From Requirements Negotiation to Software Architectural Decisions

Hoh In
Dept. of Computer Science
Texas A&M University
College Station, TX 77843
hohin@cs.tamu.edu

Rick Kazman
Software Engineering Institution
Carnegie Mellon University
Pittsburgh, PA, USA 15213
Kazman@sei.cmu.edu

David Olson
Dept. of Info. Operations and Management
Texas A&M University
College Station, TX 77843
dolson@tamu.edu

Abstract

Uncertainty of system properties (e.g., performance, reliability, security, interoperability, usability, etc.) often hinders the progress of requirements negotiation. Software architecture evaluation techniques enable stakeholders to clarify the uncertainty of system properties. In another hand, software architecture alternatives cannot be evaluated in a thorough way without consideration of different stakeholders' negotiated requirements. Effective requirements negotiation is therefore needed to evaluate architecture alternatives.

This paper proposes an integrated decision-making framework from software requirements negotiation to architecture evaluation based on WinWin and CBAM (Cost Benefit Analysis Method). The integrated framework helps stakeholders elicit, explore, evaluate, negotiate, and agree upon software architecture alternatives based negotiated requirements.

Keywords: ABASs, ATAM, CBAM, conflict resolution, requirements negotiation, WinWin.

1. Motivation

Many software projects have failed because their requirements were poorly negotiated among stakeholders [4]. Several keynote speakers in the International Conference on Software Engineering (ICSE) emphasized the importance of requirements negotiation as follows:

- "How the requirements were negotiated is far more important than how the requirements were specified" (Tom De Marco, ICSE 96)
- "Negotiation is the best way to avoid "Death March" projects" (Ed Yourdon, ICSE 97)
- "Problems with reaching agreement were more critical to my projects' success than such factors as tools, process maturity, and design methods" (Mark Weiser, ICSE 97)

The WinWin negotiation model, developed by the USC Center for Software Engineering, provides a general framework for successful requirements negotiation. In WinWin, stakeholders elicit their win conditions, identify issues/conflicts, generate options to resolve the issues, negotiate the options and reach agreement [1,2,3]. However, it is not clear which architecture alternatives should be considered as the options and/or how they should be explored, evaluated, and negotiated in order to reach agreement among stakeholders.

As an architecture evaluation technique, the CMU Software Engineering Institute has developed the Cost Benefit Analysis Method (CBAM) that explores, analyzes, and makes decisions regarding software architecture alternatives (called "architecture strategies") [14]. Still, it is not clear how the explored architecture strategies satisfy the initial requirements (goals/constraints) of stakeholders who have different roles, responsibilities, and priorities. A general negotiation framework to aid in progressing from requirements to architectural decisions is needed.

In this paper, we propose an integrated decision-making framework, based on WinWin and CBAM, that aids in systematically determining architecture alternatives from negotiated requirements among stakeholders. WinWin provides a general negotiation framework to elicit requirements, explore architecture alternatives, and reach agreement. CBAM helps stakeholders negotiate architecture alternatives in a systematic way.

This paper is organized as following: Section 2 describes the context of the work. Section 3 presents and describes the proposed framework. In Section 4 and 5, future research challenges and conclusions are presented.

2. Context For the Work

2.1 WinWin Negotiation Model

The WinWin model provides a general framework for identifying and resolving requirements conflicts by eliciting and negotiating artifacts such as win conditions, issues, options, and agreements. The WinWin model uses Theory W [5], "Make everyone a winner", to generate the stakeholder win-win situation incrementally through the Spiral Model. WinWin assists stakeholders to identify and negotiate issues (i.e., conflicts among their win conditions), since the goal of Theory W involves stakeholders identifying their win conditions, and reconciling conflicts among win conditions.

The dotted-lined box (steps 1,2,3, and 8) shown in Figure 1 presents the WinWin Negotiation Model. Stakeholders begin by entering their win conditions (step 1). If a conflict among stakeholders' win conditions is identified, an issue schema is composed, summarizing the conflict and the win conditions it involves (step 2). For each issue, stakeholders prepare candidate option schemas addressing the issue (step 3). Stakeholders then evaluate the options, delay decision on some, agree to reject others, and ultimately converge on a mutually satisfactory 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 (step 8). Experience also indicates that WinWin is not a panacea for all conflict situations, but generally increases stakeholders' levels of cooperation and trust [4, 11].

Agreement is not always guaranteed. There are often tradeoffs among win conditions that need to be balanced. CBAM provides a means to balance these tradeoffs, and a framework for discussion that can lead to resolution.

2.2 CBAM (Cost-Benefit Analysis Method)

The ATAM [13] uncovers the architectural decisions made in a software project and links them to business goals and QA (quality attribute) response measures. The CBAM [14] builds on this foundation by additionally determining the costs, benefits, and uncertainties associated with these decisions.

Given this information, the stakeholders can then decide how to address their important QA response measures. For example, if they felt that the system's reliability was not sufficiently high they could use the ATAM/CBAM methods to decide whether to use redundant hardware, checkpointing, or some other architectural decision addressed at increasing the system's reliability. Or the stakeholders can choose to invest their finite resources in some other QA—perhaps believing that higher performance will have a better benefit/cost ratio. A system always has a limited budget for creation or upgrade and so every architectural choice is, in some sense, competing with every other one for inclusion.

The CBAM is a framework and it does not make decisions for the stakeholders; it simply aids them in the elicitation and documentation of costs, benefits, and uncertainty and gives them a rational decision-making process.

When an ATAM is completed, we expect to have a set of artifacts documented as follows:

- a description of the *business goals* that are crucial to the success of the system
- a set of *architectural views* that document that existing or proposed architecture
- a *utility tree* which represents a decomposition of the stakeholders' goals, for the architecture. The utility tree starts with high-level statements of QAs and decomposes these into specific instances of performance, availability, etc. requirements and realizes these as scenarios
- a set of *risks* that have been identified
- a set of *sensitivity points* (architectural decisions that affect some QA measure of concern)
- a set of *tradeoff points* (architectural decisions that affect more than one QA measure, some positively and some negatively)

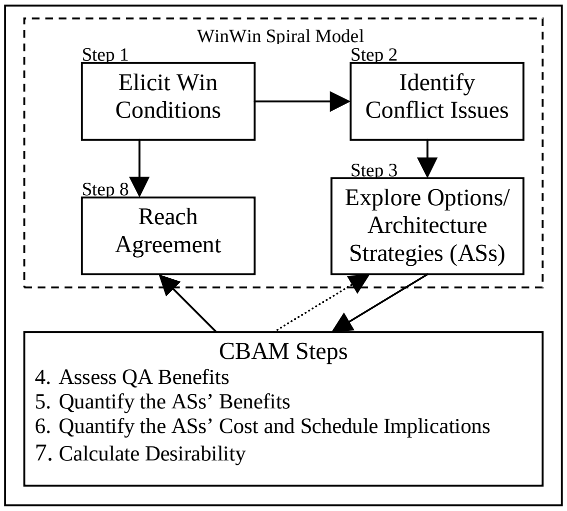
The CBAM builds upon this foundation of information by probing the architectural strategies (ASs) that are proposed in response to the scenarios, risks, sensitivity points, and tradeoffs. The steps of the CBAM are as follows. Each of these steps can be executed in the first (triage) and second (detailed examination) phases:

1. Choose Scenarios and Architectural Strategies
2. Assess QA Benefits
3. Quantify the Architectural Strategies' Benefits
4. Quantify the Architectural Strategies' Costs and Schedule Implications
5. Calculate Desirability
6. Make Decisions

3. The Steps of the Integrated Framework

The integrated framework shown Figure 1 begins with the WinWin process, which elicits what stakeholders need, identifies conflicts in these needs among the stakeholders, and explores the conflict-resolution options. CBAM is proposed here as a means to supplement the WinWin process of systematically evaluating and negotiating software architecture alternatives (as conflict-resolution options) by eliciting stakeholders' benefits and costs. The process may lead to agreement by itself (although this is not guaranteed). Reviewing each stakeholder' win conditions at this final stage may further aid the next cycle of reconciliation or compromise in the WinWin Spiral Process model.

Figure 1: *The Integrated Framework*


WinWin Spiral Model
Step 1
Elicit Win Conditions
Step 2
Identify Conflict Issues
Step 8
Reach Agreement
Step 3
Explore Options/ Architecture Strategies (ASs)
CBAM Steps
4. Assess QA Benefits
5. Quantify the ASs' Benefits
6. Quantify the ASs' Cost and Schedule Implications
7. Calculate Desirability

The steps shown in Figure 1 are elaborated in the following subsections.

Step 1: Elicit Win Conditions

Each stakeholder identifies their win conditions. This step provides the basis for identification of ideal project features by stakeholders.

Step 2: Identify Quality Attribute Conflicts/Issues

The lists of win conditions are then reviewed to identify quality attribute conflicts. The identified conflicts are then categorized as being either a direct conflict or a potential conflict. This step may be accomplished manually, but future work may be able to incorporate software agents.

Step 3: Explore Architecture Strategies as Conflict-Resolution Options

Based upon the conflicts/issues generated in step 2, the stakeholders can now generate conflict-resolution options. It is best to generate a list of options which may emphasize those characteristics preferred by each stakeholder, but that include some balance representing needed conditions of all stakeholders. These options are called Architectural Strategies (ASs) in the CBAM. Where do such ASs come from? They can come from any number of areas: from the architects' experience, by borrowing from systems that have experienced similar problems in the past, or from repositories of design solutions, such as Design Patterns [10] or ABASs [15].

Step 4. Assess Quality Attribute (QA) Benefits

To aid in decision making, the stakeholders now need to determine both the costs and benefits that accrue to the various ASs. Determining costs is a well-established component of software engineering. This is not addressed directly by the CBAM—we assume that some methods of doing this exists in the organization. Determining benefits is less well-established and this is the province of the CBAM. As a means of determining the benefit of an individual AS, a benefit evaluation function is created. Benefit must be correlated with the degree to which an Architectural Strategy supports QA goals, which in turn relates back to the business goals for the system. These goals are both outputs of the ATAM.

To do this we have each of the stakeholders assign a *Quality Attribute Score* (*QAScore*) to each QA system goal. We let the customer determine which stakeholders should be in a decision-making capacity. The stakeholders are instructed to choose these scores such that they total 100. For example:

Performance:	15
Security:	15
Modifiability:	30
Reliability:	25
Interoperability:	15

We also ask each stakeholder to describe the particular aspect of the quality attribute that caused them to make this score. For example, modifiability has a score of 30 and above, but it is *GUI modifiability* that is the primary determinant of this score.

Step 5: Quantify the Architecture Strategies' Benefits

We then use these scores to evaluate each of the ASs. Very rarely does an AS only affect a single QA. ASs will have effects on multiple QAs, some positive and some negative, and to varying degrees. To capture this, we ask the stakeholders to rank each AS in terms of its *contribution* (*Cont*) to each QA on a scale of –1 to +1. A +1 means that this AS has a substantial positive effect on the QA (for example, an AS under consideration might have a substantial positive effect on performance) and a -1 means the opposite. Based upon this information each AS_i can now be assigned a computed benefit score from –100 to +100 "Benys" using the following formula:

$$\textbf{\textit{Benefit}}(AS_i) = \textbf{\textit{Sum}}\ (Cont_{ij} * QAScore_j)$$

For example, given the QAscores listed above, we can calculate benefit scores for two hypothetical ASs as follows (note that we only consider the QAs for which there is a non-zero contribution):

AS5: Performance (-0.2), Modifiability (0.6), Interoperability (0.3)

Benefit(AS5) = -0.2 * 15 + 0.6 * 30 + 0.3 * 15
= 20.5

AS6: Performance (0.8), Reliability(-0.2), Security(-0.4)

Benefit (AS6) = 0.8 * 15 + -0.2 * 25 + -0.4 * 15
= 1

This score allows us to rank the benefit of every architectural change that has been contemplated. But clearly this evaluation is fraught with uncertainty. We can capture this uncertainty by recording the variations in stakeholder judgements.

In the CBAM we use Kendall's concordance coefficient for the group as a whole as a measure of the uncertainty of the group, as described in [14]. The more highly correlated the group, the higher the concordance coefficient and hence the lower the uncertainty.

Step 6: Quantify the Architecture Strategies' Cost and Schedule Implications

Now that the benefits have been estimated by the stakeholders, we must capture two other crucial pieces of information about the various ASs: their costs and their schedule implications. We propose no special cost estimation technique here (although we do think that cost estimation methods that take architecture into account are a desirable and inevitable improvement to existing methods). We assume that an organization has some method (even if it is *ad hoc*) of estimating the costs of implementing new services and features. We simply need to capture these estimates, as they are associated with each AS.

In addition we need to capture any schedule implications of the ASs. For example, do several ASs require the use of the same critical resource (personnel, hardware, software)? If so then attempting to implement them simultaneously might be impossible even if their cost/benefit numbers indicate that this is the strategy that brings the organization the greatest profit. Similarly, we want to note cases where AS_i depends upon AS_j and so implementing AS_j first actually reduces the cost of implementing $AS_{i.}$

Step 7: Calculate Desirability

Given this information we are in a position to calculate a "Desirability" metric, as follows:

$$\textbf{\textit{Desirability}}(AS_i) = \textbf{\textit{Benefit}}(AS_i) / \textbf{\textit{Cost}}\ (AS_i)$$

This metric indicates ASs that will bring high benefit to the organization at relatively low cost. In addition to calculating this metric, the absolute benefit and cost numbers need to be considered as does the uncertainty surrounding all of these numbers, as discussed in [14].

Step 8: Reach Agreement

At this point the negotiating among the stakeholders can begin in earnest. This negotiation will be informed, rather than simply a matter of opinion. The costs, benefits, and uncertainty of each of the ASs will be plain for each stakeholder to see, as well as the schedule implications and dependencies (if any). These ASs can be tied back to the business goals, and hence the win conditions of each of the stakeholders.

What results is much less an argument than a discussion about priorities, risk averseness, and the assumptions underlying the model. Stakeholders may still disagree about what direction to take the architecture and the system, but they will do so based upon a large base of facts and accumulated evidence and such disagreements can be more easily moderated than those which are simply based upon opinion and prejudice.

4. Further Research Challenges

The integrated decision-making framework offers useful tools to aid the stakeholder negotiation process from requirements to architecture evaluation. However, there are a number of challenges involved in its process. These challenges are given briefly here due to lack of space, but provide a great deal of fruitful scope for future research.

Exploration of Architecture Strategies: An important issue is how to sort these issues/conflicts (shown in Step 2) in order to explore Architecture Strategies as conflict-resolution options (shown in Step 3) reflecting the win conditions of the different stakeholders. That is, should several Architecture Strategies be explored for each issue/conflict, or per a set of issues/conflicts, or for all issues? Our feasible solution approach could be to classify (cluster) conflicting issues and identify a small set of Architecture Strategies for each set of the clustered issues/conflicts. One possible solution would use a cross-impact (or dependency) analysis technique, which would identify clusters of stakeholder positions. An ideal solution for each cluster could be used as the basis of an Architecture Strategy. This would imply the need to consider the entire set of criteria.

Determining Detailed Benefit-Cost Criteria: Agreeing on detailed criteria of benefits and costs among stakeholders is a challenging problem [7, 8, 9, 12, 15, 20]. An initial, complete list of criteria can be generated first. This long list of micro-criteria can then be reviewed and grouped by theme, yielding the macro-criteria. The identification of the macro-criteria is usually a natural grouping of micro-criteria representing a common theme.

Sensitivity to Uncertainty in Benefit, Cost Values: Objectively quantifying the benefits and costs of Architecture Strategies (shown in Step 5 and 6) is another challenging problem. Even though we can quantify them with popular cost and/or benefit models, the accuracy of the estimates is limited. Thus, the inherent uncertainty in the models must be taken into account [14].

Reaching Agreement from Desirability Results: Reaching agreement is a difficult task. One way to accomplish agreement is to let an arbitrator (e.g., the responsible manager) make the decision. Use of a group support system lets all stakeholders have the opportunity to provide their inputs. That alleviates some of the apparent arbitrariness usually perceived in dictatorial decisions. At the opposite extreme, the decision could be made by voting. As in political decision-making, this does not guarantee complete acceptance. Quite the contrary, all of the mechanisms that have been applied to reaching a decision can be applied in the proposed system. Hopefully, the opportunity to express win conditions, the use of group support systems, and objective cost-benefit evaluation modeling will create a decision-making environment that gains broader support from participating stakeholders.

5. Conclusions

In this paper, we have introduced an integrated framework for coordinating architectural decisions with requirements negotiation framework. This integrated framework has the following synergy compared to the requirements negotiation models [16, 17, 18, 19, 21] or to software architecture evaluation work (e.g., [13, 14, 15]):

- Enabling a more powerful multi-viewpoint analysis of architecture evaluation. Architecture alternatives (e.g., "Architecture Strategies") can be evaluated based on requirements elicited, explored, and negotiated from and by multiple stakeholders who have different roles, responsibilities, and priorities.
- Facilitating requirements negotiation in a more systematic way. Uncertainty in requirements negotiation can be clarified through the exploration, evaluation, and negotiation of architecture alternatives.

As a logical step, we will examine a real-world case study to obtain more insight of better way to overcome the future research challenges presented in Section 4.

In conclusion, we expect that the integrated framework provides a systematic, yet practical method for stakeholders to negotiate from requirements to architecture alternatives.

ACKNOWLEDGEMENTS

This work is partially supported by funding from NASA JPL under the contract C00-00443 with Texas A&M University. We would like to thank Drs. Barry Boehm, Robert Briggs, and Tom Rodgers for the helpful discussion on this topic.

REFERENCES

[1] Boehm, B., Bose, P., Horowitz, E., and Lee, M., “Software Requirements as Negotiated Win Conditions”, in *First International Conference on Requirements Engineering* (ICRE94). Colorado Springs: IEEE Computer Society Press, 1994.

[2] Boehm, B., Bose, P., Horowtiz, E., and Lee, M., "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach”, in *17th International Conference on Software Engineering* (ICSE-17). Seattle: IEEE Computer Society Press, 1995.

[3] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J., and Madachy, R., "A Stakeholder Win-Win Approach to Software Engineering Education”, *Annals of Software Engineering*, 1999.

[4] Boehm, B. and In, H., "Identifying Quality-Requirement Conflicts", IEEE Software, Vol. 13, No. 2, pp. 25-35, March 1996.

[5] Boehm, B. and Ross R., "Theory W Software Project Management: Principles and Examples", *IEEE Transactions on Software Engineering*, 1989. July: p. 902-916.

[6] Gruenbacher, Paul & Briggs, R.O. "Surfacing Tacit Knowledge in Requirements Negotiation: Experiences Using EasyWinWin", *Proceedings of the 34th Thirty Fourth Hawaii International Conference on System Sciences*, CLUS09, 2001

[7] Choo, E.U., Schoner, B., and Wedley, W.C., “Interpretation of Criteria Weights in Multicriteria Decision Making”, *Computers and Industrial Engineering* 37, 1999, pp. 527-541.

[8] Dyer, J.S. and Sarin, R.K., “Measurable Multiattribute Value Functions”, *Operations Research* 27, 1979, pp. 810-822.

[9] Edwards, W. and Barron, F.H., “SMARTS and SMARTER: Improved Simple Methods for Multiattribute Utility Measurement”, *Organizational Behavior and Human Decision Processes* 60, 1994, pp. 306-325.

[10] Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns, Addison Wesley, 1995

[11] In, H., Boehm, B., Rodgers, T., and Deutsch, M., "Applying WinWin to Quality Requirements: A Case Study", IEEE International Conference on Software Engineering (ICSE 2001), IEEE Computer Society Press, Toronto, Canada, May 12-19, (to appear)

[12] Keeney, R.L. and Raiffa, H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, 1976.

[13] Kazman, R., Klein, M., Clements, P., “ATAM: A Method for Architecture Evaluation”, CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 2000.

[14] Kazman, R., Asundi, J., Klein, M., “Quantifying the Costs and Benefits of Architectural Decisions”, *Proceedings of the 23rd International Conference on Software Engineering (ICSE 23)*, (Toronto, Canada), May 2001, to appear.

[15] M. Klein, R. Kazman, L. Bass, S. J. Carriere, M. Barbacci, H. Lipson, “Attribute-Based Architectural Styles”, *Software Architecture* (*Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1))*, (San Antonio, TX), February 1999, 225-243.

[16] Klein, M., “Supporting Conflict Resolution in Cooperative Design Systems”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 6, pp1379-1390, Nov/Dec 1991.

[17] Lee, J., “SIBYL: A Qualitative Decision Management System”, In Artificial Intelligence at MIT: Expanding Frontiers, Edited by P. Winston and S. Shellard, MIT Press, Cambridge, 1990.

[18] Mylopoulos, J., Chung, L., Wang, H.Q., Liao, S., Yu, E. “Extending Object-Oriented Analysis to Explore Alternatives”, IEEE Software. February 2001.

[19] Nuseibeh, B. A., Easterbrook, S. M., and Russo, A., "Leveraging Inconsistency in Software Development", IEEE Computer, Volume 33, No. 4, Pages 24-29, April 2000.

[20] Olson, D.L., *Decision Aids for Selection Problems.* Springer, New York, 1996.

[21] Ramesh, B. and Sengupta, K., “Managing Cognitive and Mixed-motive Conflicts in Concurrent Engineering”, Concurrent Engineering, Vol. 18, No. 6, 1992, pp 498-519.