Value-Based Product Modeling

Ray Madachy\textsuperscript{1, 2}

\textsuperscript{1}USC Center for Software Engineering
\textsuperscript{2}Cost Xpert Group

USC-CSE Annual Research Review
March 15, 2005
Outline

• Model Overview
• Product Quality and Stakeholder Value Modeling
• Application Examples
  – Example 1: Dynamically Changing Scope and Reliability
  – Example 2: Determining the Reliability Sweet Spot
• Conclusions and Future Work
Model Background

- **Purpose**: Support software business decision-making by experimenting with product strategies and development practices to assess *real* earned value

- **Description**: System dynamics model relates the interactions between product specifications and investments, software processes including quality practices, market share, license retention, pricing and revenue generation for a commercial software enterprise
Model Features

- A Value-Based Software Engineering (VBSE) model covering the following VBSE elements:
  - Stakeholders’ value proposition elicitation and reconciliation
  - Business case analysis
  - Value-based monitoring and control
- Integrated modeling of business value, software products and processes to help make difficult tradeoffs between perspectives
  - Value-based production functions used to relate different attributes
- Addresses the planning and control aspect of VBSE to manage the value delivered to stakeholders
  - Experiment with different strategies and track financial measures over time
  - Allows easy investigation of different strategy combinations
- Can be used dynamically before or during a project
  - User inputs and model factors can vary over the project duration as opposed to a static model
  - Suitable for actual project usage or “flight simulation” training where simulations are interrupted to make midstream decisions
Model Sectors and Major Interfaces

- Software process and product sector computes the staffing and quality over time
- Market and sales sector accounts for market dynamics including effect of quality reputation
- Finance sector computes financial measures from investments and revenues
Software Process and Product

effort and schedule calculation with dynamic COCOMO variant

product defect flows
Finances, Market and Sales

investment and revenue flows

software license sales

market share dynamics including quality reputation
Outline

• Model Overview

• Product Quality and Stakeholder Value Modeling

• Application Examples
  – Example 1: Dynamically Changing Scope and Reliability
  – Example 2: Determining the Reliability Sweet Spot

• Conclusions and Future Work
Quality Assumptions

- COCOMO cost driver *Required Software Reliability* is a proxy for all quality practices
- Resulting quality will modulate the actual sales relative to the highest potential
- Perception of quality in the market matters
  - Quality reputation quickly lost and takes much longer to regain (bad news travels fast)
  - Modeled as asymmetrical information smoothing via negative feedback loop

![Graph with labeled axes: Perceived Quality (1) and Current Indicator of Quality (2)]
Market Share Production Function and Feature Sets

- **Core Features**
- **High Payoff Features**
- **Features with Diminishing Returns**

Cases from Example 1

Reference Case (700 Function Points)

Case 1 and Case 2 (550 Function Points)
Sales Production Function and Reliability

- **Reference Case** and Case 1
- **Case 2**
- **Low**, **Nominal**, **High**, **Very High**

- **Percent of Potential Sales**:
  - Low: 30%
  - Nominal: 60%
  - High: 90%
  - Very High: 100%

- **Relative Effort to Achieve Reliability**:
  - 0.9
  - 1.0
  - 1.1
  - 1.2
  - 1.3

**Cases from Example 1**

**Required Reliability Settings**
Outline

• Model Overview
• Product Quality and Stakeholder Value Modeling
• Application Examples
  – Example 1: Dynamically Changing Scope and Reliability
  – Example 2: Determining the Reliability Sweet Spot
• Conclusions and Future Work
Example 1: Dynamically Changing Scope and Reliability

- Shows how model can assess the effects of combined strategies by varying the scope and required reliability independently or simultaneously
- Simulates midstream descoping, a frequent strategy to meet time constraints by shedding features
- Three cases are demonstrated:
  - Unperturbed reference case
  - Midstream descoping of the reference case after ½ year
  - Simultaneous midstream descoping and lowered required reliability at ½ year
Control Panel and Simulation Results

Function Points

- 500
- 1000
- 700

Reliability Setting

- 0.92
- 1.00
- 1.26

Potential Market Share Increase

- 0
- 20
- 25

Unperturbed Reference Case

Descope

Case 1

Descope + Lower Reliability

Case 2
## 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: Unperturbed</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 at Time = ½ years</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 at Time = ½ years</td>
<td>550</td>
<td>0.92</td>
<td>3.30</td>
<td>1.5</td>
<td>12%</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Example 2: Determining the Reliability Sweet Spot

• Analysis process
  – Vary reliability across runs
  – Use risk exposure framework to find process optimum
  – Assess risk consequences of opposing trends: market delays and bad quality losses
  – Sum market losses and development costs
  – Calculate resulting net revenue

• Simulation parameters
  – A new 80 KSLOC product release can potentially increase market share by 15%-30% (varied in model runs)
  – 75% schedule acceleration
  – Initial total market size = $64M annual revenue
    • vendor has 15% of market
    • overall market doubles in 5 years
Cost Components

3-year time horizon

Cost (Millions)

Software Reliability

Cost Components

development cost
market delay loss
bad quality loss
total cost

Low
Nominal
High
Very High

©USC-CSE
Sweet Spot Depends on Time Horizon

Software Reliability

- Low
- Nominal
- High
- Very High

Profit (Millions)

- 2 year time horizon
- 3 year time horizon
- 5 year time horizon

©USC-CSE
Outline

• Model Overview
• Product Quality and Stakeholder Value Modeling
• Application Examples
  – Example 1: Dynamically Changing Scope and Reliability
  – Example 2: Determining the Reliability Sweet Spot
• Conclusions and Future Work
Conclusions

- To achieve real earned value, business value attainment must be a key consideration when designing software products and processes.
- Software enterprise decision-making can improve with information from simulation models that integrate business and technical perspectives.
- Optimal policies operate within a multi-attribute decision space including various stakeholder value functions, opposing market factors and business constraints.
- Risk exposure is a convenient framework for software decision analysis.
- Commercial process sweet spots with respect to reliability are a balance between market delay losses and quality losses.
- Model demonstrates a stakeholder value chain whereby the value of software to end-users ultimately translates into value for the software development organization.
Future Work

• Enhance product defect model with dynamic version of COQUALMO to enable more constructive insight into quality practices
• Add maintenance and operational support activities in the workflows
• Elaborate market and sales for other considerations including pricing scheme impacts, varying market assumptions and periodic upgrades of varying quality
• Account for feedback loops to generate product specifications (closed-loop control)
  – External feedback from users to incorporate new features
  – Internal feedback on product initiatives from organizational planning and control entity to the software process
• More empirical data on attribute relationships in the model will 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