

Integrated Modeling of Business Value and Software Processes

Raymond Madachy^{1, 2}

¹ USC Center for Software Engineering
Department of Computer Science, SAL 318
University of Southern California
Los Angeles, CA 90089-0781
madachy@usc.edu

² Cost Xpert Group
2990 Jamacha Rd., #250
San Diego, CA 92019

Abstract. Business value attainment should be a key consideration when designing software processes. Ideally they are structured to meet organizational business goals, but it is usually difficult to integrate the process and business perspectives quantitatively. This research uses modeling and simulation to assess process tradeoffs for business case analysis. A 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. The system dynamics model 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 goals. Examples are shown 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.

1 Introduction

Software-related decisions should not be extricated from business value concerns. Unfortunately, software engineering practice and research frequently lacks a value-oriented perspective. Value-Based Software Engineering (VBSE) seeks to integrate value considerations into current and emerging software engineering principles and practices [1].

This macroprocess research from [2] addresses the planning and control aspect of VBSE to manage the value delivered to stakeholders. Techniques to model cost, schedule and quality are integrated with business case analysis to allow tradeoff studies in a commercial software development context. Business value is accounted for in terms of return-on-investment (ROI) of different product and process strategies.

Many current practices are done in a value-neutral setting, such as standard earned-value techniques that track cost and schedule but not stakeholder or business value. The latter can be considered the “real” earned value. A value-oriented approach provides explicit guidance for making products useful to people by considering different people’s utility functions or value propositions. The value propositions are used to determine relevant measures for given scenarios.

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 were developed and are included in the integrated model.

It is a challenge to tradeoff different software attributes and particularly between different perspectives such as business and software development. Software process modeling and simulation can be used to reason about 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. A related approach to VBSE is the iDAVE static spreadsheet model used to estimate the ROI of investments in software dependability [3].

Some important elements of VBSE that this research includes are stakeholders’ value proposition elicitation and reconciliation, business case analysis, and value-based monitoring and control. It is assumed in this modeling context that a benefits realization analysis has been performed to substantiate a new product initiative, though the model can clearly be used in this capacity. The stakeholder values of profit and ROI have also already been elicited. These steps have been performed in the field for companies in which this business case model has been applied.

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 Fig. 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.

Fig. 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 num-

ber 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.

Fig. 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).

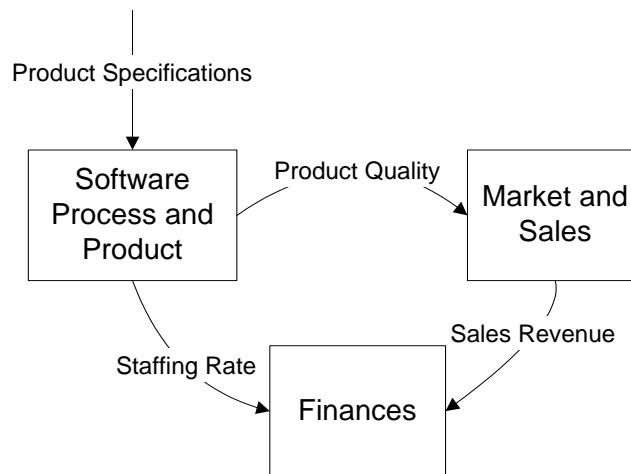


Fig. 1. Model sectors and major interfaces

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 Fig. 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].

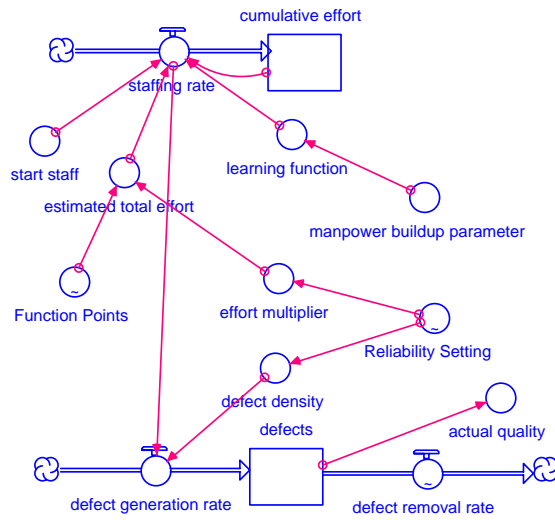


Fig. 2. Software process and product sector

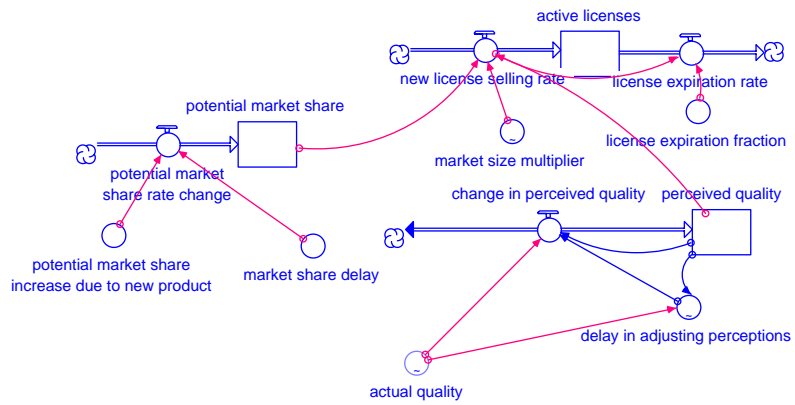


Fig. 3. Market and sales sector

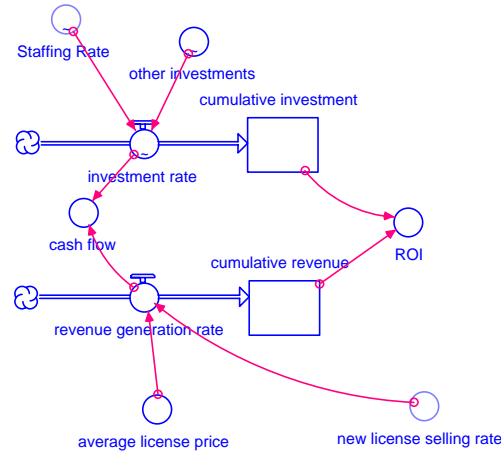


Fig. 4. Finance sector

2.1 Quality Modeling and Value Functions

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. There are four different settings of reliability from low to very high that correspond to four 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.

Several rounds of a continuing Delphi poll of software marketing experts were conducted to help quantify the relative sales impact of different quality levels. Now the mapping between reliability and the relative impact to sales from the Delphi results is captured as a production function and used in the model with the latest refined numbers.

Collectively there are two value-based production functions in the model to describe value relationships (they are illustrated in the first applied example). A market share production function 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 and is needed to vary the sales due to quality.

The value function for actual sale attainment 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. It relates the percent of poten-

tial sales attained in the market against reliability costs. Illustrations of the production functions are shown in the next section.

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 Fig. 5 with a variable delay in adjusting perceptions.

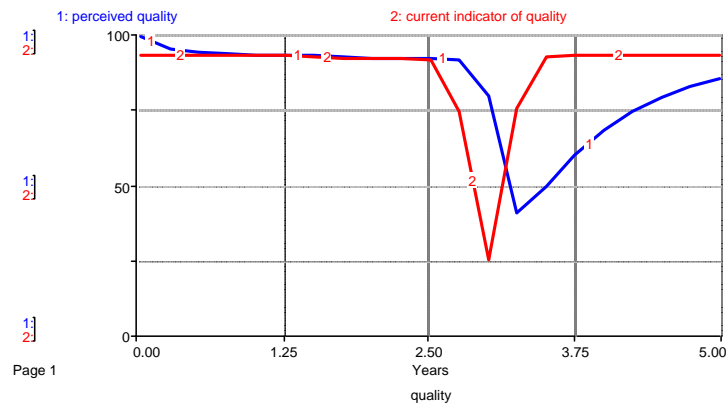


Fig. 5. Perceived quality trends with high and low quality product deliveries

3 Applied Examples

Three representative business decision scenarios are demonstrated next. The first one demonstrates the ability to dynamically assess combined strategies for scope and reliability. The second example looks at strategies of multiple releases of varying quality. Finally the model is used to determine a process sweet spot for reliability.

3.1 Example 1: Dynamically Changing 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 in at least two ways by a decision-maker:

- assessing the impact of changed product specifications during the course of a project

- 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 Fig. 6 relates the potential business value against the cost of development for different feature sets. The actual sales production function against reliability costs is shown in Fig. 7, and it is applied against the potential market capture. The four discrete points correspond to required reliability levels of low, nominal, high and very high. Settings for the three cases are shown in both production functions.

Fig. 8 shows a sample control panel interface to the model. 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.

The reliability settings on the control panel slider are the relative effort multipliers to achieve reliability levels from low to very high. These are input by user via the slider for “Reliability Setting”. The attainable market share derived from the sales production function in Fig. 6 is input by user on the slider “Potential Market Share Increase”.

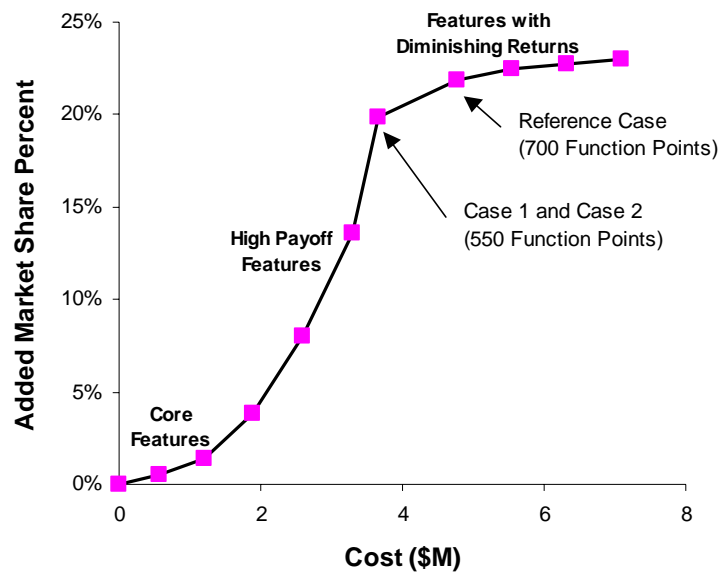


Fig. 6. Market share production function and feature sets

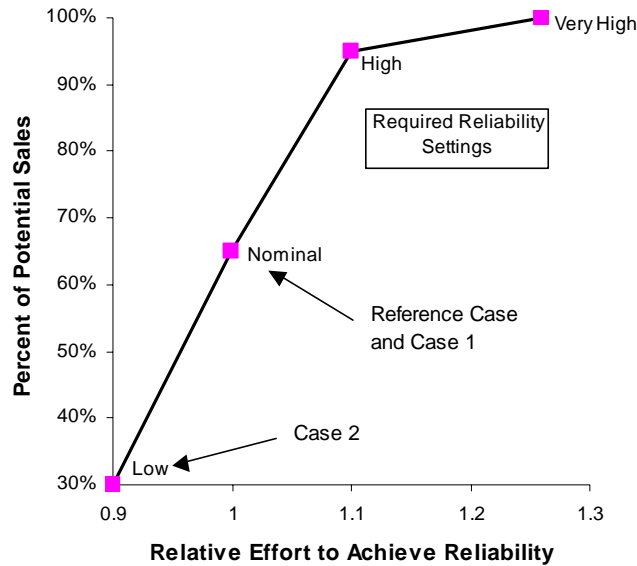


Fig. 7. Sales production function and reliability

Fig. 8 also shows the simulation results for the initial reference case. The default case of 700 function points is delivered with nominal reliability at 2.1 years with a potential 20% market share increase. This project is unperturbed during its course and the 5 year ROI of the project is 1.3.

Trend comparisons between the three cases can be visualized on Figures 8-10. Fig. 9 illustrates the initial case perturbed early to descope low-ROI features (see Fig. 6 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 18%. With lower development costs and earlier delivery the ROI increases substantially to 2.2.

A combined strategy is modeled in Fig. 10. The scope is decreased the same as before in Case 1 (Fig. 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.

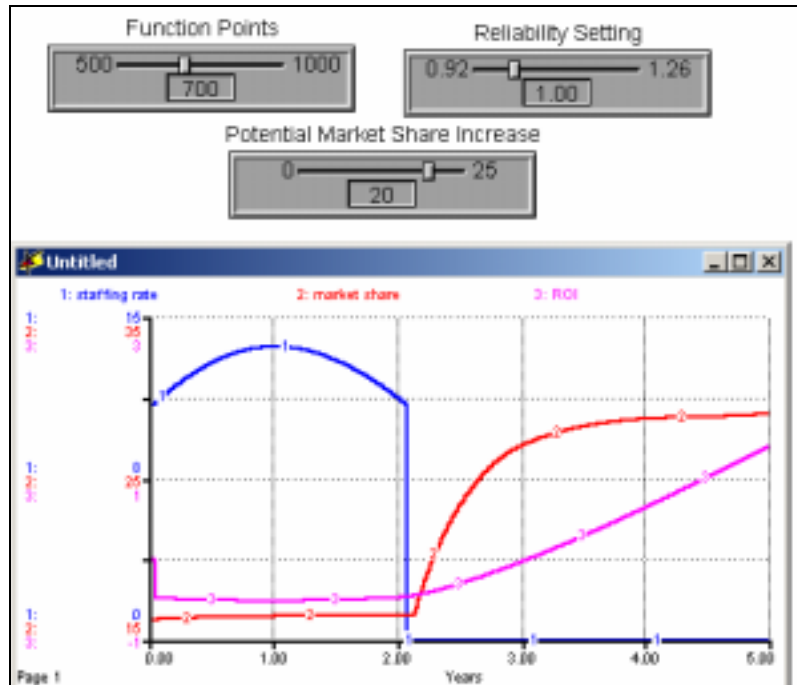


Fig. 8. Sample control panel and reference case (unperturbed)

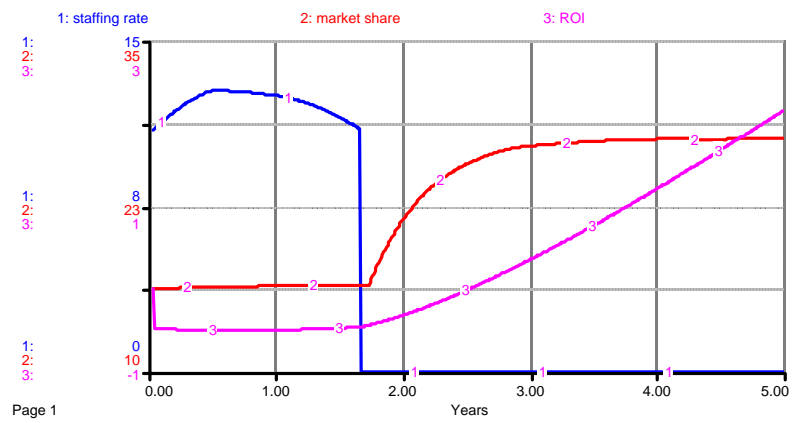


Fig. 9. Case 1 - Descoping of low ROI features at time = .5 Years

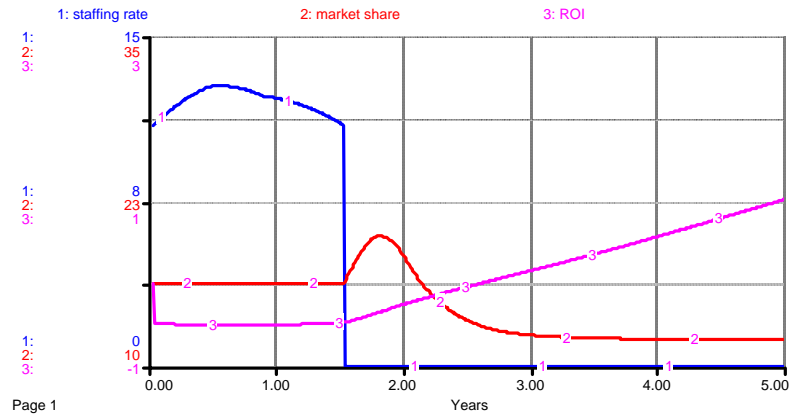


Fig. 10. Case 2 - Descoping of low ROI features and reliability lowering at time = .5 years

Table 1. Case summaries

Case	Delivered Size (Function Points)	Delivered Reliability Setting	Cost (\$M)	Delivery Time (Years)	Final Market Share	ROI
Reference Case: Unperturbed	700	1.0	4.78	2.1	28%	1.3
Case 1: Descope	550	1.0	3.70	1.7	28%	2.2
Case 2: Descope and Lower Reliability	550	.92	3.30	1.5	12%	1.0

3.2 Example 2: Multiple Releases

This example shows a more realistic scenario for maintenance and operational support. Investments are allocated to ongoing maintenance and the effects of additional releases of varying quality are shown.

The reference case contains two product rollouts at years 1 and 3, each with the potential to capture an additional 10% of the market share. These potentials are attained because both deliveries are of high quality as seen in Figures 11-12.

A contrasting case in Figures 13-14 illustrates the impact if the second delivery has poor quality yet is fixed quickly (Fig. 4 shows the quality trends for this case). This results in a change of revenue from \$11.5 M to \$9.6M, ROI from 1.3 to 0.9.

This example is another illustration of the sensitivity of the market to varying quality. Only one poor release in a series of releases may have serious long term consequences.

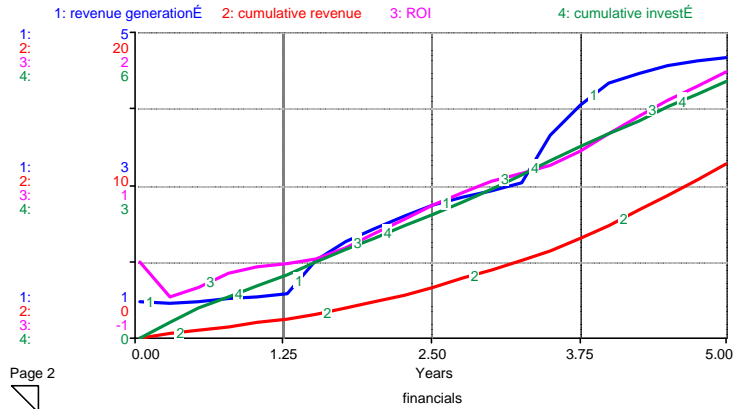


Fig. 11. Reference case financials for two high quality product deliveries

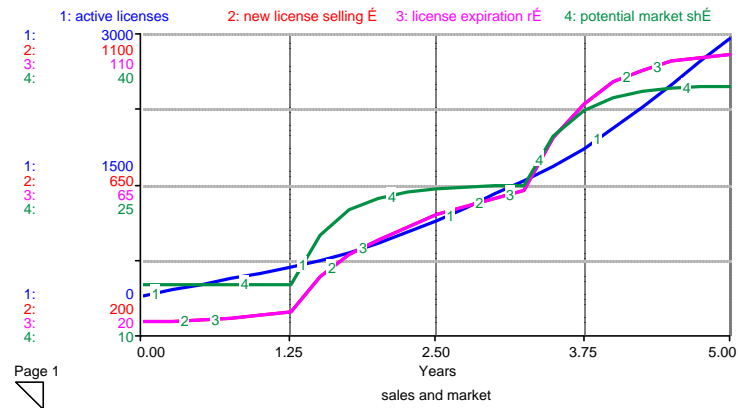


Fig. 12. Reference case sales and market for two high quality product deliveries

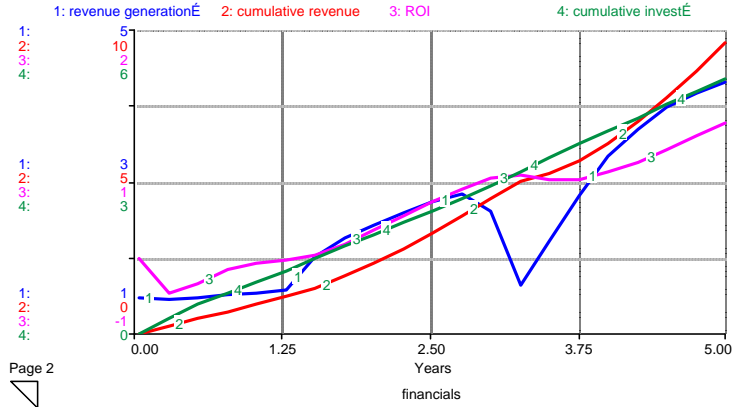


Fig. 13. Financials for high and low quality product deliveries

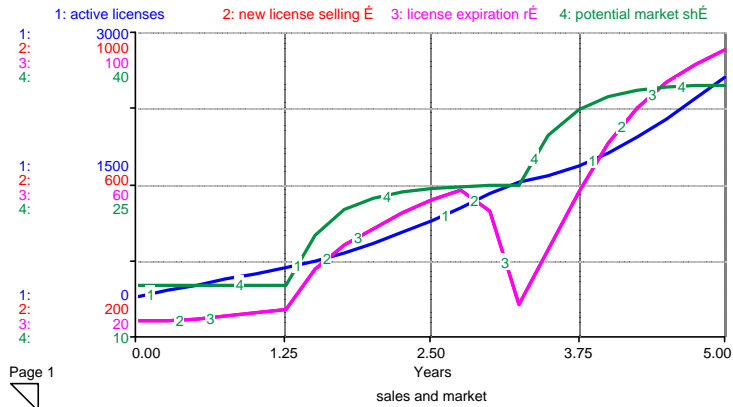


Fig. 14. Sales and market for high and low quality product deliveries

3.3 Example 3: Finding the Sweet Spot

This example derived from [2] shows how the value-based product model can support software business decision-making by using risk consequence to find the quality sweet spot with respect to ROI. The following analysis 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

$$\text{Bad Quality Loss} = \text{Maximum Potential Revenue with Same Timing} - \text{Revenue.} \quad (1)$$

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

$$\text{Market Delay Cost} = \text{Maximum Potential Revenue} - \text{Revenue.} \quad (2)$$

Fig. 15 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. The total cost consisting of delay losses, reliability losses and development cost is minimum at that setting for a 3-year time horizon. Details of the 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 cost view is transformed into a profit maximization view by accounting for revenues). The results are shown in Fig. 16.

The figure illustrates that the sweet spot moves from reliability equals low to high to very high. 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.

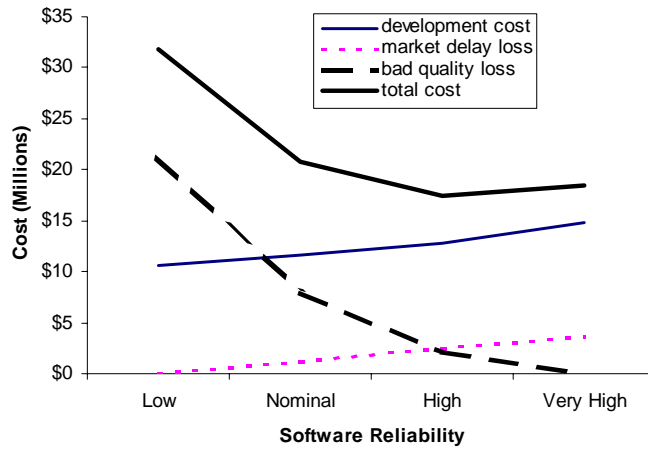


Fig. 15. Calculating reliability sweet spot (3-year timeframe)

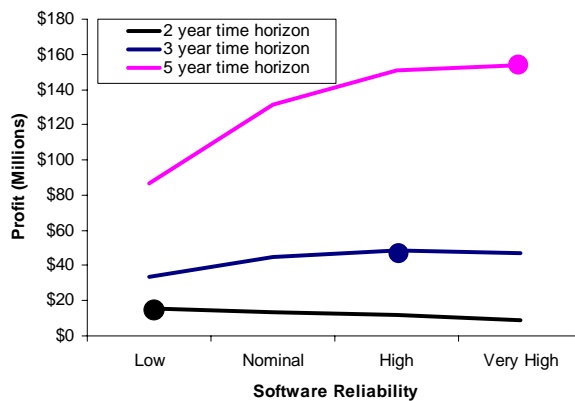


Fig 16. Reliability sweet spot as a function of time horizon

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). This feedback includes:

- external feedback from user to incorporate new features
- 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 descopings 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 software product rollouts.

References

1. Boehm, B., Huang, L.: Value-Based Software Engineering: A Case Study. *IEEE Software*, Vol. 20, No. 2 (2003) 33-41
2. Madachy R.: *Software Process Dynamics*. IEEE Computer Society Press, Washington D.C. (2005)
3. Boehm, B., Huang, L., Jain, A., Madachy R.: Reasoning about the ROI of Software Dependability: the iDAVE Model. *IEEE Software*, Vol. 21, No. 3 (2004) 54-61
4. Boehm B., Abts C., Brown W., Chulani S., Clark B., Horowitz E., Madachy R., Reifer D., Steece B.: *Software Cost Estimation with COCOMO II*. Prentice-Hall (2000)
5. Chulani S., Boehm B.: Modeling software defect introduction and removal: COQUALMO (CONstructive QUALity MOdel). USC-CSE Technical Report 99-510, (1999)