Simulations for Very Early Lifecycle Quality Evaluations

Eliza Chiang
Dept. of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall, Vancouver, BC, Canada
Email: echiang@interchange.ubc.ca
http://www.ece.ubc.ca/˜ elizac/vio/

Tim Menzies
Lane Department of Computer Science
West Virginia University, PO Box 6109
Morgantown, WV, 26506-6109, USA
Email: tim@menzies.com
http://tim.menzies.com

Abstract

Chung et al. have proposed a graphical model that captures the inter-dependencies between design alternatives in terms of synergy and tradeoffs. This model can assist in identifying quality/risk trade-offs early in the life cycle of software development, such as architectural design and testing process choices. The Chung et.al. method is an analysis framework only: their technique does not include an execution or analysis module. This paper presents a simulation tool developed to analyze such a model, and techniques to facilitate decision making by reducing the space of options worth considering. Our techniques combine Monte Carlo simulations to generate options with a machine learner to determine which option yields the most/least favorable outcome. Experiments based on the above methodology were performed on two case studies, and the results showed that treatment learning successfully pinpointed the key attributes among uncertainties in our test domains.

1. Introduction

Software system must meet all the functional requirements in order to provide desired functionalities to users. In addition, it must exhibit extra non-functional software quality attributes such as accuracy, security, performance, and other business goals. As there are no clear-cut criteria to determine whether these goals are satisfied, *Chung, Nixon, Yu and Mylopoulous* [1] used the notion of softgoals to represent such goals. *Chung et. al.* also define an entire softgoal modeling framework, featuring tradeoffs and inter-dependencies between system quality attributes and design alternatives. But their framework is a paper design only: if an analyst wants to simulate a softgoal system, they face the problem of simulations across a space of uncertainties intrinsic to softgoals. For example, an analyst can connect two softgoals with qualitative terms such as *helps* and say *softgoal1 helps softgoal2*. Qualitative influences such as *helps* are subjective beliefs, and are thus prone to be inconsistent. Our goal, therefore, is to develop a simulation tool that finds stable conclusions across inconsistencies within softgoal frameworks.

Aside from inconsistent beliefs, another problem with drawing conclusions from a softgoal framework is the lack of supportive data. In the current software engineering practice, there is not much data available to perform statistical analysis [2]. This is especially true during the early lifecycle of software development, when decisions are made based on uncertain and subjective knowledge. And in the case of advanced technologies and systems, there is little past experience to learn from. Without supportive data, the relevance of any conclusion drawn from a softgoal framework is questionable. In spite of this, estimations on the potential risks and benefits of design decisions during the earlier requirement phase is essential, because these early decisions have the most leverage to influence the development to follow. The *Softgoal Simulation Tool* presented in this paper, therefore, is designed to aid decision making in times such as early software development lifecycle, a time when domain knowledge is incomplete and inconsistent.

The premise of our methodology is that within a large space of uncertainties generated from a model, there often exist emergent stable properties [3]. If isolated, these properties can be used to drive a system towards the more/less preferred direction. In order to find such consistent behaviors, we apply "bounded" randomness to handle imprecise knowledge, and utilize *Monte Carlo* simulation [4] to explore a wide range of system behaviors. TAR2 treatment learner [5] is employed to automatically summarize these behaviors, and return recommendations that can drive the system to some preferred mode.

As for the implementation of the Softgoal Simulation Tool, it is designed to be light-weight and highly customizable to different business goals. In our approach: *(i)* a model is defined that is a set of logical constraints between variables; *(ii)* a solution is generated from that model that satisfies those constraints. In the ideal case, all model constraints can be satisfied. However, in the case of models generated from competing stakeholders, this may not be possible. Hence, our approach offers a range of operators which tries to satisfying all, many, or one of a set of constraints. The appropriate selection of operators depends on the business at hand. Examples

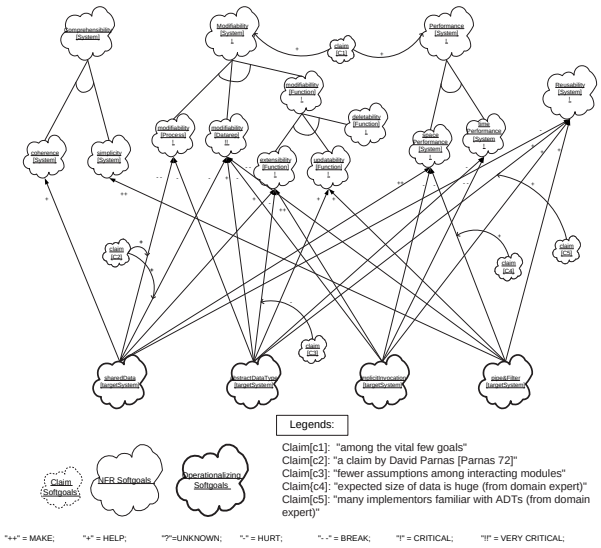

Fig. 1. KWIC framework

and discussions on using these operators are presented in sections 3 and 4.

2. Softgoal Modeling and Simulation

This section outlines our modeling and simulation approach, which involves the softgoal framework, *Monte Carlo* simulation, and treatment learning.

2.1 Overview of the Softgoal Framework

Softgoal framework consists of three types of softgoals: the *Non-Functional-Requirement(NFR) softgoals*, the *operationalizing softgoals*, and the *claim softgoals*. *NFR softgoals* represent quality requirements such as "time-performance". *Operationalizing softgoals* comprise of possible solutions or design alternatives to achieve the *NFR softgoals* (e.g. "incorporate javascript in online storefront"). *Claim softgoals* argue the rationale and explain the context for a softgoal or interdependency link (e.g. a claim may argue that "client-side scripting loads faster"). As there are no clear cut criteria for success, *NFR softgoals* may not be absolutely achieved, yet they can be sufficiently satisficed [6], meaning that they are achieved with acceptable limits.

To explain the notion of softgoal modeling, we used the *Keyword in Context (KWIC) System* to illustrate how the softgoal framework captures design alternatives and quality attributes. The *KWIC* framework, as shown in figure 1, defines the tradeoffs among *NFR*s and the architectural design alternatives within the *KWIC* domain. The top-level *NFR softgoals - Comprehensibility, Modifiability, Performance, Reusability* - are the quality requirements to be satisficed. The design alternatives - *Shared Data, Abstract Data Type, Implicit Invocation, Pipes and Filters* - populate the bottom-level as *operationalizing softgoals*. The sub-softgoals at the mid-level of the framework are obtained by decomposing the top-level *NFR softgoals*. The inter-dependencies between softgoals are specified in the framework with symbols such as "$--$" and "$+$". In addition, softgoals can have associated priorities, denoted by "!", "!!" in the figure.

The *KWIC* framework described above demonstrates how similar technique can be applied to other systems to capture the tradeoffs/synergy between quality attributes and design alternatives. After a framework is constructed, it is encoded into text format and fed to the *Softgoal Simulation Tool* for automatic inference.

2.2 Softgoal Simulation Tool

1) Inference and Cost/Benefit Calculation: Each search performed on the softgoal framework generates a consistent "world" - a scenario where the top-level softgoal is satisficed when a set of softgoals are satisficed/denied. This "world" can be different for each search, depending on the combination logics between softgoals and randomness embedded in the framework definition. Different business concerns can be addressed by using different combinations of logical operators in the analysis of softgoal frameworks. The logic operators supported by the tool are AND, OR, and ANY[1].

After a "world" is generated, its desirability is rated by computing the "benefit" and "cost" based on user-configured parameters. These ratings, once obtained by performing *Monte Carlo* process, are used by TAR2 treatment learner for classification.

2) Treatment Learning: TAR2 treatment learner classifies each "world" according to its "cost" and "benefit" scores. Different preference scheme can be configured by modifying a user-defined ranking scheme that maps ranges of costs & benefits to appropriate classes[2]. With this ranking scheme, TAR2 searches the datasets for the candidate attribute ranges, i.e., ranges that are more common in the highly ranked classes than the other classes. Knowing this range of attributes can greatly assist in making design decisions, as the space of considerations is narrowed down to only the attributes that would assert significant impacts to the system.

3. Experiments and Results on the *KWIC* system

This section presents the study on the *KWIC* framework as detailed in section 2.1. The objective of this study is to look for design alternatives that would significantly

[1]Logic ANY is similar to OR in its satisficing criteria, except that the inference engine would try to prove more than one of the chained softgoals .

[2]For example, the *high-benefit/low-cost* class has a higher rank than the *low-benefit/high-cost* class.

impact the *KWIC* system among inconsistent knowledge. This study is performed on two logical interpretations, representing the rigorous and weak quality assurance scheme. Details on these experiments are given below.

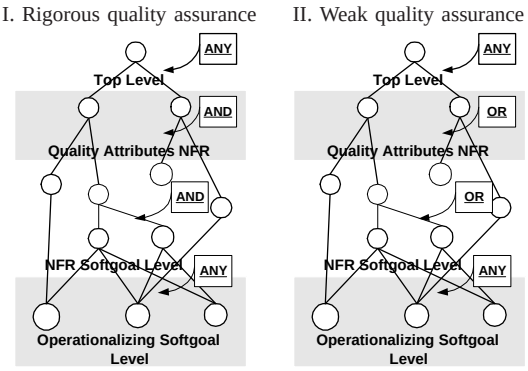

Fig. 2. Logic configurations for the KWIC Framework

3.1 Experiment 1: Rigorous Quality Assurance Modeling

TABLE 1. Rigorous quality assurance Percentage distributions of $benefits$ and $costs$ seen in 10,000 runs of fig 2.I

Cost	Benefit <5.5	<11	<17	≥17	total
0					
1	3.9				3.9
2	33				33
3	10	3.4			13
4	6.7	23	5.9		36
5	1.1	6.7	4.9	1.0	14
total	55	33	11	1.4	100

a. No treatment

Cost	Benefit <5.5	<11	<17	≥17	total
0					
1					
2					
3					
4	10	25	3.5		39
5	4.7	31	22	3.6	61
total	15	56	25	3.7	100

b. Treatments applied (*Shared Data=y; Abstract Data Type=y; Implicit Invocation=y; c3=y*)

This experiment is intended for system designs which strict quality assurance is mandatory. The goal is to find out what design alternatives would optimize system quality attributes. The logic composition, as shown in figure 2.I, enforced strict constraints (with the use of ANDs) to satisfice the overall system quality. The ranking scheme takes account on both benefit and cost, with slight preference towards lower cost[3]. Results from the *Softgoal Simulation* process are presented in table 1. Table 1.a shows the cost/benefit distributions of samples without treatment. Table 1.b shows the distributions after applying treatment to the *KWIC* system. These results shows that variance in behavior decreased and the mean benefit scores improved after treatment applied. The mean benefit drifted from <5.5 before treatment(table 1.a) to <11 after treatment(table 1.b). Moreover, the number of samples fell into the high benefit ranges (≥17) increased after treatment. Base on this result, developers may focus on key issues that would greatly impact overall software quality, such as whether or not to implement *Shared Data* for the system. Alternatively, if in some dispute situation, an analyst could use *c2; c4; c5* as bargaining chips. Since these claims have little overall impact, analyst could offer them in any configuration as part of some compromise deal in exchange for the other key decisions being endorsed.

[3]E.g. class *Cost=zero/Benefit=high* has a higher ranking than class *Cost=one/Benefit=veryhigh*.

3.2 Experiment 2: Weak Quality Assurance Modeling

Often, when outside consultants are called in to offer a rapid assessment on how to improve a problematic project, they seek the fewest actions that offer the most benefit. To handle this situation, we defined a variation of the *KWIC* framework(figure 2.II) to simulate a weaker form of quality assurance.

TABLE 2. Weak quality assurance Percentage distributions of $benefit$s and $cost$s seen in 10,000 runs of fig 2.II

Cost	Benefit <15	<29	<44	≥58.67	total
0					
1	3.1				3.1
2	7.0	10	.62		18
3	4.2	25	12	1.4	43
4	1.7	13	13	4.0	32
5		1.6	1.8	.6	4.3
total	16	50	28	6.0	100

a. No treatment

Cost	Benefit <15	<29	<44	≥58.67	total
0					
1	20				20
2	8.4	28	4.6		41
3	1.8	16	16	4.8	39
4					
5					
total	30	44	21	4.8	100

b. Treatments applied (*Pipe & Filter=n;c2=y;c3=y;c4=y*)

The goal of this experiment is to determine what would negatively impact software quality in the most liberal quality assurance scheme. To handle this, the class rankings are reversed as opposed to that of the rigorous quality assurance scheme. The results are shown in table 2. Comparing table 2.b (after treatments) with table 2.a (before treatments), the number of samples fell into the lowest benefit range (<15) increased, which showed that benefit suffered as treatments accumulated. The results also suggest that, using the weaker form of quality assurance scheme, the overall software quality of the *KWIC* system suffers if *Pipe & Filter* is not implemented. Hence, users may center their discussions on the possibilities of implementing the *Pipe & Filter* option.

4. Case Study: NASA IV&V Activity Prioritization (SR-1 Project)

The experiment discussed in section III demonstrates our simulation technique working on a small example, albeit one often cited in the literature. The SR-1 Project presents in this section shows how the *Softgoal Simulation Tool* scales up to modern real world software.

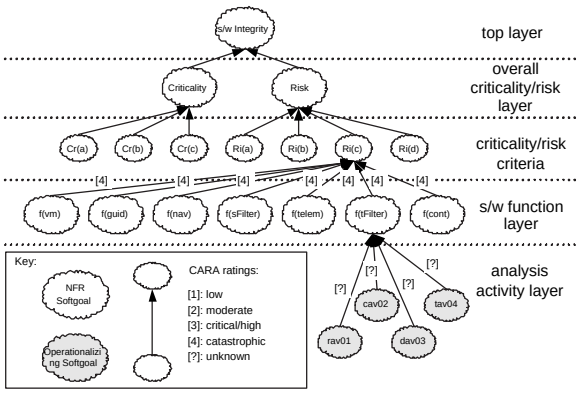

Fig. 3. Segment of the SR-1 framework

4.1 The SR-1 Project

The *NASA* SR-1 [7] project refers to the technologies involved in advance satellite design. *NASA IV&V Facility* is one of the organizations performing Verification & Validation(V&V)[4] on software projects such as SR-1. The *Criticality Analysis and Risk Assessment*(*CARA*) [7] process is a quantitative analysis used by the *NASA IV&V* personnel to determine the appropriate scope of V&V on a project.

Like many other companies, project managers at *NASA IV&V Facility* has to deal with business issues such as delivery deadlines and resource allocations. It is every manager's goal to optimize resource usage, reduce project costs while meeting deadline dates. On the other hand, each *IV&V* analysis activities consumes different degree of resources, and some of these activities perform better in the V&V process than the others. Finding out which of these V&V activities are more powerful, and less costly at the same time, would be helpful for project resource management and task prioritization. The objective of our study, therefore, is to look for the analysis activities that are more cost-effective than others.

In our study of the SR-1 project, we applied the softgoal framework idea to sketch out the inter-dependencies between *IV&V* analysis activities and the *CARA* Ratings on SR-1 functions. Figure 3 shows the resulting SR-1 framework.

As we proceeded on our analysis, we found that the SR-1 softgoal framework displayed typical features of real world systems - lack of domain knowledge and supporting data. We were unable to obtain any expert opinions regarding to the effectiveness of each analysis activities. Therefore, we could not define the *inter-dependencies* between *IV&V* analysis activities and the SR-1 s/w functions. Similarly, we have no information on the *priorities* of SR-1 s/w functions. Moreover, there are discrepancies in the scaling factors for cost calculations. Nonetheless, by making adjustments to the unknowns, we were able to perform analysis and draw useful conclusions. Results are detailed in the next section.

[4]Verification & Validation is a process to evaluate the correctness and quality of a software product throughout its life cycle [8]. Independent V&V (IV&V) is performed by organization that are independent of the development organization.

TABLE 3. SR-1 Variant 1 Percentage distributions of $benefit$s and $cost$s seen in 10,000 runs of fig 4.I

Cost \ Benefit	vlow	low	high	vhigh	Total
vlow	34				34
low		4.0	6.3	5.6	16
high		6.2	10	8.8	25
vhigh		5.6	8.8	11	25
Total	34	16	25	25	100

a. No treatment

Cost \ Benefit	vlow	low	high	vhigh	Total
vlow					
low		4.7	7.7	7.3	20
high		10	16	14	40
vhigh		9.1	14	17	40
total		24	38	39	100

b. Most preferred system (*activity: tav09=y*)

Cost \ Benefit	vlow	low	high	vhigh	Total
vlow					
low					
high		5.2	9.5	8.6	23
vhigh		17	27	33	77
Total		22	36	41	100

c. Least preferred system (*activity: cav10=y*)

4.2 Experiments and Results

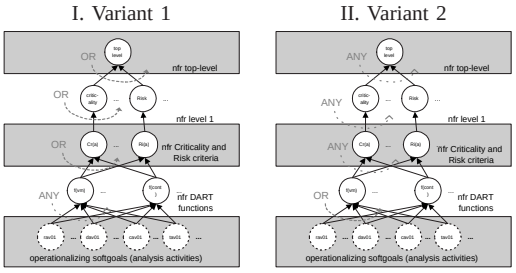

Fig. 4. SR-1 framework variants

We constructed and performed experiments on two variations of the SR-1 framework that differed in their logical compositions. Figure 4.I shows the weakest quality assurance scheme, whereas figure 4.II represents the strongest. The class ranking scheme used for these frameworks is similar to that of the *KWIC* framework discussed in section 2.1. Two studies were conducted on each variant, based on the same ranking function:

- In the **most preferred** study, TAR2 looks for behaviors that would contribute to the integrity of the SR-1 functions;
- Conversely, in the **least preferred** study, we reversed the order of class ranks to find treatments that would assert negative impacts;

TABLE 4. SR-1 Variant 2 Percentage distributions of $benefit$s and $cost$s seen in 10,000 runs of fig 4.II

Cost	Benefit vlow	low	high	vhigh	Total
vlow	18	2.2	2.7	2.5	25
low	3.8	8.8	6.2	6.2	25
high	2.5	8	7.1	7.4	25
vhigh	1.1	6	9.1	8.9	25
Total	25	25	25	25	100

a. No treatment

Cost	Benefit vlow	low	high	vhigh	Total
vlow	25	3.1	3.8	3.6	36
low	4.9	12	8.4	8.7	34
high	2.3	7.7	6.7	7.6	24
vhigh		1.6	2.2	2.3	6.4
Total	33	24	21	22	100

b. Most preferred system (*activity: dav12=n*)

Cost	Benefit vlow	low	high	vhigh	Total
vlow					
low	3.4	5.7	1.7	2.1	13
high	5.5	16	7.7	7.0	36
vhigh	4.1	13	17	17	51
Total	13	34	26	26	100

c. Least preferred system (*activity: cav07=y*)

Monte Carlo simulations were applied to each framework variant twice, each time with different cost functions. Treatment learning was then applied to each set of data in finding the most/least favorable treatments. The effects of these treatments were compared with the control situations (i.e. no treatment) in terms of costs and benefits. Results from these experimentations are presented in two groups: table 3 for weak quality assurance; and table 4 for rigorous quality assurance.

Similar to that of the *KWIC* framework, the class ranking scheme for SR-1 accounted on both benefit and cost with slight preference towards lower cost. As a result, treatment learner recommended treatments that sacrifice a lower benefit for a lower cost. Our result sets in tables 3 and 4 reflected this particular setting.

Several features of these results deserve comment. Consider the results on SR-1 framework variant 1 for weak quality assurance(figure 4.I): firstly, treatment eliminated samples within the *Cost=vlow*, *Benefit=vlow* range, from 34% to 0%. Secondly, for the **most preferred** system(table 3.b), treatment learning drove the sample distributions towards a higher benefit range (*Benefit=[high,vhigh]* occupied 77%, as opposed to 50% with no treatment). Third, the distributions of benefit received after treatments were roughly the same for both **most preferred** and **least preferred** system. However, the **least preferred** system (table 3.c) suffered from very high cost (77% in *Cost=vhigh*), compared to the **most preferred** system (40%).

The result for the rigorous quality assurance (figure 4.II) scheme is presented in a similar fashion. First of all, treatments for the **most preferred** system(table 4.b) resulted in an increase of samples in the *Cost=[vlow,low]* range (from 50% to 70%. However, the samples within the range *Benefit=[vlow,low]* also increased (from 50% to 57%), a clear indication on the proportionality of cost and benefit. On the other hand, treatment for the **least preferred** system(table 4.c) resulted in very high cost (87% in the *Cost=[high,vhigh]* range) compared to that of no treatment (50%). The benefit increased only slightly, from 50% to 52% in the *Benefit=[high,vhigh]* range.

To sum up the SR-1 case study:

- The use of logic components significantly affected the treatments that TAR2 recommended;
- Variation on cost functions was found to have no observable effect to the resulting treatment recommendations from all SR-1 framework variants studied;
- For all experimentations on the SR-1 framework, TAR2's treatment recommendations have been 10-way cross-validated [9] and their trustworthiness ensured. In other words, all of TAR2's treatment recommendation remained stable, in spite of the uncertainties lay within the framework;
- For the results shown in table 4.b, TAR2 suggested not to do activity *dav12* would be beneficial. Our treatment learning method can give advice on which activities not to be done in order to receive the most preferred outcome;

5. Conclusion

From what we have observed in our experiments on both *KWIC* and SR-1 frameworks, our simulation tool successfully discovered consistent behaviors within each framework despite various uncertainty factors, and provided treatment recommendations relevant to business concerns. As this approach does not require much concrete domain knowledge, time and expenses dedicated to data collection (e.g. appointments with domain experts, gathering surveys) can be minimized. Moreover, TAR2's treatments found the most critical decision towards the problem domain, thus users can focus on this key issue and allot less time in discussing the non-critical ones.

References

[1] Chung, Nixon, Yu, and Mylopoulos, *Non-Functional Requirements in Software Engineering.* Kluwer Academic Publishers, 1999.

[2] T. Menzies, "Practical machine learning for software engineering and knowledge engineering," in *Handbook of Software Engineering and Knowledge Engineering*. World-Scientific, 981-02-4973-X, December 2001.

[3] T. Menzies, E. Chiang, M. Feather, Y. Hu, and J. Kiper, "Condensing uncertainty via incremental treatment learning," *Annals of Software Engineering; special issue on Computational Intelligence*, 2002.

[4] M. Kalos and P. Whitlock, *Monte Carlo Methods, Volume 1: Basics.* New York: J. Wiley, 1986.

[5] T. Menzies and Y. Hu, "Constraining discussions in requirements engineering via models," in *International Workshop on Model-based Requirements Engineering*, 2001.

[6] H. Simon, "Rational choice and the structure of the environment," in *Models of Man*, H. Simon, Ed. John Wiley, NY, 1957.

[7] T. S. Corporation, *IV&V Catastrophic/Critical/High Risk Function List*, 2002.

[8] Nasa iv&v web site. [Online]. Available: http://www.ivv.nasa.gov/

[9] K. R, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *IJCAI-95*, 1995.