01 Apr 2010

Using Quality Attributes and Computational Intelligence to Generate and Evaluate System Architecture Alternatives

Atmika Singh

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

A. Singh and C. H. Dagli, "Using Quality Attributes and Computational Intelligence to Generate and Evaluate System Architecture Alternatives," *Proceedings of the 2010 4th Annual IEEE Systems Conference*, Institute of Electrical and Electronics Engineers (IEEE), Apr 2010.
The definitive version is available at https://doi.org/10.1109/SYSTEMS.2010.5482332

Using Quality Attributes and Computational Intelligence to Generate and Evaluate System Architecture Alternatives

Atmika Singh
Smart Engineering Systems Laboratory
Missouri University of Science and Technology
Rolla, Missouri-65401
Email: asfff@mst.edu

Cihan H. Dagli
Smart Engineering Systems Laboratory
Missouri University of Science and Technology
Rolla, Missouri-65401
Email: dagli@mst.edu

***Abstract*—This paper proposes an evolutionary algorithm based approach for evolving architecture alternatives using quality attributes as design drivers. A novel fuzzy architecture assessment approach is presented to quantitatively evaluate the set of possible solutions based on linguistic assessments of architecture quality attributes elicited from the stakeholders. The proposed approach makes a valuable contribution to the systems architecting knowledge base by presenting a measurable and quantifiable approach to architecture design and evaluation.**

I. Introduction

Early in the design life-cycle,the systems architect generates concept alternatives, evaluates them, and selects the best one for further refinement. The design trade space of a complex system is vast, and overlooking potential architecture alternatives can have an adverse impact on the final outcome. The use of computationally intelligent techniques such as Evolutionary Algorithms (EA), Genetic Algorithms (GA) in particular, is proposed for searching the design trade space for candidate architectures. The evolutionary nature of the architecture search process lends itself to computationally intelligent approaches especially evolutionary algorithms. However, the early design phases are also marked by high levels of ambiguity about the final form and function of the desired artifact. Thus, there exists the need for a methodology to automatically generate, evolve and evaluate architecture alternatives using vague and incomplete information.

To make a decision about the success or failure of a candidate architecture, it needs to be evaluated over a set of criteria that can accurately assess the system's ability to deliver its intended functionality and be acceptable to stakeholders and decision-makers. Measures of Effectiveness (MOE), such as affordability, flexibility, security, reliability etc. MOEs are also referred to as Architectural Quality Attributes (AQA) in the software domain and are used by stakeholders to qualify the value delivered by a system [1,2]. These are a set of characteristics that the architecture of a system may have. The interactions between the desired function of the system and its quality attributes determine the effectiveness of a system in delivering the target functionality. Traditional architecting approaches propose the development of system functional architectures which are then adapted to satisfy system quality requirements. It has long been accepted by the software community [2,3,4] that quality attributes of a system are systemic properties that are manifested by the chosen system architecture. Accommodating them into the system architecture as an after-thought is neither easy nor very effective. Reference [5] has discussed the importance and applicability of this point of view to systems engineering. This paper proposes an approach for evolving and assessing concept alternatives by leveraging architectural quality attributes as design drivers. Quality attributes of a system may complement or conflict with each other. An architect can leverage these quality attributes to perform design trade-offs based on the impact of the design decision on an associated AQA. Each architectural decision will result in the achievement of a set of AQAs to a particular level. Based on the overall impact of a set of design decisions on the AQAs, concept alternatives can be developed. A novel fuzzy architecture assessment approach has been developed to quantitatively evaluate the set of possible solutions based on a set of AQAs specified by the stakeholders. By combining the fuzzy architecture assessment approach with an EA, architecture alternatives can be generated and evaluated automatically.

II. Architectural Quality Attributes as Design Drivers

Architectural quality attributes have been defined as a set of key behavioral attributes unique to a system [1,3,4,6]. For this research, AQAs are defined as proposed by [1]-" ... *are standards against which the capability of a solution to meet the needs of a problem may be judged*". They are the criteria by which the stakeholder judges the success of a system and thus, they need to be specified by the stakeholders[6]. Quality attributes directly or indirectly enable the system to deliver its intended function to the satisfaction of the stakeholders. Previous work has established the utility of using quality requirements as design drivers [1,2,3,4,5,6,7]. Attribute Driven

Design [2] uses a set of predetermined architectural patterns known as attribute primitives to achieve quality attributes during software design. Related work by [7,8] has applied quality attribute scenarios derived from business objectives to develop architectural tactics that correspond to each quality attribute. Quality attribute workshops(QAW) have been suggested as a means to elicit AQAs from the stakeholders [9]. QAWs elicit quality attribute requirements and their mapping to overall system objectives. A case for the use of quality attributes in systems architecting as a mechanism for making objective decisions about architectural trade-offs and for predicting how well candidate architectures will meet customer expectations was made by [5]. Reference [10] demonstrated the use of decision science and value focused thinking to aggregate AQAs in order to develop an assessment for a candidate conceptual architecture. We propose the use of fuzzy set theory as means for aggregating AQAs to develop a quantitative assessment of an architectural alternative.

A. Aggregation of AQAs Using Fuzzy Set Theory

The ambiguity in the architecting process can be captured very effectively by using fuzzy representations and rules that can be represented in the form of Fuzzy Associative Memories (FAM). These FAMs can successfully capture both the domain knowledge of the system and the value judgments of the customers.

To ensure operational feasibility of the resulting system configuration, key attributes need to be engineered into the system right from the start. Many of these system attributes are ill-defined and difficult to quantify, especially in the early stages of the architecting process. These attributes exhibit non-linear cross-functional relationships that are difficult to determine and model. To develop an acceptable system, the architect must consider the trade-offs between the desired MOEs. The expectations and preferences of the stakeholders are usually stated linguistically. These linguistic preference structures may be incomplete, ambiguous and subjective. Fuzzy logic is a super-set of classical logic that allows computation with words and imprecise relationships between linguistic variables. For this reason, it is proposed that fuzzy logic forms a natural bridge between the objectivity of mathematical rigor and the naturally ambiguous quantities being considered. It does not need accurate quantitative inputs and allows design engineers to describe the system's expected performance in linguistic terms which can be manipulated with fuzzy set theory. A brief introduction to fuzzy logic and fuzzy rules follows.

1) Classical vs. Fuzzy Set Theory: Fuzzy logic extends classical logic to all values in the interval [0,1]. A fuzzy set is a class of objects in which there is no sharp boundary between those objects that belong to the class and those that do not. In a fuzzy set, an object may have a grade of membership intermediate between full membership, represented by 1, and non-membership, represented by 0. The principle difference between classical and fuzzy set theory is the idea of partial membership. In fuzzy set theory, an element can assume degrees of membership between 0 and 1. Fuzzy sets allow an element of a set to have "a lot" of membership in a set $X1$ and "a little" membership in a set $X2$. This characteristic makes fuzzy sets ideal for representing poorly defined and ambiguous system attributes. A full treatment on the mechanics and theory of fuzzy sets and fuzzy logic are beyond the scope of this paper but are well established in published literature [11]. However, some fundamental concepts of fuzzy set theory used in this paper are briefly discussed next.

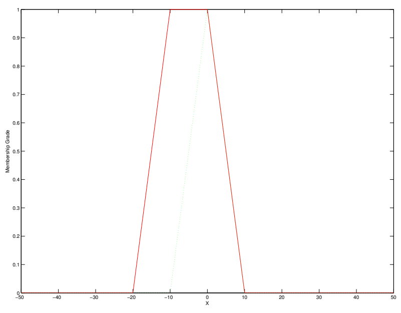

Fig. 1. Fuzzy number representation

2) Fuzzy Numbers and Fuzzy Arithmetic: Fuzzy sets defined on the set of real numbers can be viewed as fuzzy numbers or intervals under certain conditions [11]. Triangular and trapezoidal membership functions are the most commonly used forms of fuzzy numbers. Trapezoidal numbers are represented by a quadruple of the form $X = \phi(a, b, c, d)$. A triangular fuzzy number is a special case of the trapezoidal fuzzy number when $b = c$. Thus the triangular fuzzy number reduces to a triple,$X = \phi(a, b, d)$. Fig. 1 shows a membership function of a trapezoidal fuzzy number $A = \phi(-20, -10, 0, 10)$ and a triangular fuzzy number $B = \phi(-10, 0, 10)$.
Let $X_1 = \phi(a_1, b_1, c_1, d_1)$ and $X_2 = \phi(a_2, b_2, c_2, d_2)$ Basic arithmetic operations on fuzzy intervals,

$$A + B = \phi(a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2)$$
$$A - B = \phi(a_1 - a_2, b_1 - b_2, c_1 - c_2, d_1 - d_2)$$

3) Fuzzy Inference System: A fuzzy inference system can be used to encode expert knowledge in the form of a rule base for evaluating a given set of inputs. These inference systems can provide a convenient and reliable way of handling uncertain and imprecise value judgments. A fuzzy rule base is a collection of fuzzy propositions, which are statements that takes a fuzzy truth value. A fuzzy proposition includes logical connectives like AND, OR, NOT and Implication. A *fuzzy rule* is a proposition in the if-then format. It contains one or more *fuzzy propositions* joined by a main implication connective. Consider two fuzzy sets 'Affordability' and 'Reliability' whose values are stated in *linguistic terms* $\{\text{High}, \text{Medium}, \text{Low}\}$. A typical *fuzzy rule* will take the form,

IF ((Affordability is Low) **AND** (Reliability is High)) **THEN**(Performance is Unacceptable) where 'Performance' is an output variable and'Unacceptable' is a membership function defined on 'Performance'.

III. Automatic Architecture Generation Using EA

The fuzzy architecture assessment approach discussed in the previous section makes it feasible for a systems architect to quantitatively evaluate a set of possible solutions and select the most suitable design. The next obvious step is the generation of suitable candidate architectures for consideration. In any large scale system, the design trade space is inherently vast making it difficult to explore all possible architecture variants. Evolutionary algorithms have long been successfully used as design space exploration techniques in many engineering disciplines [12,13]. Even though EAs have found use in the design optimization of complex systems most applications are at the detailed design level where design peroformance measures are explicitly known [14,15]. Reference [16] uses EAs for automating design exploration at the component level. A novel EA based trade space exploration strategy is proposed for explicating all possible system architectures at the conceptual level. Excellent discussions on the fundamental theory of EAs can be found in [17].

The EA based trade space exploration strategy simulates the biological process of natural selection to generate a set of potential design concepts. The algorithm proceeds by generating a population of candidate architectures, modifying the parent population using predetermined genetic operators, evaluating their 'fitness' and finally selecting a child population. This process is continued iteratively until a near-optimal set of solutions emerges. In a systems context, each individual in a population may represent a possible functional, behavioral or structural architecture. Child populations are created by selecting features from a random set of parents to create novel architectures that embody the best aspects of the parent designs.

IV. Concept Generation Methodology

The process of generating architectures at the conceptual design level primarily consists of selecting design approaches or strategies to achieve a system's functional objectives. An evolutionary process can help weed out suitable architecture alternatives based on their impact on the AQAs and the cost of implementing each alternative. Thus a selected concept will consist of a set of approaches that when implemented will help a system deliver its target functionality with the most optimal balance of AQAs and cost.

Each possible approach is linked to a set of AQAs which in turn determine the overall acceptability of a candidate architecture. For example, robustness can be engineered into a system by using architecting strategies that strengthen the reliability of components and their interconnections. These may include adding redundancy (Hardware and software) and providing alternate paths within the structure. Building in modularity may also have a positive impact on robustness, however it may increase cost. It should be noted that at the conceptual level these strategies simply indicate the nature of the solution without dictating what that solution must be. The system designers can then decide whether to use redundant hardware, or other architectural decisions to increase the system's robustness.

The major steps involved in modeling the problem space for the proposed approach are listed. In defining these we draw on ideas from Keeney's value function theory [18] and the ADD [1] approach.

- Based on the need statement and requirements definitions, identify the key externally delivered function or value of the system to be architected.
- Using QAW described in [9], identify and finalize the AQAs that the stakeholders will use to judge the acceptability of the system architecture. It is vital that the set of AQAs be properly grounded in the system of interest, failing which the subsequent attribute-strategy hierarchy may not address the system in question.
- Once the externally delivered function and its qualifying attributes have been identified, they should be organized into a hierarchy of attributes and sub-attributes. At the lowest tier, each sub-attribute must be achievable by the implementation of a unique architectural strategy. Once all the strategies necessary to achieve the identified sub-attributes have been ascertained, these are placed at the bottom of the attribute-strategy hierarchy.
- The next step is to generate expert evaluations of the impact of adopting a strategy on the attributes on the next higher level of the hierarchy. The experts present their evaluations in linguistic terms while simultaneously assigning a fuzzy cost value to the strategy itself.

The architecture exploration starts with an initial architecture concept represented by a set of architectural strategies. Evolutionary operators like crossover and mutation are then applied to the initial set to search for concept variants. The fitness or acceptability of each concept variant is determined based on its impact on the AQAs. In order to automate the search process, two key elements are required in addition to the mechanics of natural selection: a goal function or fitness assessment and a representation of the architecture.

A. Problem Representation

A key component of automated intelligent search is a suitable representation of the system architecture. This representation is in essence a model of the conceptual architecture itself. The genotype representation is highly dependent upon the system being modeled and the level of ambiguity involved. The attribute-strategy hierarchy is used to generate the genotypic representation for the EA. Each potential strategy is represented by a digit in a binary string that represents the chromosome. A '0' means selection and '1' represents rejection of the strategy. Depending on the structure of the hierarchy chromosomes may have more than one bit strings to represent strategies are different hierarchical levels.

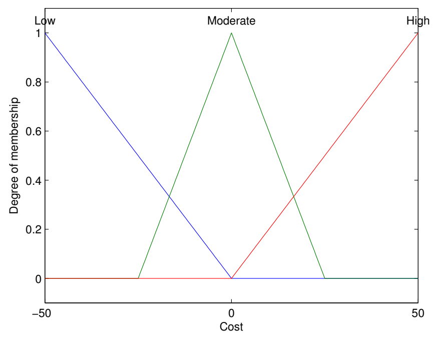

Fig. 2. Linguistic term set for cost

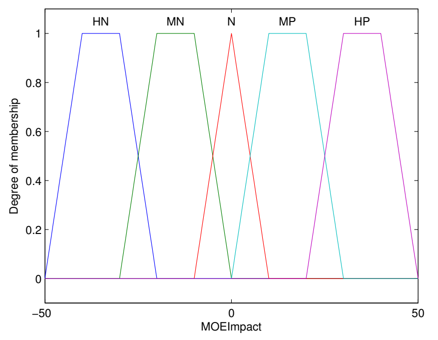

Fig. 3. Linguistic term set for Attribute Impact

B. Fuzzy Fitness Assessment

The 'fitness' of a solution depends on how well it satisfies the designers' requirements and constraints. Creating a robust and repeatable fitness function is the most challenging aspect of EA based design search. This problem is addressed by the use of the fuzzy architecture assessment technique discussed. The fuzzy fitness assessor can aggregate multiple objectives into a single fitness measure allowing system architects to rank potential solutions by level of 'acceptability'. The inputs to the fuzzy fitness function are the linguistic assessments of the impact and cost of the architectural strategies. A sample linguistic term sets representing these assessments is,

$$\begin{aligned} AttributeImpact = \{&HighNegative(HN),\\ &ModerateNegative(MN),\\ &Neutral(N), ModeratePositive(MP),\\ &HighPositive(HP)\}\\ Cost = \{&Low, Moderate, High\} \end{aligned}$$

As shown in Fig. 2 and Fig. 3, the attribute impact assessments are assigned positive or negative evaluations based on the nature of their impact on the AQAs. Since cost cannot be negative, the cost assessments are generated on a positive scale. The granularity of these term sets can be chosen based on the nature of the information available. Higher granularity can be used when more precise information is available. These

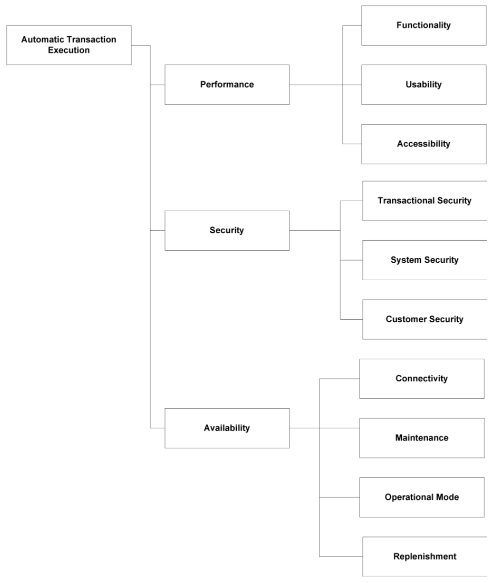

Fig. 4. Attribute hierarchy for the ATM

assessments have to be aggregated into a single fuzzy number in order to determine the overall impact for each of the AQAs. This is done by adding all positive impact assessments and subtracting all negative assessments for each individual AQA. Let the impact of a set of strategies $S_1, S_2, S-3$ on an attribute A_1 be $S_1 = \text{'}MN', S_2 = \text{'}N', S_3 = \text{'}HP'$. Then the aggregate impact on the attribute A_1 can be calculated using fuzzy addition of trapezoidal numbers as follows,

$$\begin{aligned} A_1 &= \phi(-30,-20,-10,0) + \phi(-10,0,0,10)\\ &\quad +\phi(20,30,40,50)\\ &= \phi(-20,10,30,60) \end{aligned}$$

The aggregated impact values for the AQAs are then input into a fuzzy inference system uses a fuzzy rule base to non-linearly map the AQAs to generate an overall acceptability rating for the candidate architecture. The rule base is designed based on trade-offs between the AQAs as stated by the stakeholders. The acceptability rating is used as a fitness measure to rank architectural alternatives.

V. Proof of Concept

The conceptual architecture of an Automated Teller Machine (ATM) is generated as a proof of concept. The externally delivered function of the ATM was identified as 'automatic transaction execution' The key Quality Attributes of the ATM as specified by the stakeholders include security (Se), availability (A) and performance (P). The attribute hierarchy of the ATM is shown in Fig. 4. Each AQA has been decomposed into sub-attributes that are directly representative of the strategies used for achieving the specified AQA. Table I lists some of the strategies for achieving the associated AQA.

A. Chromosome Representation

Each chromosome is a binary string with 24 digits, representing the 24 architectural strategies to choose from. A '1'

TABLE I
ARCHITECTURAL STRATEGIES FOR OPERATIONAL MODE AND REPLENISHMENT

Sub-attribute	Strategy
Operational Mode	off-line, on-line
Replenishment	Manual, Partially automated

TABLE II
FUZZY ASSESSMENTS FOR COST AND ATTRIBUTE IMPACT

Strategy	Performance	Security	Availability	Cost
S1	HP	N	N	Low
S2	HP	N	MN	Moderate
S3	HP	N	N	Low
S4	HP	N	N	Low
S5	HP	N	N	Low
S5	MP	N	N	Moderate
S6	HP	N	N	High
S7	MP	N	MP	Moderate
S8	HP	MP	HP	High
S9	MP	MP	N	Low
S10	HP	HP	MN	High
S11	MP	MP	N	Low
S12	N	HP	N	Moderate
S13	N	HP	N	Moderate
S14	N	MP	N	Low
S15	N	HP	N	High
S16	MN	HP	HP	Low
S17	MP	MP	MP	Moderate
S18	HP	MP	N	High
S19	N	MN	HP	Low
S20	N	MP	MN	High
S21	MN	MN	HN	Low
S22	HP	MP	HP	High
S23	N	MN	MN	High
S24	N	MP	MP	Moderate

indicates selection and a '0' indicates rejection of a strategy.

B. Fitness Evaluation

The attribute impact assessments and cost assessments for the ATM are shown in Table II. These assessments were elicited from the stakeholders and other experts in the domain. The membership function definitions for these linguistic assessments were shown in Fig. 2 and Fig. 3. The rule base for combining the AQA aggregated ratings into a combined architecture assessment rating was used to capture the nonlinear mapping from the AQA to the architecture rating in the form of IF-THEN rules. These rules are also developed based on inputs by domain experts and attribute rankings provided by the stakeholders.

C. Genetic Algorithm Implementation

A genetic algorithm with a fuzzy fitness assessor was implemented to quickly generate and evaluate alternate solutions for the ATM problem. The algorithm begins by randomly initializing a population of size N and calculating the fitness with the fuzzy assessor. A binary tournament selection procedure was employed to select the chromosome for crossover. A tournament size of 2 was used. The selected chromosome in the population was crossed over with a randomly selected second chromosome. Crossover was performed with a fixed probability. Crossover was double, as each crossover produced two offspring. Mutation was performed with a low fixed probability at a randomly selected location. Bit mutation operation is incorporated in the algorithm to aid in forestalling the problems of premature convergence associated with the repeated use of crossover. The elitism operator was active for the crossover and mutation. This means that the chromosome with the highest fitness in a generation was not crossed-over or mutated. The child population was ranked on the basis of fitness and the best chromosomes were chosen to form part of the next generation. If the termination criterion is satisfied, the algorithm terminates.

TABLE III
FINAL CHROMOSOME

Chromosome	101111111111111110010010

VI. RESULTS

Bit-wise mutation and double crossover operations were used. The algorithm was run for a 500 generations and a mutation rate of 0.1 was used. The crossover rate was set to 0.6. The genetic algorithm was implemented with an initial population size of 20. The sensitivity of the fitness value to changes in population size are shown in Fig. 5. It can be seen that an increase in population size leads to a decrease in the time taken to achieve the same level of fitness. The highest fitness value achieved is approximately 90. This fitness represents the most favorable set of strategies from the original set of 24 strategies. Fig. 6 shows the sensitivity analysis results for varying mutation rates while maintaining constant cross over rates. As the mutation probability increases the convergence speed increases. The final chromosome for the architectural strategy set with the highest fitness value is shown in Table III. The final concept selected is for an ATM that includes strategies S1, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S20 and S23. The eliminated strategies include partially automated replenishment, remote maintenance, biometric authentication and voice based input to the ATM. One of the observations is that each of the eliminated strategies had a significantly negative impact on the cost. This is expected since a higher weight-age was given to cost in the rule base used for combining the quality attributes.

VII. CONCLUSION

A system architecture design methodology that evolves architectures based on stakeholder quality expectations as design drivers and an intelligent decision-making algorithm is presented. A successful architecture manifests the earliest design decisions and these decisions are the most important in the life cycle of a system. To successfully meet customer

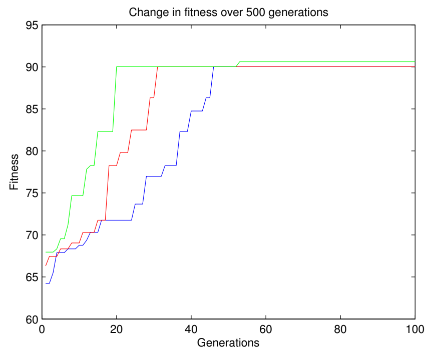

Fig. 5. Fitness with varying population sizes

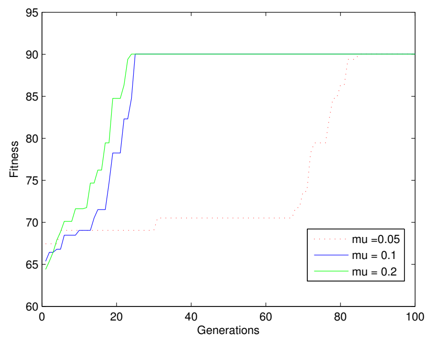

Fig. 6. Fitness with varying mutation probabilities

expectations, we must address quality expectations early in the life cycle. The ability of a system to meet these quality expectations is dependent on its architecture. There exists a need for techniques and tools that support the development of candidate system architectures and perform trade-offs based on a set of system attributes which can then be used to evaluate the success of an architecture. The proposed approach contributes to the systems architecting knowledge base by presenting a measurable and quantifiable approach to architecture design and evaluation. As future work the concept selection methodology will be extended to automate all aspects of architecture generation, from concept selection to component selection.

References

[1] N. Sproles, “Coming to grips with measures of effectiveness,” Systems Engineering, vol. 3, 2000, pp. 50-58.

[2] L. Bass, M. Klein, and F. Bachmann,“Quality attribute design primitives and the attribute driven design method,” Lecture Notes in Computer Science, 2002, pp.169-186.

[3] J. Voas, “Software’s secret sauce: The“-ilities,” IEEE SOFTWARE, 2004, pp. 1415.

[4] J. Voas and W.W. Agresti, “Software quality from a behavioral perspective,” IT PROFESSIONAL, 2004, pp. 40-50.

[5] P. Croll, “Quality Attributes - Architecting Systems to Meet Customer Expectations,” Systems Conference, 2008 2nd Annual IEEE, 2008, pp. 1-8.

[6] N. Sproles, “Formulating measures of effectiveness,” Systems Engineering, vol. 5, 2002, pp. 253-263.

[7] I. Ozkaya, L. Bass, R.S. Sangwan, and R.L. Nord, “Making practical use of quality attribute information,” IEEE SOFTWARE, 2008, pp. 25-33.

[8] D. Sims, “How Business Goals Drive Architectural Design,” Computer, 2007.

[9] M. Barbacci et al., *Quality Attribute Workshop Participants Handbook*, special report CMU/SEI-2000-SR-001, Software Engineering Institute, Carnegie Mellon Univ., 2000

[10] N. Smith and T. Clark, “A Framework to Model and Measure System Effectiveness,”ICCRTS,Cambridge, 2006.

[11] G.J. Klir and B. Yuan, Fuzzy sets and fuzzy logic: theory and applications, Prentice Hall Upper Saddle River, NJ, 1995.

[12] M.S. Bittermann, Ciftcioglu, and I.S. Sariyildiz, “A cognitive system based on fuzzy information processing and multi-objective evolutionary algorithm,” IEEE Congress on Evolutionary Computation, 2009. CEC’09, 2009, pp. 1271-1280.

[13] Y. Li, “An Intelligent, Knowledge-based Multiple Criteria Decision Making Advisor for Systems Design,” Citeseer, 2007.

[14] R.J. Terrile, M. Kordon, D. Mandutianu, J. Salcedo, E. Wood, and M. Hashemi, “Automated design of spacecraft power subsystems,” IEEE Aerospace Conference, 2006, p. 14.

[15] R.J. Terrile, M. Kordon, M. Postma, J. Salcedo, D. Hanks, and E. Wood, “Automated Design of Spacecraft Telecommunication Subsystems Using Evolutionary Computational Techniques,” 2007 IEEE Aerospace Conference, 2007, pp. 9.

[16] A. Sutcliffe, W. Chang, and R.S. Neville, “Applying Evolutionary Computing to Complex Systems Design,’ IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 37, 2007, pp. 770-779.

[17] C.A. Coello, G.B. Lamont, and D.A. Van Veldhuizen, “Evolutionary algorithms for solving multi-objective problems,” Springer-Verlag New York Inc, 2007.

[18] R.L. Keeney and H. Raiffa, “Decisions with multiple objectives: Preferences and value tradeoffs,” Cambridge Univ Pr, 1993.