, and M.-T. Shing. Application of real options theory to dod software acquisitions. Technical report. last visited May 2011.

[7] R. Shishko, D. H. Ebbeler, and G. Fox. Nasa technology assessment using real options valuation. *Systems Engineering*, 7(1), 2004.

[8] K. J. Sullivan, W. G. Griswold, Y. Cai, and B. Hallen. The structure and value of modularity in software design. In *Proceedings of ESEC / SIGSOFT FSE*, pages 99–108, 2001.