Conflict Analysis and Negotiation Aids for Cost-Quality Requirements

Barry Boehm and Hoh In
(boehm, hoin)@sunet.usc.edu
Voice: (213) 740-8163; Fax: (213) 740-4927
Center for Software Engineering and Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781, U.S.A.

Abstract

The process of resolving conflicts among software quality requirements is complex and difficult because of incompatibilities among stakeholders' interests and priorities, complex cost-quality requirements dependencies, and an exponentially increasing resolution option space for larger systems. This paper describes an exploratory knowledge-based tool, the Software Cost Option Strategy Tool (S-COST), which assists stakeholders to 1) surface appropriate resolution options for cost-quality conflicts; 2) visualize the options; and 3) negotiate a mutually satisfactory balance of quality requirements and cost.

S-COST operates in the context of the USC-CSE WinWin system (a groupware support system for determining software and system requirements as negotiated win conditions), QARCC (Quality Attribute and Risk Conflict Consultant - a support system for identifying quality conflicts in software requirements), and COCOMO (CONstructive COST estimation MОdel). Initial analyses of its capabilities indicate that its semiautomated approach provides users with improved capabilities for addressing cost-quality requirements issues.

1. Introduction

1.1 Evolving Concepts of Software Quality

Much of the early institutional focus on software quality was initiated by the U.S. Department of Defense (DoD), which had numerous quality problems with its large software systems. Consistent with its focus on requirements-driven, contract-oriented waterfall-model software development, its major 1974 standard, MIL-S-52779, "Software Quality Assurance Program Requirements," (DoD, 1974), defined the objective of software QA as, "to assure that the software delivered under the contract meets the requirements of the contract."

The major pitfall of this approach is that if your contract specified poor quality software, your software QA program would assure that you got poor quality software. This happened to DoD and commercial organizations in numerous ways: specifying poor user interfaces, specifying requirements obtained from the wrong users, getting unmaintainable software by neglecting to specify maintenance and diagnostic requirements.

Based on the work of Deming, Juran, and others, the 1980's saw a trend away from the 1970's contract-oriented specification compliance toward service-oriented customer satisfaction as the primary quality objective. Approaches such as Total Quality Management (Deming, 1989) and Quality Function Deployment (Eureeka-Ryan, 1988) based quality on "the voice of the customer," which was generally interpreted to include the product's users. Thus, the 1990 definition of "quality" in the IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990) added "... meets customer or user needs or expectations" to its earlier definition of "...meets specified requirements."

The major difficulty with the customer-satisfaction approach is that customers often have a poor grasp of the tradeoffs and interactions among the qualities they are interested in, and often neglect qualities such as maintainability which affect them only indirectly. Very often, customers have pushed overly ambitious performance objectives which led to unaffordable and/or unmaintainable software systems, or have pushed to adopt a poorly-architected prototype with nice usability features but poor scalability, dependability, and/or portability.

Initiatives to address these problems in the 1990's have focused on identifying the full set of key stakeholders in a software system and pursuing the objective of negotiated stakeholder win-win relationships among software quality attributes. This expands the scope of "quality" to include the voice of the buyer on cost or affordability, the voice of the maintainer on modifiability, the voice of neighboring stakeholders on interoperability, and others such as the voice of the general public on safety or privacy. This has led to new organizational approaches such as Integrated Product Teams, expanded versions of QFD (Pardee, 1996), and process approaches such as the WinWin Spiral Model (Boehm et al., 1995a).

1 This paper will be published in a new journal, Software Quality Professional (January 1998).
1.2 Supporting Systems for Emerging Software Quality Concepts

In order to support emerging 1990's concepts of quality as a stakeholder-negotiated win-win relationship among quality attributes (including cost as "affordability" and schedule as "timeliness"), one needs a support framework for resolving conflicts among quality attributes.

The conflict resolution process for the right balance of quality requirements is complex and difficult due to the following obstacles:

- **Difficulties in coordinating multiple stakeholders' interests and priorities.** Users feel that full functionality, dependability, and ease of use are the most important attributes. Most concerns of Customers are cost and schedule. Developers are usually mostly concerned with low project risk and reusing assets. Maintainers are strongly concerned with good diagnostics and easy maintenance. Finding the middle ground among these requirements commitments is difficult.

- **Complicated dependencies and tradeoff analyses between quality attributes.** Every decision to improve some quality attributes may impact others, particularly the cost and schedule. Some requirement decisions may be not compatible with others.

- **Exponentially increasing resolution option space.** In order to resolve a conflict involving a cost overrun, several items should be considered. For example, which modules should be reduced and by how much to get the project back on track? Which modules can be degraded in terms of their quality attributes? How much of which qualities should be degraded?

Given the overall scarcity of software expertise and the complexity of cost-quality conflict resolution, it is worth trying to capture such expertise and make it more broadly available via automated aids for cost-quality conflict resolution.

At least three major capabilities are necessary to resolve cost-quality conflicts among software requirements:

- **A general capability to surface and negotiate cost conflicts and risks among requirements.** The USC-CSE WinWin system [Boehm et al., 1994; 1995a] provides an example of such a capability.

- **Capabilities to support the resolution of cost conflicts with functional quality requirements based on early information.** The following aids provide examples of such capabilities:
  - Aids for identifying cost conflicts with functional requirements. The Cocomo (Constructive Cost estimation Model) [Boehm, 1981; Boehm et al., 1995b] provides an example of such a capability.
  - Aids for identifying cost conflicts with quality requirements based on early information. The QARCC (Quality Attribute Risk and Conflict Consultant) tool [Boehm-In. 1996a] provides an example of such a capability.

- **Capabilities to generate, visualize, and negotiate potential resolution options for cost conflicts.** The S-COST system described in this paper operates on the win conditions captured by the WinWin system and the results of COCOMO and QARCC analyses to provide such a capability.

This paper discusses S-COST as a model and support system for analyzing and negotiating cost-quality conflicts among software requirements. It will present the S-COST context (section 2), concept of operation (section 3), primary cost options and stakeholder relationships (section 4), S-COST visualization and negotiation aids (section 5), related work (section 6), and conclusions (section 7).
between them. Stakeholders begin by entering their Win Conditions, using a schema provided by the WinWin system. If a conflict among stakeholders' Win Conditions is determined, an Issue schema is composed, summarizing the conflict and the Win Conditions it involves.

For each Issue, stakeholders prepare candidate Option schemas addressing the Issue. Stakeholders then evaluate the Options, iterate some, agree to reject others, and ultimately converge on a mutually satisfactory (i.e., win-win) Option. The adoption of this Option is formally proposed and ratified by an Agreement schema, including a check to ensure that the stakeholders' iterated Win Conditions are indeed covered by the Agreement.

Our experience with WinWin usage indicates that, as applications reach the size of several dozen Win Conditions, it becomes hard for stakeholders to identify the likely conflicts among them. Thus, we have been experimenting with automated aids such as QARCC and S-COST to help stakeholders identify Issues and formulate Options for resolving them.

2.2 QARCC

QARCC is an exploratory knowledge-based tool for identifying potential conflicts and risks among quality requirements early in the software life cycle.

QARCC uses the "Attributes" portion of WinWin's domain taxonomy to identify potential quality attributes conflicts. As stakeholders enter Win Conditions, they identify which domain taxonomy elements are relevant.

The top-level quality attributes in the WinWin domain taxonomy are shown at the bottom of figure 3. Suppose a stakeholder enters a Win Condition schema and puts "Assurance" in the Taxonomy Elements slot. QARCC will then draw on its knowledge base to analyze potential conflicts between Assurance and other quality attributes. It will then notify the affected stakeholders of the potential conflicts.

To determine which stakeholders to notify of a potential conflict between Assurance and (say) Interoperability, QARCC uses the Stakeholder/Quality-Attribute Relationships shown in figure 3. In this case, QARCC would notify the Interoperator and User stakeholders of the potential conflict, as they are the stakeholders generally most concerned with Interoperability.

QARCC is good for making top-level suggestions about potential quality attribute conflicts, but it lacks detail. S-COST is an effort to provide such detail in the area of quality attribute conflicts involving cost.

The S-COST concept of operation is shown in figure 4. Using the WinWin system, stakeholders enter their new Win Conditions. These may involve functions, quality goals or constraints. As shown in Screen 1 of figure 5, Win Condition schemas have attributes such as Priority and domain Taxonomy Elements. For Win Conditions with quality attribute and cost/schedule Taxonomy Elements, QARCC examines its architectural and process strategies to search for potential conflicts. For example, Layering the architecture to meet the Portability Win Condition in Screen 1 of figure 5 produces likely conflicts with Cost/Schedule and Performance (Screen 2 of figure 5; in the initial version of QARCC, Cost and Schedule were combined into Development Affordability, and Performance was called Efficiency). QARCC then generates a draft Issue identifying this potential conflict for stakeholders to consider (Screen 3 of figure 5).
Continuing with the scenario in figure 4, once stakeholders are presented with a set of draft Issues from QARCC involving cost, they can then use COCOMO to analyze the potential cost conflicts identified. If the resulting estimated cost exceeds the target cost, a stakeholder enters this as a WinWin Issue whose solution needs to be negotiated by the stakeholders.

As indicated in figure 4, S-COST operates on the Issue and COCOMO estimate information to:

- Suggest options for resolving cost issues;
- Notify affected stakeholders of their availability and implications;
- Provide visualization and negotiation aids for cost issues/opinion resolution.

After the stakeholders converge on a mutually satisfactory (win-win) combination of options, they draft a WinWin Agreement schema, and follow WinWin’s procedures for voting on and adopting the Agreement.

The next section discusses the portions of the S-COST knowledge base supporting cost-resolution option generation and stakeholder notification. Section 5 then illustrates these and S-COST’s option analysis and negotiation support capabilities via an example.

4. Primary cost options and stakeholder relations

Screen 1: A new Win Condition entered by Stakeholder

Screen 2: Potential conflicts identified by QARCC

Screen 3A A draft issue suggested by QARCC

Table 1 shows the top-level option strategies for resolving software cost (and most schedule) issues. The strategies are primarily based on analysis of COCOMO cost drivers (Boehm et al., 1995b) whose labels are shown in column 2 of table 1.

- Reduce/defer functionality, which reduces cost by reducing the program (KDSI) and/or database (DATA) size, at least for the Initial Operational Capability (IOC).
- Reduce/defer software quality, which reduces cost by relaxing reliability (RELY) or performance (TIME) constraints, by forgoing the complexities (CPLX) of providing such capabilities as graceful degradation or information security, or by reducing documentation (DOCU).
- Improve tools, techniques or platform, via a platform that is more powerful (TIME, STOR) or stable (PVOL); or via better software tools (TOOL).
- Relax the delivery schedule constraint (SCED).
- Improve personnel capabilities, via more capable teams of analysts and programmers (ACAP, PCAP); stronger applications, platform, or language/tool experience (AEXP, PEXP, LTEX); or equivalent but lower-cost personnel ($K/MM).

Barry Boehm and Hal In
Conflict Analysis and Negotiation Aids for Cost-Quality Requirements
• Reuse software assets, by reusing or adopting more software assets (ADSI), with lower level(s) of design, code, or integration modification (DM, CM, IM).
• Improve coordination of multiple project stakeholders as a team, via synchronizing the project stakeholders by reconciliation of different objectives and culture and having experience in operating as a team (TEAM).
• Architecture and risk resolution, which reduces rework cost through risk assessment and control, architecture thoroughness, including verification of architectural specifications (RESL).
• Improve process maturity level, which reduces cost through SEI (Software Engineering Institute's CMM (Capability Maturity Model) (PMAT). Exercising this option involves choice of alternative suppliers, as process maturity cannot be increased instantaneously.
• Improve preconditions and development flexibility, which reduces cost via improving organizational understanding of project objectives and experience working with related software (PREC), or via eliminating need for software conformance with pre-established requirements and with external interface specifications (FLEX).
• Increase budget. This can be justified if the current win condition budget is insufficient to generate a critical-mass competitive product, or if the return on investment (ROI) is sufficiently increased. Of course, this option strategy cannot be used if added funds are not available (These points illustrate the option strategy's Pros and Cons shown in columns 3 and 4 of Table 1).

Table 1. The Attributes of Cost-Resolution Option Strategies

<table>
<thead>
<tr>
<th>Option Strategy</th>
<th>COCOMO Factor</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce/defer Functionality</td>
<td>KSIS, DATA</td>
<td>Reduce cost, IOC, and schedule</td>
<td>Smaller product to maintain</td>
</tr>
<tr>
<td>Reduce/defer Quality</td>
<td>RELY, CPLX, TIME, DOCU</td>
<td>Reduce cost, schedule, and complexity</td>
<td></td>
</tr>
<tr>
<td>Improve tools, techniques, platform</td>
<td>TIME, STRC, PVCL, TO, SILE</td>
<td>Reduce SW cost and schedule</td>
<td>Improve maintainability and other qualities</td>
</tr>
<tr>
<td>Relax schedule constraint</td>
<td>SCED</td>
<td>Reduce cost if schedule was tight</td>
<td></td>
</tr>
</tbody>
</table>

4.1 Formalism of Option Strategies

Table 1 shows the informal level of suggestions about how to create the options for resolving cost conflicts. However, in order to provide automated support for resolving cost conflicts in a specific project situation, a more detailed and formalized structure for S-COST resolution option strategies is necessary. The formalized structure also helps stakeholders communicate each other effectively for achieving agreements.

The formalized structure used by S-COST is based on information of COCOMO cost drivers and function elements:

Option-Strategy Name (Module Name). COCOMO cost driver, the value before the strategy is applied, the value after the strategy is applied, the reduced schedule.

-projects long better people will suffer Potential staffing difficulties and delays
-users may lose quality capabilities
-risk of overestimating costs
-uncontrollable for some situations
-added budget for risk management is necessary
-may enable product to reach more competitive critical mass
-added funds may not be available
Each stakeholder suggests how much the COCOMO cost drivers for the Option Strategies should be changed (e.g., COCOMO cost driver, the value before and after the Strategies are applied). Given the information taken from stakeholders, S-COST produces the pros and cons of the Strategies using the knowledge base for the Option Strategies and the reduced cost and schedule using the COCOMO. The knowledge base for suggesting the pros and cons of the Strategies contains the default knowledge, but it can be refined and situated from project to project.

4.2 Stakeholder/Option Strategy Relationships

Reducing reliability or performance may be acceptable options for some stakeholders but not for others. S-COST uses the cost option/stakeholder relationship (figure 6) to notify the appropriate stakeholders of options which may have first-order consequences for them. For some of the options in Table 1, figure 6 shows the stakeholders who would generally be directly concerned with the exercise of the option. Thus, for example, “Increase budget” can potentially affect any of the stakeholders by providing them more capability, but the directly-concerned stakeholder is the Customer, who must find a way to justify and obtain the budget increase.

5. S-COST analysis and negotiation aids

This section describes an initial S-COST prototype based on COCOMO 81 [Boehm, 1981] and reported in [Boehm-In, 1996b]. It also discusses current improvements we have been making to S-COST based on COCOMO II and feedback on the prototype from the USC Center for Software Engineering’s industry and government affiliates.

In a hypothetical, but representative project, called “Strikeware”, the S-COST analysis and negotiation aids will be illustrated with respect to a satellite data processing scenario. In the scenario, the user, customer, and developer of a system have negotiated a $5 million, 16-month upgrade to add Satellite Surveillance data services to a Mission Data Integration Facility (MDIF). The Strikeware user has determined that it will be important to add weather data services to the MDIF upgrade. A COCOMO estimate of the resulting added software indicates a $6.66M cost and 17.6 month schedule, but the customer is strongly constrained to keep to the original $5M cost and 16 month schedule.

As seen in figure 7, the initial S-COST prototype has several capabilities to help the user, customer, and developer determine cost reduction options, visualize their impact on the problem situation, and negotiate a new win-win solution. For example, the Visualization window for option generation in figure 7 shows the cost reduction target of $5M as a ▼ mark. It uses data on the priorities of the stakeholders’ functional module win conditions, and the corresponding COCOMO estimates of the module’s cost contribution, to produce a display of module cost contributions by priority.

Using this display, the stakeholders could simply agree to drop or defer the lowest-priority modules until the cost target is reached. But there may be better options. For example, the stakeholders split a module into higher and lower priority modules via the Operations button; or adjust other cost drivers such as personnel capability and experience, improved tools, or software reuse via the Strategies button. Or, if justified and feasible, the customer could increase the budget, raising the target cost in the Target Cost field (figure 7).

5.1 Visualization window features for option generation

The S-COST Visualization window in figure 7 has a number of option strategies and display aids for stakeholders. Some selected features are highlighted below.

Function Elements (FEs) in the FE window (right top) are sorted by the priority using “Sort-by-Priority”, one of the Operations buttons, based on the user-determined FEs. COCOMO-estimated cost and schedule appear in the left window. Status Area, along with User’s Id & Role, Priority of Option, and target cost & schedule. The Operations button is available to enable stakeholders to
5.2 Visualization window features for option negotiation

Another visualization aid (obtained via the Negotiation Aid button in figure 7) provides the Option Strategy list (figure 9), which summarizes the strategy of negotiating a combination of Option Strategies suggested by the various stakeholders. The Options can be displayed in order of originating stakeholders, the Option Strategies' priority, or the type of the Option Strategy via the Options button. Stakeholders can revise their options after considering other stakeholders' strategies and their priorities. This helps stakeholders reach a win-win combination of Cost-Resolution Strategies.

The stakeholders continue to interact with S-COST and each other until they converge on a win-win cost reduction strategy with no win-lose side effects.
6. Related work

Early characterizations of software quality attribute relationships were developed in [Boehm et al., 1976] and [McCall et al., 1977]. Rome Laboratory sponsored a number of followons to the McCall study, including [Bowen et al., 1985; Lasky-Donaghy, 1993; and Murine, 1995]. More recent work includes the manual scenario-based Software Attribute Analysis Method, SAAM [Kazman et al., 1994], further CMU-SEI work on software quality attribute relationships [Kazman-Bass, 1994], and efforts at Univ. of Toronto to provide automated assistance in dealing with interactions among non-functional requirements [Chung et al., 1995].

7. Conclusions

We have done a comparative analysis of the options surfaced by stakeholders in the initial WinWin Strikeware exercise and the options generated and analyzed by S-COST. S-COST provides a more thorough set of candidate cost-quality conflict resolution options and analysis of their pros and cons. On the other hand, S-COST could not generate situation-specific options, such as the User deciding to break up the Query/Display module into higher and lower priority submodules. Given these strengths and limitations of automated approaches, we conclude that S-COST's resulting semi-automated approach is stronger than a heavily manual approach or a heavily automated approach.

Based on analysis a number of additional WinWin-based requirements negotiations, we have found that stakeholders are frequently unaware of attractive options to resolve cost/quality/functionality conflicts. The S-COST prototype and its current extensions can enable stakeholders to identify, analyze, and obtain experience-based advice on better ways to resolve such conflicts.

Acknowledgments


References


Conflict Analysis and Negotiation Aids for Cost-Quality Requirements