Simulation for Business Value and Software Process/Product Tradeoff Decisions

Raymond Madachy
USC Center for Software Engineering
Dept. of Computer Science, SAL 318
Los Angeles, CA, 90089-0781
1 213 740 5703
madachy@usc.edu

ABSTRACT
Business value goals should be considered when making software process and product decisions, but it is usually difficult to integrate the perspectives quantitatively. This research uses simulation to assess product/process tradeoffs for economic business case analysis. A system dynamics model for commercial software enterprises relates the dynamics between product specifications, investment costs, schedule, software quality practices, market size, license retention, pricing and revenue generation. It allows one to experiment with different product strategies, software processes, marketing practices and pricing schemes while tracking financial measures over time. It can be used to determine the appropriate balance of process activities to meet business goals and product criteria. It can be used to determine the appropriate balance of process activities to meet business goals and product criteria. Applications are demonstrated for varying scope, reliability, delivery of multiple releases, and determining the quality sweet spot for different time horizons. Results show that optimal policies depend on various stakeholder value functions, opposing market factors and business constraints. Future model improvements are also identified.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Management – cost estimation, software process, time estimation.

General Terms
Management, Measurement, Economics, Reliability

Keywords
Software process simulation, value-based software engineering, system dynamics.

1. INTRODUCTION
Value-Based Software Engineering (VBSE) integrates value considerations into software engineering principles and practices [1]. This research from [2] addresses the planning and control aspect of VBSE to manage the value delivered to stakeholders. A system dynamics simulation model of cost, schedule, quality and business case analysis allows decision tradeoff studies for a commercial software development context. Business value is accounted for in terms of return-on-investment (ROI) of different product and process strategies.

It is a challenge to tradeoff different software attributes and particularly between different perspectives such as business and software development. System dynamics is a rich and integrative methodology to model interactions and feedback loops over time, and thus is handy to reason about complex software value decisions. It can help find the right balance of activities that contribute to stakeholder value with other constraints such as cost, schedule or quality goals. An example of a static modeling approach to VBSE is the iDAVE spreadsheet model used to estimate the ROI of investments in software dependability [3].

Two major aspects of stakeholder value are addressed here. One is the business value to the development organization stemming from software sales. Another is the value to the end-user stakeholder from varying feature sets and quality. Production functions relating different aspects of value to their costs are included in the integrated model.

This position paper overviews the system dynamics model, provides a worked out example of a tradeoff analysis, summarizes the results of additional analyses, and discusses improvements and future work.

2. MODEL OVERVIEW
The system dynamics model represents a business case for commercial software development. The user inputs and model factors can vary over the project duration as opposed to a static model. Inputs can be modified interactively by the user during the course of a run and the model responds to the midstream changes. It can be used dynamically before or during a project. Hence it is suitable for “flight simulation” training or actual project usage to reflect actuals to-date.

The sectors of the model and their major interfaces are shown in Figure 1. The software process and product sector computes the staffing profile and quality over time based on the software size, reliability setting, and other inputs. The staffing rate becomes one of the investment flows in the finances sector, while the actual quality is a primary factor in market and sales. The resulting sales are used in the finance sector to compute various financial measures.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EDSER '06, May 27, 2006, Shanghai, China.
Copyright 2006 ACM 1-59593-085-X/06/0005…$5.00.
Figure 2 shows a diagram of the software process and product sector. It dynamically calculates effort, schedule and defects. The staffing rate over time is calculated with a version of Dynamic COCOMO [4] using a variant of a Rayleigh curve calibrated to the COCOMO II cost model at the top level. The project effort is based on the number of function points and the reliability setting. There are also some parameters that determine the shape of the staffing curve.

There is a simple defect model to calculate defect levels used in the market and sales sector to modulate sales. Defect generation is modeled as a co-flow with the software development rate, and the defect removal rate accounts for their finding and fixing. See [2] for more background on these standard flow structures for effort and defects.

Figure 3 shows the market and sales sector accounting for market share dynamics and software license sales. The perceived quality is a reputation factor that can reduce the number of sales if products have many defects (see the next section).

The market and sales model presented herein is a simplification of a more extensive being used in industry that accounts for additional marketing initiatives and software license maintenance sales.

The finance sector is shown in Figure 4. Investments include the labor costs for software development, maintenance and associated activities. Revenue is derived from the number of license sales. Sales are a function of the overall market size and market share percentage for the software product. The market share is computed using a potential market share adjusted by perceived quality. The additional market share derivable from a new product is attained at an average delay time. More details of the overall model are provided in [2].

An interactive control panel interface to the model provides dynamic input sliders for scope, reliability, and market parameters. More detailed inputs and parameterizations are provided at a model level interface.

2.1 Stakeholder Value Functions

There are two value-based production functions in the model to describe value relationships, as shown in Figures 5-6. The test cases identified on these are from the first applied example in the next section.
A market share production function in Figure 5 addresses the organizational business value of product features. The business value is quantified in terms of added potential market share attainable by the features. The relationship assumes that all features are implemented to the highest quality. Since the required reliability will impact how well the features actually work, the relationship between reliability costs and actual sales is needed to vary the sales due to quality.

![Market share production function and feature sets](image)

**Figure 5. Market share production function and feature sets**

The production function for actual sale attainment in Figure 6 relates the percent of potential sales attained in the market against reliability costs, and is relevant to two classes of stakeholders. It describes the value of different reliability levels in terms of sales attainment, and is essentially a proxy for user value as well. The four discrete points correspond to required reliability levels of low, nominal, high and very high from [4]. A multi-round Delphi poll of software marketing experts was conducted to help quantify the relative sales impact of different quality levels.

![Sales production function and reliability](image)

**Figure 6. Sales production function and reliability**

### 2.2 Quality Modeling

For simplification, software reliability as defined in the COCOMO II model [4] is used as a proxy for all quality practices. It models the tradeoff between reliability and development cost. The different settings of reliability from low to very high correspond to development options. The tradeoff is increased cost and longer development time for increased quality. This simplification can be replaced with a more comprehensive quality model (see Conclusions and Future Work).

The resulting quality will modulate the actual sales relative to the highest potential. A lower quality product will be done quicker; it will be available on the market sooner but sales will suffer from poor quality.

The market and sales sector also has a provision to modulate sales based on the perceived quality reputation. A perception of poor quality due to many defects will reduce the number of sales. A bad quality reputation takes hold almost immediately with a buggy product (bad news travels fast), and takes a long time to recover from in the market perception even after defects are fixed. This phenomenon is represented with asymmetrical information smoothing as shown in Figure 7 with a variable delay in adjusting perceptions.

![Perceived quality trends with high and low quality product deliveries](image)

**Figure 7. Perceived quality trends with high and low quality product deliveries**

Test cases to simulate multiple deliveries in maintenance and operational support illustrate sensitivity of the market to varying quality. Figure 8 illustrates if the second of two deliveries has poor quality yet is fixed quickly. It results in a change of ROI from 1.3 to 0.9 vs. the case of two high quality deliveries. The model demonstrates how one poor release in a series of releases may have serious long term consequences.

![Sales and market for high and low quality product deliveries](image)

**Figure 8. Sales and market for high and low quality product deliveries**
3. APPLIED EXAMPLES
Representative business decision scenarios are demonstrated next. The first one demonstrates the ability to dynamically assess combined strategies for scope and reliability. Next the model is used to determine a process sweet spot for reliability.

3.1 Dynamic Scope and Reliability
The model can be used to assess the effects of individual or combined strategies for overall scope and reliability. This example will show how it can be used to change product specifications midstream as a re-plan.

Static cost models typically do not lend themselves to re-plans after the project starts, as all factors remain constant through time. This dynamic capability can be used several ways by a decision-maker including 1) assessing the impact of changed product specifications during the course of a project and 2) before the project starts, determining if and how late during the project specifications can be changed based on new considerations that might come up.

Three cases are simulated: 1) an unperturbed reference case, 2) a midstream descoping of the reference case and 3) a simultaneous descoping and lowered required reliability. Such descoping is a frequent strategy to meet time constraints by shedding features.

The market share production function in Figure 5 relates the potential business value against the cost of development for the different feature sets. The actual sales production function against reliability costs in Figure 6 is applied against the potential market capture. Settings for the three cases are shown in both production functions.

The primary inputs for product specifications are the size in function points (also called scope) and required reliability. The number of function points is the size to implement given features. The size and associated cost varies as the number of features to incorporate.

Figure 9 illustrates the reference case perturbed early to descope low-ROI features (see Figure 5 for the points on the production function). The scope goes down to 550 function points and the staffing profile adjusts dynamically for it. The schedule is reduced by a few months. In this case the potential market share increase is lowered by only two percentage points to 28%. With lower development costs and earlier delivery the ROI increases substantially to 2.2 from the reference case of 1.3.

A combined strategy is modeled in Figure 10. The scope is decreased the same as before in Case 1 (Figure 9) plus the reliability setting is lowered from nominal to low. Though overall development costs decrease due to lowered reliability, the market responds poorly. This case provides the worst return of the three options and market share is lost instead of gained.

There is an early hump in sales due to the initial hype of the brand new product, but the market soon discovers the poor quality and then sales suffer dramatically. These early buyers and others assume the previous quality of the product line and are anxious to use the new, “improved” product. Some may have pre-ordered and some are early adopters that always buy when new products come out. They are the ones that find out about the lowered quality and the word starts spreading fast.

A summary of the three cases is shown in Table 1. Case 1 is the best business plan to shed undesirable features with diminishing returns. Case 2 severely hurts the enterprise because quality is too poor.

### Table 1. Case summaries

<table>
<thead>
<tr>
<th>Case</th>
<th>Delivered Size (Function Points)</th>
<th>Delivered Reliability Setting</th>
<th>Cost ($M)</th>
<th>Delivery Time (Years)</th>
<th>Final Market Share</th>
<th>ROI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Case</td>
<td>700</td>
<td>1.0</td>
<td>4.78</td>
<td>2.1</td>
<td>28%</td>
<td>1.3</td>
</tr>
<tr>
<td>Case 1: Descope</td>
<td>550</td>
<td>1.0</td>
<td>3.70</td>
<td>1.7</td>
<td>28%</td>
<td>2.2</td>
</tr>
<tr>
<td>Case 2: Descope and Lower Reliability</td>
<td>550</td>
<td>.92</td>
<td>3.30</td>
<td>1.5</td>
<td>12%</td>
<td>1.0</td>
</tr>
</tbody>
</table>

3.2 Finding the Sweet Spot
This example analysis derived from [2] shows how the model can support software business decision-making by using
risk consequence to find the quality sweet spot with respect to ROI. The following steps are performed to find the process sweet spot:

- vary reliability across runs
- assess risk consequences of opposing trends: market delays and bad quality losses
- sum market losses and development costs
- calculate resulting net revenue to find process optimum.

The risk consequences are calculated for the different options. Only point estimates are used for the sake of this example. A more comprehensive risk analysis would consider probability distributions to obtain a range of results. Probability is considered constant for each case and is not explicitly used in the calculations. Only the costs (or losses) are determined.

A set of runs is performed that simulate the development and market release of a new 80 KSLOC product. The product can potentially increase market share by 30%, but the actual gains depend on the level of quality. Only the highest quality will attain the full 30%. Other parameterizations are an initial total market size = $64M annual revenue, the vendor has 15% initial market share, and the overall market doubles in 5 years.

A reference case is needed to determine the losses due to inferior quality. The expected revenues for a sub-quality delivery must be subtracted from the maximum potential revenues (i.e. revenue for a maximum quality product delivered at a given time). The latter is defined as delivering a maximum quality product at a given time that achieves the full potential market capture. The equation for calculating the loss due to bad quality is

\[ \text{Bad Quality Loss} = \text{Max. Potential Revenue at Same Timing} - \text{Revenue}. \]

The loss due to market delay is computed keeping the quality constant. To neutralize the effect of varying quality, only the time of delay is varied. The loss for a given option is the difference between the revenue for the highest quality product at the first market opportunity and the revenue corresponding to the completion time for the given option (assuming the same highest quality). It is calculated with

\[ \text{Market Delay Cost} = \text{Max. Potential Revenue} - \text{Revenue}. \]

Figure 11 shows the experimental results for an 80 KSLOC product, fully compressed development schedules and a 3-year revenue timeframe for different reliability options. The resultant sweet spot corresponds to reliability—high accounting for delay losses, reliability losses and development cost for a 3-year time horizon. Intermediate calculations for the loss components are provided in [2].

The sweet spot depends on the applicable time horizon, among other things. The horizon may vary due for several reasons such as another planned major upgrade or new release, other upcoming changes in the business model, or because investors mandate a specific timeframe to make their return.

The experiment was re-run for typical time horizons of 2, 3 and 5 years using a profit view. The results illustrate that the sweet spot moves from reliability equals low to high to very high respectively. It is evident that the optimal reliability depends on the time window. A short-lived product (a prototype is an extreme example) does not need to be developed to as stringent reliability as one that will live in the field longer.

![Figure 11. Calculating reliability sweet spot (3-year timeframe)](image)

4. CONCLUSIONS AND FUTURE WORK

It is important to integrate value-based methods into the software engineering discipline to improve processes and maximize software utility. To achieve real earned value, business value attainment must be a key consideration when designing software products and processes. This work shows several ways how software business decision-making can improve with value information gained from simulation models that integrate business and technical perspectives.

The model demonstrates a stakeholder value chain whereby the value of software to end users ultimately translates into value for the software development organization. It also illustrates that commercial process sweet spots with respect to reliability are a balance between market delay losses and quality losses. Quality does impact the bottom line.

The model can be elaborated to account for feedback loops to generate revised product specifications (closed-loop control) including 1) external feedback from user to incorporate new features and 2) internal feedback on product initiatives from an organizational planning and control entity to the software process.

A more comprehensive model would consider long term product evolution and periodic upgrades. Another related aspect to include is general maintenance by adding explicit activities for operational support.

The product defect model can be enhanced with a dynamic version of COQUALMO [5] to enable more constructive insight into quality practices. This would replace the current construct based on the single factor for required software reliability.

Other considerations for the model are in the market and sales sector. The impact of different pricing schemes and varying market assumptions on initial sales and maintenance can all be
explored. Some of these provisions are already accounted for in a proprietary version of the model.

The model application examples were run with idealized inputs for sake of demonstration, but more sophisticated dynamic scenarios can be easily handled to model real situations. For example discrete descoping were shown, but in many instances scope will exhibit continuous or fluctuating growth over time.

More empirical data on the relationships in the model will also help identify areas of improvement. Assessment of overall dynamics includes more collection and analysis of field data on business value and quality measures from actual product rollouts.

5. REFERENCES