Conflict Analysis and Negotiation Aids for Cost-Quality Requirements

Barry Boehm and Hoh In
(boehm, hohin)@sunset.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 MOdel). Initial analyses of its capabilities indicate that its semiautomated approach provides users with improved capabilities for addressing cost-quality requirements issues.

Keywords: Requirements engineering, software quality attributes, risk, conflict resolution, negotiation, software cost analysis

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 [Eureka-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].

1.2 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. The primary 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 among 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 and 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).

2. Context

S-COST operates in the context of the USC-CSE WinWin system (Section 2.1) and the QARCC tool (Section 2.2) to extend their capabilities.

2.1 WinWin

WinWin is a groupware support system for determining software requirements as negotiated win conditions. It is based on the WinWin Spiral Model [Boehm et al., 1994; 1995a] which uses Theory W [Boehm-Ross, 1989] to generate the stakeholder win-win situation incrementally through the Spiral Model. WinWin assists the identified stakeholders in identifying and negotiating issues (i.e., conflicts among their win conditions), since the goal of Theory W, "Make everyone a winner," involves stakeholders identifying their win conditions (sector 2 in figure 1), and reconciling conflicts among win conditions (sector 3).

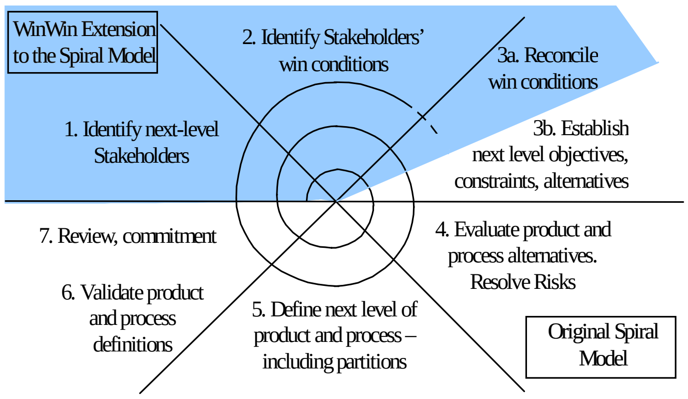

WinWin Extension to the Spiral Model
2. Identify Stakeholders' win conditions
3a. Reconcile win conditions
1. Identify next-level Stakeholders
3b. Establish next level objectives, constraints, alternatives
7. Review, commitment
4. Evaluate product and process alternatives. Resolve Risks
6. Validate product and process definitions
5. Define next level of product and process – including partitions
Original Spiral Model

Figure 1. The WinWin Spiral Model

Figure 2 shows the negotiation model used by WinWin, in terms of its primary schemas and the relationships 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.

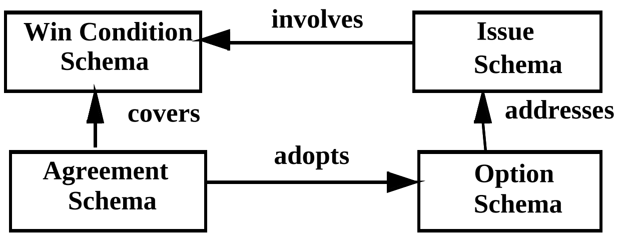


Figure 2. WinWin Negotiation Model

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. Usage experience also indicates that WinWin is not a panacea for all conflict situations, but generally increases stakeholders' levels of cooperation and trust [Boehm et al., 1998]

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.

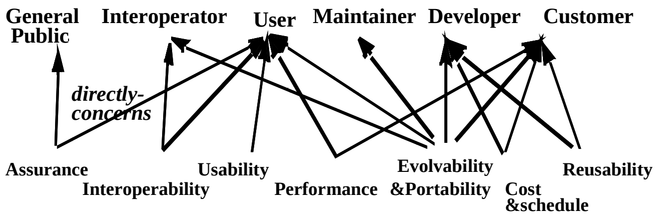


Figure 3. Stakeholder / Quality-Attribute Relationship

3. S-COST Concept of Operation

S-COST extends WinWin and complements QARCC by using an additional software cost knowledge base related to another component of WinWin: the COnstructive COst MOdel, or COCOMO. S-COST uses the COCOMO cost drivers, cost estimates, and related experience base to sharpen QARCC's identification of cost-conflict Issues. It also suggests in-depth cost-conflict resolution Options, and provides Option visualization and negotiation aids to help stakeholders resolve the cost-conflict Issues.

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 [Boehm-In, 1996a] 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).

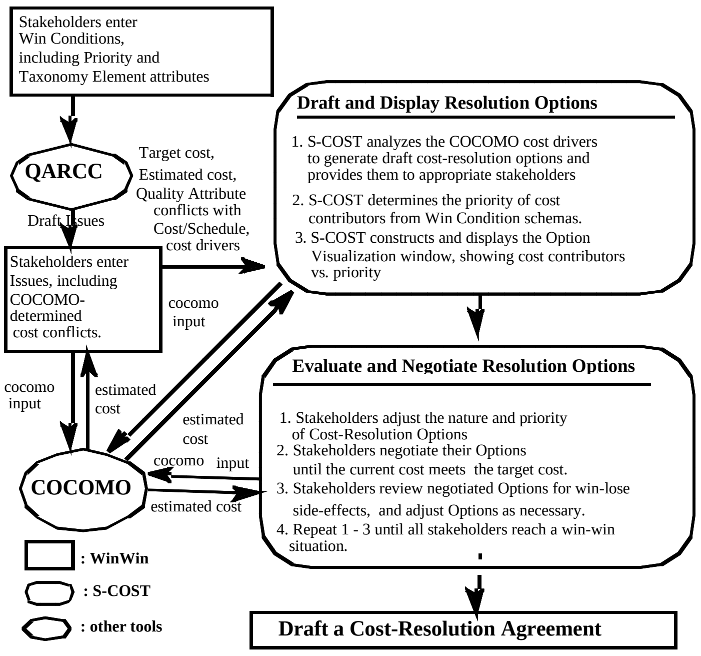
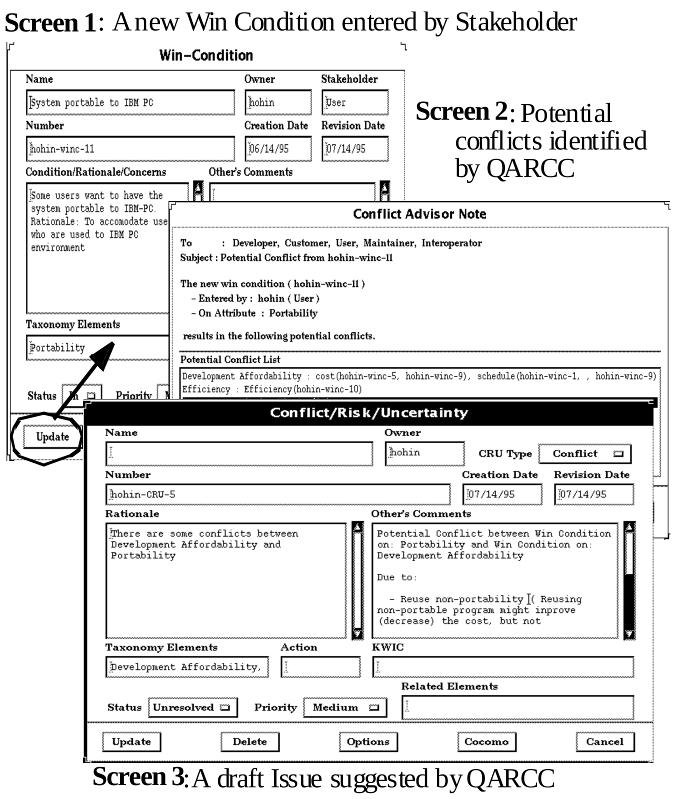

Stakeholders enter Win Conditions, including Priority and Taxonomy Element attributes
QARCC
Target cost, Estimated cost, Quality Attribute conflicts with Cost/Schedule, cost drivers
Draft Issues
Stakeholders enter Issues, including COCOMO-determined cost conflicts.
cocomo input
estimated cost
cocomo input
estimated cost
cocomo input
estimated cost
COCOMO
Draft and Display Resolution Options
1. S-COST analyzes the COCOMO cost drivers to generate draft cost-resolution options and provides them to appropriate stakeholders
2. S-COST determines the priority of cost contributors from Win Condition schemas.
3. S-COST constructs and displays the Option Visualization window, showing cost contributors vs. priority
Evaluate and Negotiate Resolution Options
1. Stakeholders adjust the nature and priority of Cost-Resolution Options
2. Stakeholders negotiate their Options until the current cost meets the target cost.
3. Stakeholders review negotiated Options for win-lose side-effects, and adjust Options as necessary.
4. Repeat 1 - 3 until all stakeholders reach a win-win situation.
Draft a Cost-Resolution Agreement
: WinWin
: S-COST
: other tools

Figure 4. S-COST Concept of Operation


Screen 1: A new Win Condition entered by Stakeholder

Win–Condition

Name: System portable to IBM PC
Owner: hohin
Stakeholder: User
Number: hohin-winc-11
Creation Date: 06/14/95
Revision Date: 07/14/95
Condition/Rationale/Concerns: Some users want to have the system portable to IBM-PC. Rationale: To accomodate use who are used to IBM PC environment
Other's Comments:
Taxonomy Elements: Portability

Screen 2: Potential conflicts identified by QARCC

Conflict Advisor Note

To : Developer, Customer, User, Maintainer, Interoperator
Subject : Potential Conflict from hohin-winc-11

The new win condition (hohin-winc-11)
- Entered by : hohin (User)
- On Attribute : Portability

results in the following potential conflicts.

Potential Conflict List
Development Affordability : cost(hohin-winc-5, hohin-winc-9), schedule(hohin-winc-1, , hohin-winc-9)
Efficiency : Efficiency(hohin-winc-10)

Conflict/Risk/Uncertainty

Name:
Owner: hohin
CRU Type: Conflict
Number: hohin-CRU-5
Creation Date: 07/14/95
Revision Date: 07/14/95
Rationale: There are some conflicts between Development Affordability and Portability
Other's Comments: Potential Conflict between Win Condition on: Portability and Win Condition on: Development Affordability

Due to:

- Reuse non-portability (Reusing non-portable program might inprove (decrease) the cost, but not
Taxonomy Elements: Development Affordability,
Action:
KWIC:
Related Elements:
Status: Unresolved
Priority: Medium

Screen 3: A draft Issue suggested by QARCC

Figure 5. An Example of QARCC

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 issue/option resolution.*

After the stakeholders converge on a mutually satisfactory (win-win) combination of options, they draft an 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

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 II cost drivers [Boehm et al., 1995b] whose labels are shown in column 2 of table 1. The new cost drivers in COCOMO II are underlined in italics.

Each of the strategies in table 1 is characterized by its effect on software cost via the relevant COCOMO II cost drivers, and a set of pros and cons for using the strategy. A good example to discuss is the second strategy: Reduce/defer quality.

The COCOMO II cost drivers RELY and DOCU indicate that a project may reduce its development costs by doing less testing, standards-checking, documentation, etc., but that these will lead to higher maintenance costs. This strategy may be workable for a product with a very short operational lifetime (prototype, special-purpose one-shot analysis), but is not a workable life-cycle cost strategy for longer-lived products.

The execution time constraint (TIME) and required complexity (CPLX) cost drivers indicate that development costs can often be reduced by reducing performance requirements (e.g., 2 second response time vs. 1 second) or by reducing the complexity involved in achieving high levels of system security or survivability. If one needs these high levels, the "Cons" part of the table indicates that these strategies are counterproductive. But in some cases (downstream performance will improve via faster hardware, or overdesigned components that create an unbalanced architecture) they may be warranted.

Option Strategies	COCOMO Parameter	Pros	Cons
Reduce/defer Functionality	KDSI, DATA	- Reduce cost, IOC, and schedule - Smaller product to maintain	- Capabilities unavailable to stakeholders - Need to pay later if deferred
Reduce/defer Quality	RELY, *DOCU* [1], TIME, CPLX	- Reduce development cost and schedule	- Stakeholders lose quality capabilities, increase maintenance costs
Improve tools, techniques or platform	TIME, STOR, PVOL, TOOL, *SITE*	- Reduce s/w cost and schedule - Improve maintainability and other qualities	- Increase tool, training, platform costs - Reducing tool, platform experience would increase software cost
Relax the delivery schedule constraint	SCED	- Reduce cost if schedule was tight	- Defer stakeholders' use of product capabilities
Improve personnel capabilities	ACAP, PCAP, *PCON,* AEXP, PEXP, LTEX, $K/PM	- Reduce cost and schedule - Improve quality from personnel capability and/or application experience - Reduce personnel turnover	- Projects losing better people will suffer - Potential staffing difficulties and delays - Increased cost/person-month unless low-cost outsourcing
Reuse software assets	ADSI, DM, CM, IM	- Reduce cost and schedule - May gain quality if the used assets have good quality	- Users may lose quality capabilities - Risk of overestimating reuse
Improve coordination via teambuilding	*TEAM*	- Reduce cost and schedule by removing the inter-personnel overhead	- Uncontrollable for some situations
Architecture and risk resolution	*RESL*	- Reduce cost and schedule by avoiding rework	- Additional overhead for risk management is necessary.
Improve process maturity level	*PMAT*	- Reduce cost and schedule by removing the efforts for fixing errors - Improve quality	- Additional overhead for applying CMM is necessary
Improve precedentedness and development flexibility	*PREC, FLEX*	- Reduce cost and schedule by familiarity and flexibility of software development	- Uncontrollable for some situations
Increase budget	Revised win condition	- May enable product to reach competitive critical mass - May increase ROI	- Added funds may not be available

Table 1. The Attributes of Cost-Resolution Option Strategies

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 with 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,
        pros of the strategy,
        cons of the strategy,
        the reduced cost,
        the reduced schedule)
```

Stakeholders can negotiate win-win agreements by suggesting 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 changed 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

[1] The new cost drivers in COCOMO II are underlined in italics.

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.

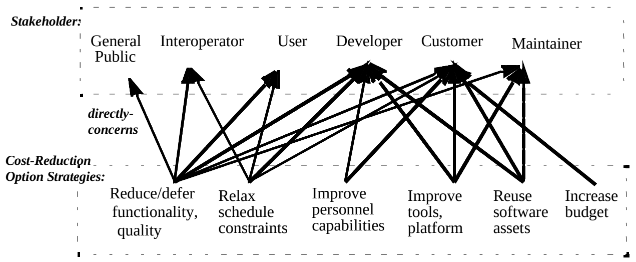


Figure 6. Cost Option / Stakeholder Relationship

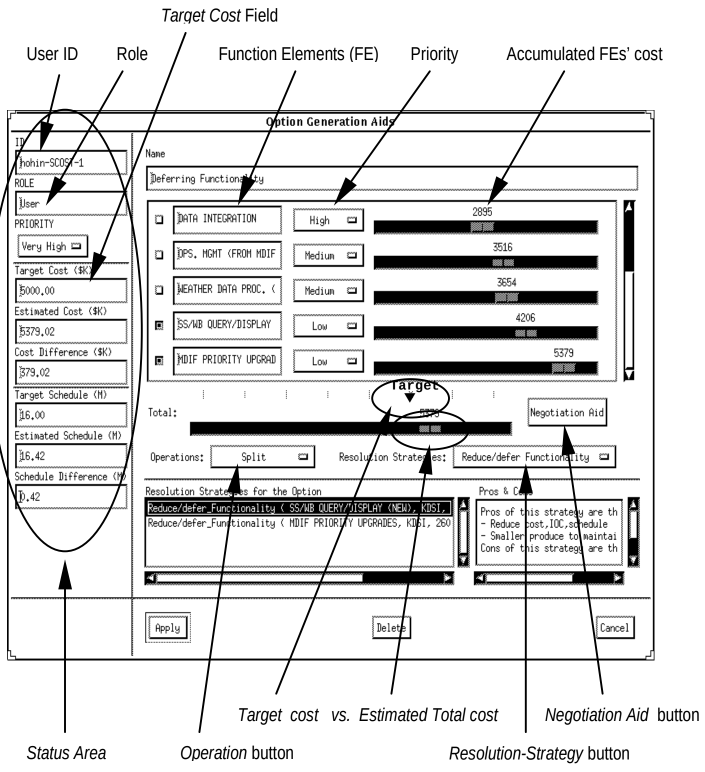
5. S-COST analysis and negotiation aids

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 $6.66 M 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 *Resolution Strategies* button. Or, if justified and feasible, the customer could increase the budget, raising the target cost in the *Target Cost* field (figure 7).



ID: hohin-SCOST-1
ROLE: User
PRIORITY: Very High
Target Cost ($K): 5000.00
Estimated Cost ($K): 5379.02
Cost Difference ($K): 379.02
Target Schedule (M): 16.00
Estimated Schedule (M): 16.42
Schedule Difference (M): 0.42

Name: Deferring Functionality

☐	DATA INTEGRATION	High	2895
☐	OPS. MGMT (FROM MDIF	Medium	3516
☐	WEATHER DATA PROC. (	Medium	3654
▣	SS/WB QUERY/DISPLAY	Low	4206
▣	MDIF PRIORITY UPGRAD	Low	5379

Total: 5379

Operations: Split
Resolution Strategies: Reduce/defer Functionality

Resolution Strategies for the Option
Reduce/defer_Functionality (SS/WB QUERY/DISPLAY (NEW), KDSI,
Reduce/defer_Functionality (MDIF PRIORITY UPGRADES, KDSI, 260

Pros & Cons
Pros of this strategy are th
- Reduce cost,IOC,schedule
- Smaller produce to maintai
Cons of this strategy are th

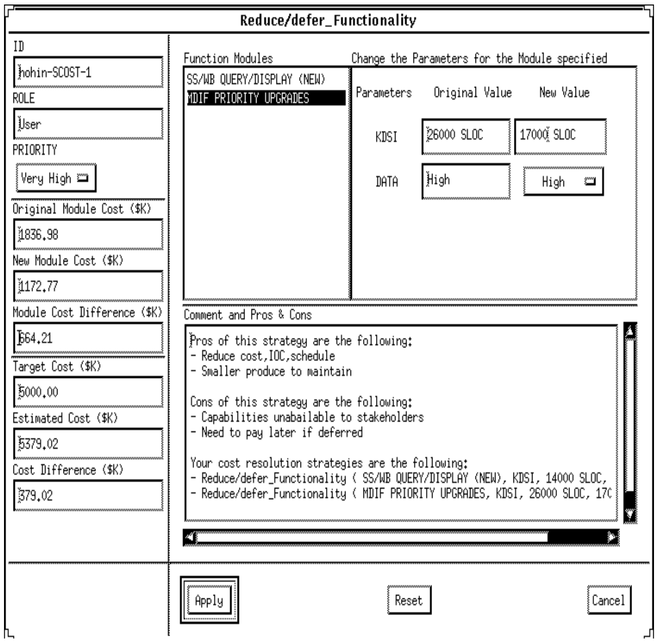
Figure 7. Visualization Window for Option Generation

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, target cost, and target schedule. The *Operations* button is available to enable stakeholders to split, merge, insert, delete, and sort FE(s), as well as select_all, unselect_all, import, and export from/to COCOMO.

The *Resolution Strategies* button brings up a menu containing the cost resolution strategies described in table 1. Each strategy has parameters which can be defined or adjusted once the strategy has been selected. Figure 8 shows an example of applying a "Reduce/defer functionality" Strategy option. Option Strategies are prepared to defer 6 KDSI of Query/Display (New) functionality and 9KDSI of MDIF Priority Upgrades functionality. The Pros & Cons area shows the potential positive and negative aspects of the option. If the stakeholder establishes this as an Option to be addressed in the WinWin system, the Pros & Cons will be included in the Option schema.



ID	hohin-SCOST-1
ROLE	User
PRIORITY	Very High
Original Module Cost ($K)	1836.98
New Module Cost ($K)	1172.77
Module Cost Difference ($K)	664.21
Target Cost ($K)	5000.00
Estimated Cost ($K)	5379.02
Cost Difference ($K)	379.02

Function Modules

- SS/WB QUERY/DISPLAY (NEW)
- MDIF PRIORITY UPGRADES

Change the Parameters for the Module specified

Parameters	Original Value	New Value
KDSI	26000 SLOC	17000 SLOC
DATA	High	High

Comment and Pros & Cons

```
Pros of this strategy are the following:
- Reduce cost,IOC,schedule
- Smaller produce to maintain

Cons of this strategy are the following:
- Capabilities unabailable to stakeholders
- Need to pay later if deferred

Your cost resolution strategies are the following:
- Reduce/defer_Functionality ( SS/WB QUERY/DISPLAY (NEW), KDSI, 14000 SLOC,
- Reduce/defer_Functionality ( MDIF PRIORITY UPGRADES, KDSI, 26000 SLOC, 17C
```

Figure 8. Aids for Applying Cost-Resolution Strategy

5.2 Visualization window features for option negotiation

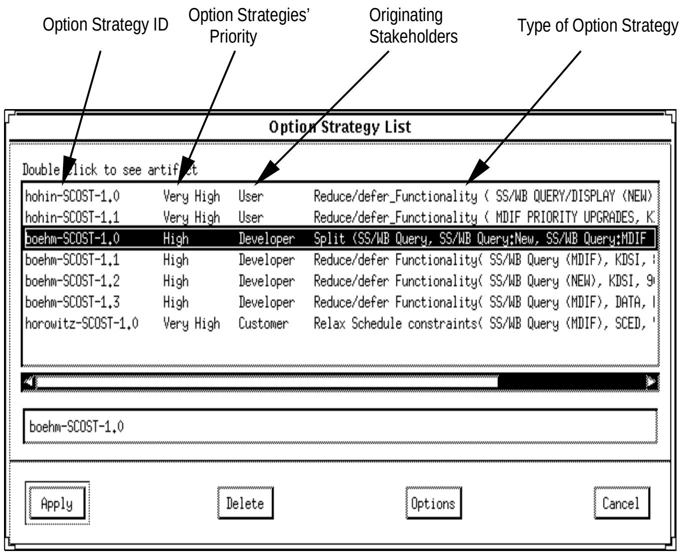
Another visualization aid (obtained via the *Negotiation Aid* button in figure 7) provides the Option Strategy list (figure 9), which summarizes the status of negotiating a combination of Option Strategies suggested by the various stakeholders. The Options can be displayed in order of originating stakeholders, 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.

When Strikeware was used as a multi-stakeholder WinWin negotiation exercise, almost the only option considered by the stakeholders was to defer functionality. It is frequently the case that stakeholders are unaware of attractive options to resolve cost/quality/functionality conflicts. S-COST enables them to explore these additional options.

The primary feedback we received on the original S-COST prototype was to:

- Expand options to include those suggested by COCOMO II and other cost models;
- Provide better interfaces among WinWin, S-COST, and COCOMO II;
- Provide better visualization for the effects of changes;
- Change from small, scrolled, and tiled windows to overlapped, expandable windows.



Double click to see artifact

hohin-SCOST-1.0	Very High	User	Reduce/defer_Functionality (SS/WB QUERY/DISPLAY (NEW)
hohin-SCOST-1.1	Very High	User	Reduce/defer_Functionality (MDIF PRIORITY UPGRADES, K
boehm-SCOST-1.0	High	Developer	Split (SS/WB Query, SS/WB Query:New, SS/WB Query:MDIF
boehm-SCOST-1.1	High	Developer	Reduce/defer Functionality(SS/WB Query (MDIF), KDSI,
boehm-SCOST-1.2	High	Developer	Reduce/defer Functionality(SS/WB Query (NEW), KDSI, 9
boehm-SCOST-1.3	High	Developer	Reduce/defer Functionality(SS/WB Query (MDIF), DATA,
horowitz-SCOST-1.0	Very High	Customer	Relax Schedule constraints(SS/WB Query (MDIF), SCED, '

boehm-SCOST-1.0

Figure 9. The Option Strategy List Window

We have described the COCOMO II extensions above. We have also developed a prototype Software Architecture Attributes Analysis Aid (A4) to interface WinWin, COCOMO II, and architecture-based analysis of cost/schedule/performance/reliability tradeoffs. We are currently reworking the design of the user interface for S-COST II to accommodate the prototype-users' suggestions.

The original S-COST prototype was supported by and available only to USC-CSE Affiliates. The revised version was also supported by DARPA and USAF, and will be generally available under USC-CSE copyright. For further status information, see "http://sunset.usc.edu/~hohin/scost/".

6. Related work

Early characterizations of software quality attribute relationships based on hierarchical models were developed in [Boehm et al., 1976] and [McCall et al., 1977]. The hierarchical models of software quality attributes are based upon a set of quality criteria, each of which has a set of measures or metrics associated with it. 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 focuses on quality analysis methods in software architecture, one of the most important tools for designing and understanding a software system. The manual scenario-based Software Attribute Analysis Method, SAAM [Kazman et al., 1994], provides a technique for analyzing the architecture under consideration with respect to how well or easily it satisfies the constraints imposed by each scenario. The scenarios were used to express the particular instances of each quality attribute important to the customer of a system. One of their difficulties is determining the proper set of the scenarios.

Further CMU-SEI work on software quality attributes [Kazman-Bass, 1994] explored the relationship between software architecture of a system and software qualities to be achieved by the system. The relationship is based on design operations (called “unit operations”) such as separation, abstraction, compression, composition, resource sharing, and replication. Their work provides a useful first-order conflict analysis of the interaction among quality attributes, though their method of deriving architectures from requirements is somewhat oversimplified. Our research focus in QARCC [Boehm-In, 1996a] used this concept and extended it with semi-formalized structure to provide a useful description language for architecture tradeoff analysis.

Univ. of Toronto [Chung et al., 1995] developed an automated assistance in dealing with interactions among non-functional quality requirements based on Non-Functional Requirements (NFRs). They focused on traceability of quality requirements with more emphasis on incorporating changes in NFRs (e.g., systematically detecting defects and supporting the process of corresponding changes in design and implementation). However, there is no consideration of stakeholders who have different quality-attribute priorities and concerns.

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 of 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

This research is sponsored by the Advanced Research Projects Agency (ARPA) through Rome Laboratory under contract F30602-94-C-0195 and by the Affiliates of the USC Center for Software Engineering: Allied Signal Corp.,, Bellcore, Boeing, Electronic Data Systems Corporation, E-Systems, FAA, GDE systems, Hughes Aircraft Company, Interactive Development Environments, Institute for Defense Analysis, Jet Propulsion Laboratory, Litton Data Systems, Lockheed Martin Corporation, MCC, Motorola Inc., Northrop Grumman Corporation, Rational Software Corporation, Raytheon, Science Applications International Corporation, Software Engineering Institute (CMU), Software Productivity Consortium, Sun Microsystems, Inc., Texas Instruments, TRW, U.S. Air Force Rome Laboratory, U.S. Army Research Laboratory, and Xerox Corporation.

References

[Boehm et al., 1976] Boehm, B., Brown, J., and Lipow, M., “Quantitative Evaluation of Software Quality”, Proceedings, ICSE2, October 1976, pp. 592-605.

[Boehm, 1981] Boehm, B., *Software Engineering Economics*, Prentice-Hall, 1981.

[Boehm-Ross, 1989] Boehm, B. and Ross, R., “Theory W Software Project Management: Principles and Examples,”, IEEE TSE, July 1986, pp. 902-916.

[Boehm et al., 1994] Boehm, B., Bose, P., Horowitz, E., and Lee, M., “Software Requirements As Negotiated Win Conditions”, *Proceedings of the First International Conference on Requirements Engineering (ICRE94)*, IEEE Computer Society Press, Colorado Springs Colorado, April 1994, pp. 74-83.

[Boehm et al., 1995a] Boehm, B., Bose, P., Horowitz, E., and Lee, M., “Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach”, *Proceedings of the 17th International Conference on Software Engineering (ICSE-17),* IEEE Computer Society Press, Seattle, April 1995.

[Boehm et al., 1995b] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R., “Cost Models for Future Life-Cycle Processes: COCOMO 2.0”, Annals of Software Engineering, 1995, pp. 57-94.

[Boehm-In, 1996a] Boehm, B. and In, H., “Identifying Quality-Requirement Conflicts”, *IEEE Software*, IEEE Computer Society Press, March 1996, pp. 25-35.

[Boehm-In, 1996b] Boehm, B. and In, H., “Software Cost Option Strategy Tool (S-COST)”, *COMPSAC96 (The Twentieths*

Annual International Computer Software and Applications Conference), IEEE Comp. Society Press, Seoul, Korea, August 1996, pp. 15-20.

[Boehm et al., 1998] Boehm, B., Port, D., Kwan, J., Egyed, A., Shah, A., and Madachy, R., “Using the WinWin Spiral Model: A Case Study”, *Computer*, July 1998.

[Bowen et al., 1985] Bowen, T., Wigle, G., and Tsai, J., “Specifications of Quality Attributes”, Technical Report RADC-TR-85-37, Vol. I-III, Rome Air Development Center, Griffiss Air Force Base, NY, 1985.

[Chung et al., 1995] Chung, L., Nixon, B., and Yu, E., “Using Non-Functional Requirements to Systematically Support Change”, Proceedings of the second *ICRE (Int. Conf. of Requirements Engineering*), IEEE Computer Press, March 1995, pp. 132-139.

[Eureka-Ryan 1988] Eureka, W.E. and Ryan, N.E., *The customer-Driven Company: Managerial Perspectives on QFD*, Dearborn, Mich.: ASI Press, 1988.

[Deming 1989] Deming, E., Out of Crisis, MIT Center for Advanced Engineering Study, Cambridge, MA, 1989.

[DoD, 1974] U.S. Department of Defense, “Software Quality Assurance Program Requirements”, MIL-S-52779 (AD), 5, April 1974.

[IEEE, 1990] “IEEE Standard Glossary of Software Engineering Terminology”, IEEE Std 610.12-1990, September 28, 1990.

[Kazman-Bass, 1994] Kazman, R. and Bass, L., “Toward Deriving Software Architectures From Quality Attributes”, CMU/SEI-94-TR-10, Pittsburgh, SEI/CMU, August 1994.

[Kazman et al., 1994] Kazman, R., Bass, L., Abowd, G., and Webb, M., “SAAM: A Method for Analyzing the Properties of Software Architectures”, Proc. of the 16th ICSE-16 (Int. Conf. of Soft. Eng.), IEEE Computer Society Press, Sorrento, Italy, May 1994, pp. 81-90.

[Lasky-Donaghy 1993] Lasky, J. and Donaghy K., “Conflict Resolution (CORE) for Software Quality Factors,”, Technical Report RL-TR-93-80, Rome Lab., Griffis Air Force Base, NY, 1993.

[McCall et al., 1977] McCall, J.A., Richards, P.K., and Walters, G.F., “Factors in Software Quality”, Technical Report RADC-TR-92-79, Rome Laboratory, Griffiss Air Force Base, NY, 1977.

[Murine 1995] Murine, G. and Murine, B., “Implementing a software quality metric program based on the Rome Laboratory initiatives”, Annals of Software Engineering , vol. 1, Baltzer Science Publishers, 1995, pp.155-177.

[Pardee 1996] Pardee, W., *To Satisfy & Delight Your Customer: How to Manage for Customer Value*, Dorset House Publishing, NY, 1996.