# New Approach to Requirements Trade-Off Analysis for Complex Systems

Jonathan Lee, *Member*, *IEEE Computer Society*, and Jong-Yih Kuo

**Abstract**—In this paper, we propose a faceted requirement classification scheme for analyzing heterogeneous requirements. The representation of vague requirements is based on Zadeh's canonical form in test-score semantics and an extension of the notion of soft conditions. The trade-off among vague requirements is analyzed by identifying the relationship between requirements, which could be either conflicting, irrelevant, cooperative, counterbalance, or independent. Parameterized aggregation operators, *fuzzy and*/*or*, are selected to combine individual requirements. An extended hierarchical aggregation structure is proposed to establish a four-level requirements hierarchy to facilitate requirements and criticalities aggregation through the *fuzzy and*/*or*. A compromise overall requirement can be obtained through the aggregation of individual requirements based on the requirements hierarchy. The proposed approach provides a framework for formally analyzing and modeling conflicts between requirements, and for users to better understand relationships among their requirements.

**Index Terms**—Vague requirements, requirements specifications, requirements trade-off analysis, requirements classification, fuzzy logic.

## 1 Introduction

A MAJOR challenge in requirements engineering of complex systems is that the requirements to be captured are imprecise in nature and usually conflicting with each other [24], [33], [43]. Balzer et al. have argued that informality is an inevitable (and ultimately desirable) feature of the specification process [1]. Similar ideas are also advocated by other researchers such as Borgida et al. [3], Feather [14], Fickas et al. [8], Reubenstein and Waters [32], and Niskier et al. [29]. Borgida et al. have further elaborated that a good requirement modeling approach should take the problem of describing nature kinds into account, which usually runs the risk of being *vague* and subject to *contradiction* [3].

However, most of the existing work on requirements modeling are limited in dealing with this problem. Traditional requirements modeling approaches (formal or informal) either require the requirements be stated precisely or completely exclude this problem out of the scope of the modeling activity (e.g., see [9] for a survey). Knowledge-based software engineering indirectly addresses problems caused by the vagueness in the specification process by converting domain-specific informal requirements into formal ones [21], [26].

In this paper, we argue that there are various kinds of information in the informal requirements (i.e., requirements are heterogeneous), and that to appropriately model the requirements, there is a need for distinguishing those information in the requirements. We propose a requirements classification scheme for classifying informal requirements based on the principles of faceted classification scheme [31]. Four facets are identified for characterizing the requirements: content, uncertainty, vagueness, and competence.

Soft functional requirements based on fuzzy logic [44], and an extension of the notion of soft conditions in TBSM [42], [25] are to alleviate the difficulties in representing vague requirements. More specifically, the soft functional requirement is represented using the canonical form in test-score semantics [45]. The trade-off among requirements is analyzed by identifying the relationship between requirements which could be either conflicting, irrelevant, cooperative, counterbalance, or independent. Parameterized aggregation operators, *fuzzy and*/*or*, are selected to combine individual requirements. An extended hierarchical aggregation structure is proposed to establish a four-level requirements hierarchy to facilitate requirements and criticalities aggregation through the *fuzzy and*/*or*. A compromise overall requirement can be obtained through the aggregation of individual requirements based on the aggregation hierarchical structure. The proposed approach, called RTA (Requirements Trade-off Analysis), provides a framework for formally analyzing and modeling vague functional requirements.

This paper is organized as follows. We first introduce the requirement classification in Section 2. The notion and formalization of soft functional requirements is discussed in Section 3. The proposed approach for analyzing soft functional requirements is presented in Section 4. RTA is applied to the ISPBEX expert system as an example to illustrate the potential benefits of our approach. Related work including multicriteria optimization problem, AI planning and multi-perspective specification, is described in Section 5. Finally, we summarize the potential benefits of the proposed approach and our future research plan in Section 7.

• *The authors are with the Software Engineering Laboratory, Department of Computer Science and Information Engineering, National Central University, Chungli, Taiwan 32054, Republic of China.*
*E-mail: {yjlee, cs8036}@se01.csie.ncu.edu.tw.*

*Manuscript received 29 Jan. 1996.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 104009.*

## 2 Requirements Classification

As was observed by Blum that informal requirements are heterogeneous, and can be organized in a variety of ways [2]. In this section, a faceted requirements classification scheme is proposed to facilitate the representation and analysis of informal requirements, which can be considered as an extension of Blum's classification in [2].

To characterize the heterogeneous requirements in the proposed framework, informal requirements are viewed as a collection of statements. Each statement can be classified under the four facets we have identified: content, uncertainty, vagueness, and competence. The most common facet is the contents of requirements: functional and nonfunctional [35]. The construction of functional requirements involves modeling the relevant internal states and behavior of both the component and its environment. Nonfunctional requirements usually define the constraints that the product must satisfy. It is of interest to note that formally specifying the nonfunctional requirements is difficult.

The second and third facets are related to the imperfect information in informal requirements. Fuzzy logic researchers have identified several categories for classifying such information:

- Zimmermann has distinguished uncertainty from imprecision [46]. Uncertainty due to lack of information is called stochastic uncertainty by contrast to the vagueness (fuzziness) concerning the description of the semantic meaning of statements. Imprecision is meant in the sense of vagueness.
- Dubois and Prade have pointed out that imprecision and uncertainty can be considered as two aspects of imperfect information: imprecision relates to the content of an item of information while uncertainty related to its truth (or conformity) to a reality [10]. In [11], the notion of imprecision has been further elaborated from the set-theoretic viewpoint. The meanings of a precise, an imprecise, and a vague statement are interpreted as a singleton, a crisp subset, and a fuzzy subset, respectively.
- Klir has divided uncertainty into two types: vagueness and ambiguity [23]. Vagueness is associated with the difficulty of making sharp or precise distinctions in the world. On the other hand, ambiguity is associated with one-to-many relations, that is, situations with two or more alternatives that are left unspecified.

Our notion of vagueness is adopted from Dubois and Prades' definition in [11] in the sense that vagueness is related to the semantic meaning of the statements, and can be described using three terms: vague, imprecise or precise. Uncertainty is associated with the users needs, and can be best described as follows: If the information in the informal requirements are only partial known, which is resulted from poorly understood users' needs, then the requirements are uncertain. On the other hand, if the requirements are completely known, then the requirements are certain.

The final facet concerns the competence of the requirements, that is, to discern if the requirement is rigid or soft [42]. A rigid requirement must always be satisfied. A soft requirement is a requirement that is desired to be satisfied. Similar concepts can also be found in other disciplines, for example,

1) in verification & validation of expert systems, Rushby distinguished a minimum competence from a desirable competence [36], and
2) in constraint-based reasoning, Borning et al. considered a solution could fall into a spectrum of admissibility from the admissible solution to the optimal one [19].

The proposed requirements classification scheme is summarized in Table 1.

Traditional requirements modeling approaches only distinguish the functional requirements from nonfunctional ones. No distinction has been made between rigid and soft requirements (i.e., all requirements are treated as rigid). Furthermore, the issues of uncertainty and vagueness involved in the informal requirements were excluded out of the scope of modeling activity. Based on our classification scheme, the famous informal requirements of library problem described in [20], can then be classified under the category of <*functional, certain, precise, rigid*>. The open requirements (i.e., the needs are poorly understood) that Blum attempted to address in [2] can be interpreted as a requirement of <*functional, uncertain, precise, rigid*>. The emphasis of the paper is on functional, certain and vague requirements with regard to soft competence (called soft functional requirements).

## 3 Soft Functional Requirements

In this section, we propose the use of *soft functional requirements*, an extension of the notion of soft conditions in TBSM [42], to explicitly capture the imprecision of functional requirements. In TBSM, the functionality of a task is specified by properties of its state-transition, <*a*, *b*>, where *b* is the state before the task, and *a* is the state after invoking the

TABLE 1
Requirements Classification

| facets | content | uncertainty | vagueness | competence |
|---|---|---|---|---|
| terms | functional | uncertain | vague | rigid |
| | nonfunctional | certain | imprecise | soft |
| | | | precise | |

task. The functional requirement of a task can thus be specified using a pair <*precondition*, *postcondition*>. In the traditional approach to software specification, the precondition and the postcondition describe properties that should be held by the states *b* and *a*. These conditions specify a rigid functionality because whether they are satisfied by a state is a black-and-white situation. Rigid functional requirements can be formally defined as follows:

DEFINITION 1 (Rigid Functional Requirement). *A rigid functional requirement of a task T is a pair of formula* $<\varphi_1, \varphi_2>$ *where* $\varphi_1$ *is a precondition and* $\varphi_2$ *is a postcondition, such that for every* $<b, a> \in T$,

$$hold(\varphi_1, b) \Rightarrow hold(\varphi_2, a),$$

*where "*$hold(\varphi, b)$*" is a function that returns either 1 (true) or 0 (false) in a state a, and* $\Rightarrow$ *denotes logic implication operator.*

However, as was mentioned earlier, imprecision often is inevitable in the functional requirements of complex software systems. We propose the use of soft functional requirements to directly express requirements that are elastic in nature. A soft functional requirement describes state properties that can be satisfied to a degree. By generalizing the definition of rigid functional requirements, we arrive at the following formal definition of soft functional requirements.

DEFINITION 2 (Soft Functional Requirement). *A soft functional requirement of a task T is a pair of formula* $<\varphi_1, \varphi_2>$ *where* $\varphi_1$ *is T's precondition and* $\varphi_2$ *is T's postcondition, such that for every* $<b, a> \in T$,

$$fhold(\varphi_1, b) \overset{f}{\Rightarrow} fhold(\varphi_2, a),$$

*where "*$fhold(\varphi, b)$*" is a function that returns the degree to which a formula* $\varphi$ *is true in state s, and*

$$\overset{f}{\Rightarrow}$$

*denotes implication operator in fuzzy logic.*

The function *fhold* is a generalization of the *hold* predicate in situation calculus [28], which states properties that are true in a given state. Note that a soft functional requirement degenerates to a rigid functional requirement when $fhold(\varphi, s)$ returns either 1 (true) or 0 (false). Therefore, a soft functional requirement is a generalization of a rigid one.

The soft conditions can be represented using the notion of *canonical form* in Zadeh's test-score semantics [45]. A basic idea underlies test-score semantics is that a proposition *p* in a natural language may be viewed as a collection of *elastic constraints*, $C_1, \ldots, C_k$, which restricts the values of a collection of variables $X = (X_1, \ldots, X_n)$. In fuzzy logic, this is accomplished by representing *p* in the canonical form

$$p \rightarrow X \text{ is } A$$

in which *A* is a fuzzy predicate or, equivalently, an *n*-ary fuzzy relation in *U*, where $U = U_1 \times U_2 \times \ldots \times U_n$ and $U_i$, $i = 1, \ldots, n$, is the domain of $X_i$. The canonical form of *p* implies that the possibility distribution of *X* is equal to *A* $\pi_X = A$ which in turns implies that $Poss\{X = u\} = \mu_A(u)$, $u \in U$, where $\mu_A$ is the membership function of *A* and $Poss\{X = u\}$ is the possibility that *X* takes *u* as its value [44]. It is in this sense that *A*, acting as an elastic constraint on *X*, restricts the possible values which *X* can take in *U*.

DEFINITION 3. *Let p be a proposition in its canonical form, X is A, X is a state variable in state s, and* $u_i$ *is the value of X in s. Then*

$$fhold(p, s) = \mu_A(u_i).$$

This definition is useful for deriving conflicting and cooperative degrees for soft requirements.

## 4 ANALYZING SOFT REQUIREMENTS

Generally, there are three steps involved in performing the trade-off analysis for soft requirements:

1) to examine the conflicting and cooperative degrees for any two individual soft requirements;
2) to identify the relationships between soft requirements based on the conflicting and cooperative degrees; and
3) to aggregate soft requirements for a compromise overall requirement.

### 4.1 Defining Conflicting and Cooperative Degrees

Intuitively, two requirements are conflicting with each other if an increase in the degree to which one requirement is satisfied often decreases the degree to which another requirement is satisfied, that is, the *fhold* decreases between the two after states (called a conflicting after state pair). On the other hand, two requirements are said to cooperate with each other if an increase (or decrease) in the degree to which one requirement is satisfied often increases (or decreases) the degree to which another requirement is satisfied, that is, the *fhold* increases (or decreases) between the two after states (called a cooperative after state pair). Note that the third possibility is that the *fhold* remains unchanged between the two after states, which is called an irrelevant after state pair. Similar ideas about the definitions of conflict and cooperation can be found in [5]. We formally define conflicting, cooperative, and irrelevant after state pairs below.

DEFINITION 4 (Conflicting, Cooperative, and Irrelevant After State Pairs). *Assume that* $R_1 = <\varphi_{11}, \varphi_{12}>$, $R_2 = <\varphi_{21}, \varphi_{22}>$ *are two requirements, and that* $A_b$ *is a set of common after state pairs w.r.t. a given before state b.*

A set of conflicting after state pairs, denoted as $CF_b$, is defined as

$$\{(a_i, a_j) \mid (fhold(\varphi_{12}, a_i) - fhold(\varphi_{12}, a_j)) \times (fhold(\varphi_{22}, a_i) - fhold(\varphi_{22}, a_j)) < 0, \text{ and } a_i, a_j \in A_b\}.$$

A set of cooperative after state pairs, denoted as $CP_b$, is defined as

$$\{(a_i, a_j) \mid (fhold(\varphi_{12}, a_i) - fhold(\varphi_{12}, a_j)) \times (fhold(\varphi_{22}, a_i) - fhold(\varphi_{22}, a_j)) > 0, \text{ and } a_i, a_j \in A_b\}.$$

A set of irrelevant after state pairs, denoted as $IR_b$, is defined as

$$\{(a_i, a_j) \mid (fhold(\varphi_{12}, a_i) - fhold(\varphi_{12}, a_j)) \times (fhold(\varphi_{22}, a_i) - fhold(\varphi_{22}, a_j)) = 0, \textit{ and } a_i, a_j \in A_b\}.$$

Hence, a set of common after state pairs, $A_b$ can be divided into three subset: conflicting, cooperative, and irrelevant, in such a way that $A_b = CF_b \cup CP_b \cup IR_b$ and $CF_b \cap CP_b \cap IR_b = \emptyset$. The formal definitions of conflicting and cooperative degrees w.r.t. a given before state are described below.

DEFINITION 5 (Conflicting and Cooperative Degrees). *Assume that $R_1 = <\varphi_{11}, \varphi_{12}>$, $R_2 = <\varphi_{21}, \varphi_{22}>$ are two requirements, $A_b$ is a set of common after state pairs w.r.t. a given before state b, and that $CF_b$ and $CP_b$ denote conflicting and cooperative after state pairs w.r.t. b, respectively.*

The conflicting degree between two requirements, $R_1$ and $R_2$, w.r.t. $b$, is defined as:

$$cf_b(R_1, R_2) = \frac{\sum_{a_i, a_j \in CF_b} (|fhold(\varphi_{12}, a_i) - fhold(\varphi_{12}, a_j)| + |fhold(\varphi_{22}, a_i) - fhold(\varphi_{22}, a_j)|)}{\sum_{a_h, a_k \in A_b} (|fhold(\varphi_{12}, a_h) - fhold(\varphi_{12}, a_k)| + |fhold(\varphi_{22}, a_h) - fhold(\varphi_{22}, a_k)|)}.$$

The cooperative degree between two requirements, $R_1$ and $R_2$, w.r.t, $b$, is defined as:

$$cp_b(R_1, R_2) = \frac{\sum_{a_i, a_j \in CP_b} (|fhold(\varphi_{12}, a_i) - fhold(\varphi_{12}, a_j)| + |fhold(\varphi_{22}, a_i) - fhold(\varphi_{22}, a_j)|)}{\sum_{a_h, a_k \in A_b} (|fhold(\varphi_{12}, a_h) - fhold(\varphi_{12}, a_k)| + |fhold(\varphi_{22}, a_h) - fhold(\varphi_{22}, a_k)|)}.$$

Having defined the conflicting and cooperative degrees, it is necessary to further define the average conflicting and cooperative degrees due to the fact that a requirement may have many before and after states.

DEFINITION 6 (Average Conflicting and Cooperative Degrees). *Assume that $R_1 = <\varphi_{11}, \varphi_{12}>$, $R_2 = <\varphi_{21}, \varphi_{22}>$ are two requirements, $A_b$ is a set of common after state pairs w.r.t. a given before state b, $B_{R_1}$ and $B_{R_2}$ are sets of all possible before states of $R_1$ and $R_2$.* The average conflicting degree between two requirements, $R_1$ and $R_2$, w.r.t. $b$, is defined as:

$$cf(R_1, R_2) = \frac{\sum_{b \in B_{R_1} \cap B_{R_2}} cf_b(R_1, R_2)}{\left\| B_{R_1} \cap B_{R_2} \right\|}.$$

The average cooperative degree between two requirements, $R_1$ and $R_2$, w.r.t. $b$, is defined as:

$$cp(R_1, R_2) = \frac{\sum_{b \in B_{R_1} \cap B_{R_2}} cp_b(R_1, R_2)}{\left\| B_{R_1} \cap B_{R_2} \right\|}.$$

$\left\| B_{R_1} \cap B_{R_2} \right\|$ is the cardinality of intersection between $B_{R_1}$ *and* $B_{R_2}$.

We have further distinguished positive cooperative degrees from negative ones. A positive cooperative degree means that if an increase in the degree to which one requirement is satisfied often increases the degree to which another requirement is satisfied. Otherwise, it is called a negative cooperative degree.

## 4.2 Relationships Between Soft Requirements

The relationships among soft requirements are crucial for adequately interpreting the intended meaning of a compromise overall requirement, because they reflect the structure of interaction among the soft requirements. Together with information about the criticality of soft requirements which usually represents users' pros and cons of the requirements, the relationships among soft requirements can serve as a guideline for requirements aggregations [17], [5].

In the proposed framework, the relationship between any two soft requirements, say $R_i$ and $R_j$, can be classified under five categories: independent, irrelevant, counterbalance, conflicting, and cooperative. $R_i$ and $R_j$ are said to be *independent* if there is no common after state shared by the requirements, that is, $A_{R_1} \cap A_{R_2} = \emptyset$. Two requirements are irrelevant if there is no conflicting and cooperative relationships between the requirements, that is, all after state pairs are irrelevant pairs (i.e., $cp = cf = 0$). In the case of counterbalance relationships, both the conflicting and cooperative relationships co-exist and their degrees are equivalent (i.e., cp = cf).

To further refine conflicting and cooperative relationships, we have identified three subcategories: strong, moderate, and weak. A relationship is said to be conflicting if the conflicting degree between $R_i$ and $R_j$ is greater than the cooperative degree. On the other hand, if the cooperative degree is greater than the conflicting degree, then $R_i$ cooperates with $R_j$. In the case that there is only conflicting after state pairs (i.e., $cp = 0$), $R_i$ is *strongly conflicting* with $R_j$. Similarly, if there is only cooperative after state pairs (i.e., $cf = 0$), two requirements are *strongly cooperative* with each other.

The co-existence of conflicting, cooperative and irrelevant after state pairs (i.e., $cp + cf < 1$) usually drops either the conflicting degree or the cooperative degree further compared with the existence of only conflicting and cooperative after state pairs (i.e., $cp + cf = 1$). The former is called *weak conflicting* or *weak cooperative*, while the later is called *moderate*. The relationships between soft requirements are summarized in Table 2.

TABLE 2
RELATIONSHIPS BETWEEN SOFT REQUIREMENTS

| relationship / condition | conflicting | | | irrelevant | counterbalance | cooperative | | | independent |
|---|---|---|---|---|---|---|---|---|---|
| | strong | moderate | weak | | | strong | moderate | weak | |
| cf-cp | cp=0 | cf-cp>0 | cf-cp>0 | cf=cp=0 | cf=cp | cf=0 | cf-cp<0 | cf-cp<0 | $A_{R1} \cap A_{R2} = \phi$ |
| cp+cf | cf=1 | cf+cp=1 | cf+cp<1 | cf+cp=0 | cf+cp≤1 | cp=1 | cp+cf=1 | cf+cp<1 | |
| cp: cooperative degree cf: conflicting degree | | | | | | | | | |

## 4.3 Hierarchical Aggregation of Requirements

Having decomposed user's requirements into different individual ones, it is then necessary to achieve some level of integration between those individual requirements. In general, there are two issues needed to be addressed in using aggregation operators for the integration of individual requirements:

- The variety of aggregation operators could make it difficult to determine which one to use in a specific application [22]. Zimmermann [46] has outlined eight general criteria: axiomatic strength, empirical fit, adaptability, numerical efficiency, compensation, range of compensation, aggregating behavior, and required scale level of membership functions. Furthermore, Yen et al. [43] have summarized several criteria for selecting an appropriate aggregation operator in the context of requirements engineering:
    1) intended relationship between requirements,
    2) feasible combined requirements,
    3) higher satisfiable realization, and
    4) requirements with criticalities.

  However, most of the existing approaches (including our earlier work) only considered one or two of the criteria in the analysis, for example, requirements with criticalities in [41], intended relationships between requirements in [43], or between objectives in [17].
- The averaging operator is symmetrical, monotonic, commutative and idempotency, but the property of associativity is not available. The lack of associativity with respect to any averaging operator raises some important issues of how to extend the operator. Several researchers such as Yager and Filev [40] and Cutello and Motero [7] have proposed different imperatives in holding the definition together as elements are added to an aggregation.

To alleviate the above-mentioned problems, we propose an extension of the hierarchical aggregation structure advocated in [39], where requirements in each disjunct and conjunct are expanded to form a requirements hierarchy. In the proposed framework, we not only explore all the criteria summarized by Yen et al. in [43], but also relax the assumption by taking the consideration that requirements from users are usually described using either *and*, or *or* natural language connectives, or both. The steps in establishing a hierarchical structure for requirements aggregation are discussed below.

### 4.3.1 Convert Requirements into DNF

To take these connectives into account, we proposed the use of disjunctive normal form (DNF) to obtain a uniform representation of the requirements. Requirements in DNF form can then be arranged based on an extension of the notion of the hierarchical aggregation structure advocated in [47], [39], [6], [7], where requirements in each disjunct and conjunct are expanded to form a requirements hierarchy. A requirements hierarchy can be established based on the notion of the requirements criticality and the positive cooperative degree. In fact, the requirements may carry different weights reflecting their degrees of criticality, where a weight is a nonnegative real number. We have adopted Saaty's pairwise comparison approach to the assignment of weights to requirements [37]. That is, the relative weights of each requirement pair are used to form a reciprocal matrix, and the absolute weight of each requirement is obtained from the normalized eigenvector using eigenvalue method. A requirement $R$ can thus be represented by a triple: $<w_R, \mu_A(x), R>$, where $w_R$ denotes the criticality associated with the requirement $R$, and $\mu_A(x)$ is based on Definition 3, $x \in X$.

### 4.3.2 Establish A Requirements Hierarchy

Assume that requirements are either connected by the conjunction or by the disjunction connective and that the hierarchy is from level 0 (the top level) to level $n$ (see Algorithm 1).

This step is important in the sense that the ordering established through the hierarchy helps to alleviate the associativity problem inherited in averaging operators, namely, a unique ordered list can thus be obtained.

```
Algorithm 1: (Establish a Requirements Hierarchy)
  1) Top-down:
       a) Sort the criticalities for all requirements.
       b) Arrange requirements from top down in a descending order of the criticalities. Requirements with the
          same criticality will be placed at the same level.
  2) Top-level requirements:
       a) If there is only one requirement with the highest criticality, place it on the top of the hierarchy.
       b) Else if there are more than one requirement with the highest criticality,
            i)   Compute the total cooperative degree for each requirement with the rest of the requirements whose
                 criticality is the same.
            ii)  Sort the total cooperative degrees computed in the previous step.
            iii) Arrange requirements from left to right on the top in a descending order of the total cooperative
                 degrees.
            iv)  Add a virtual requirement on the top of those requirements.
  3) Grouping requirements (between two adjacent levels):
       a) For requirements (other than requirements at the top level) with the same criticality, compute all the co-
          operative degrees for each requirement at level $i$ with every requirement at level $i-1$.
       b) Given a requirement, $R_h$, at level $i$, for every requirements, $R_k$, at level $i-1$, sort the cooperative degree of
          $R_h$ and $R_k$, obtained from the previous step. Group the requirement whose cooperative degree is the high-
          est under $R_h$.
       c) Continue the previous step until all the requirements at level $i$ have been considered.
  4) Left-right (for each group):
       a) Given a requirement, $R_k$, at level $i-1$, for every requirements, $R_h$, at level $i$, sort the cooperative degrees
          of $R_h$ and $R_k$.
       b) Arrange from left to right the requirements at level $i$ in a descending order.
```

### 4.3.3 *Select Aggregation Operators*

To select appropriate aggregation operators, we propose the consideration of operators that can:

1) reflect the intended relationship between requirements,
2) associate the criticality with each requirement,
3) fit the aggregation structure, and
4) incorporate the notion of conflicting and cooperative degrees.

We have chosen *fuzzy and* and *fuzzy or* operators proposed in [39], due to the fact that *fuzzy and* operator can be used within each conjunction, while *fuzzy or* can be applied between each disjunction, and that the *compensation* between aggregated sets can be achieved by incorporating the conflicting and cooperative degrees into the parameters in these two operators, which in turn can reflect the relationships between requirements. In RTA, all the requirements are considered during the aggregation process, and hence no requirement will be excluded out. However, the lower the degree of the criticality of a requirement, the lower the impact of the requirement on the result of the aggregation. Therefore, the exclusive or aggregation is not considered in RTA. In addition, the mathematical structure of these operators is easy and can be handled efficiently [39]. In order to better match our aggregation structure, these two operators are also adopted for criticalities aggregation to reflect different relationships between the requirements. We formally define these two operators below.

DEFINITION 7. (Extended *Fuzzy and*). *Assume two requirements* $<w_{R_1}, \mu_A(x), R_1>$, $<w_{R_2}, \mu_B(x), R_2>$, $\forall x \in X$.

$$<w_{R_1}, \mu_A(x), R_1> \wedge_{\gamma_{and}} <w_{R_2}, \mu_B(x), R_2>$$

*is defined as*

$$<w_{and}(w_{R_1}, w_{R_2}), \mu_{and}(\mu_A(x), \mu_B(x)), R_1 \wedge R_2>$$

*where*

$$\mu_{and}(\mu_A(x), \mu_B(x)) = \gamma_{and}\, min\, \{\mu_A(x), \mu_B(x)\} + \frac{(1-\gamma_{and})(\mu_A(x)+\mu_B(x))}{2},$$

$$w_{and}(w_{R_1}, w_{R_2}) = \gamma_{and}\, min\, \{w_{R_1}, w_{R_2}\} + \frac{(1-\gamma_{and})(w_{R_1}+w_{R_2})}{2},$$

$$\gamma_{and} = (cf - cp + 1)/2 \text{ and } \gamma_{and} \in [0, 1].$$

DEFINITION 8. (Extended *Fuzzy or*). *Assume two requirements* $<w_{R_1}, \mu_A(x), R_1>$, $<w_{R_2}, \mu_B(x), R_2>$, $\forall x \in X$.

$$<w_{R_1}, \mu_A(x), R_1> \vee_{\gamma_{or}} <w_{R_2}, \mu_B(x), R_2>$$

*is defined as*

$$<w_{or}(w_{R_1}, w_{R_2}), \mu_{or}(\mu_A(x), \mu_B(x)), R_1 \vee R_2>$$

*where*

$$\mu_{or}(\mu_A(x), \mu_B(x)) = \gamma_{or}\, max\, \{\mu_A(x), \mu_B(x)\} + \frac{(1-\gamma_{or})(\mu_A(x)+\mu_B(x))}{2},$$

$$w_{or}(w_{R_1}, w_{R_2}) = \gamma_{or}\, max\, \{w_{R_1}, w_{R_2}\} + \frac{(1-\gamma_{or})(w_{R_1}+w_{R_2})}{2}.$$

$$\gamma_{or} = (cp - cf + 1)/2 \text{ and } \gamma_{or} \in [0, 1].$$

In the case that two requirements are connected by *and*, there are two situations:

1) if $\gamma_{and}$ equals to 1 (i.e., requirements are strongly conflicting), the *fuzzy and* operator reduces to *min*, and
2) if $\gamma_{and}$ equals to 0 (i.e., requirements are strongly cooperative), the operator becomes *arithmetic mean*.

If the requirements are connected by *or*, the *fuzzy or* operator yields *max* under the condition that $\gamma_{or}$ equals to 1; whereas, the operator boils down to *arithmetic mean* if $\gamma_{or}$ equals to 0. Although the ranges for the parameters $\gamma_{and}/\gamma_{or}$ in the case of "moderate" and "weak" relationships are similar, a subtle distinction can still be made through the observation of *cf* and *cp*. That is, in the case of "moderate" relationship, *cf* + *cp* = 1; whereas, for "weak" relationship, *cf* + *cp* < 1. This distinction can have some impacts on the computation of $\gamma_{and}$ and $\gamma_{or}$.

$\gamma_{and} = \gamma_{or} = 0.5$ indicates that two requirements are either *irrelevant* or *counterbalance*. Generally, if two requirements are *independent*, there is no need for performing the aggregation. These are summarized in Table 3.

### 4.3.4 Aggregate Requirements

To aggregate requirements in a requirement hierarchy, there are two steps involved:

1) to utilize the breadth first search algorithm to traverse the requirements hierarchy to form an ordered list, and
2) to apply *fuzzy and* or *fuzzy or* operator recursively to the requirements in the list.

Finally, a four-level hierarchical aggregation structure can thus be built (see Fig. 1):

1) Requirements hierarchies built from disjuncts by applying Algorithm 1 are placed at the bottom of the hierarchical structure. Each requirement hierarchy is converted into its ordered list.
2) *Fuzzy and* operator is applied recursively to glue requirements in an ordered list to establish an aggregated requirement, which is placed on the top of the requirements hierarchy.

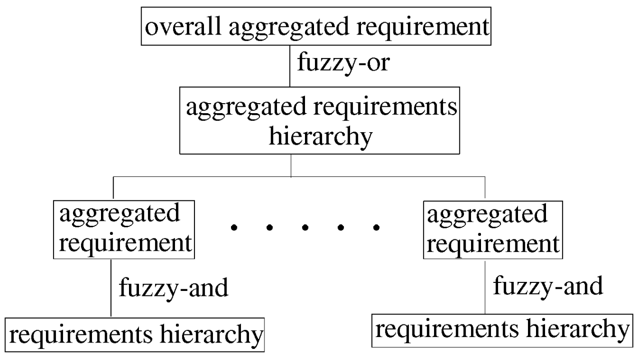


Fig. 1. The extended hierarchical aggregation structure.

3) All the aggregated requirements will be used to build an aggregated requirements hierarchy, in which the aggregated requirement in the hierarchy are in turn combined using *fuzzy or* operator recursively to form an overall aggregated requirement. It is of interest to note that a hierarchy in RTA is established mainly based on the relationships among the requirements and the criticalities of the requirements. Therefore, only the change of relationships and criticalities will result in the change of the hierarchy. Namely, given a new set of relationships and criticalities, RTA will rebuild a new hierarchy instead of reusing the previous hierarchy.

## 5 RELATED WORK

Work in a number of fileds has made its mark on our requirements trade-off analysis framework. Analogies of trade-off analysis based on relationships may be found in multicriteria decision making [5], [16], AI planning [27], [38], and multiperspectives specifications [43], [14], [12], [18], [30].

### 5.1 Multicriteria Decision Making

Relationship analysis approaches in multicriteria decision making [4], [5], [17], [15] focus on an explicit modeling of

TABLE 3
RELATIONSHIPS VS. AGGREGATION OPERATORS

| relation / operator | conflicting | | | irrelevant | counterbalance | cooperative | | |
|---|---|---|---|---|---|---|---|---|
| | strong | moderate | weak | | | strong | moderate | weak |
| $\wedge$ ($\gamma_{and}=(cf-cp+1)/2$) | min | fuzzy-and | | fuzzy-and | | arith mean | fuzzy-and | |
| | $\gamma_{and}=1$ | $1>\gamma_{and}>0.5$ | $1>\gamma_{and}>0.5$ | ($\gamma_{and}=0.5$) | | $\gamma_{and}=0$ | $0.5>\gamma_{and}>0$ | $0.5>\gamma_{and}>0$ |
| $\vee$ ($\gamma_{or}=(cp-cf+1)/2$) | arith mean | fuzzy-or | | fuzzy-or | | max | fuzzy-or | |
| | $\gamma_{or}=0$ | $0.5>\gamma_{or}>0$ | $0.5>\gamma_{or}>0$ | ($\gamma_{or}=0.5$) | | $\gamma_{or}=1$ | $1>\gamma_{or}>0.5$ | $1>\gamma_{or}>0.5$ |

relationships between goals, and on the determination of the final set of decision alternatives according to the relationships. Carlsson and Fuller [5] propose an approach to fuzzy multiple objective programming (FMOP) with interdependency relationships among objectives, which is an extension of Carlsson's MOP [4] to fuzzy logic. Three kinds of relationships have been identified: supportive, conflicting, and independent. The basic idea is to utilize these relationships to modify the membership function of the so called "good solution." Felix [15] and Felix et al. [17] propose an approach, called DMRG (Decision Making Based on Relationship between Goals), to defining a spectrum of relationships based on fuzzy inclusion and fuzzy noninclusion: independent, assist, cooperate, analogous, hinder, compete, trade-off, and unspecified dependent, and to determining the final set of decision alternatives according to the relationships. These approaches are similar to ours in two aspects: the problems of modeling the relationships, and the issues of aggregation.

## 5.2 AI Planning

Research in the areas of goals conflict in AI planning tackles issues similar to the trade-off analysis, for example,

- Sycara [38] provides a negotiation method, that is called PERSUADER, to find a compromise acceptable to all agents under the situations that planning goals are ill-specified, subgoals cannot be enumerated and the utilities with the subgoals are not precisely known. The negotiation is performed through proposal and modification of goal relaxation. There are two ways of reacting to negative feedback through negotiation: changing the rejecting agent's evaluation of the plan through persuasive argumentation, and modifying/repairing the plan so that it will be more acceptable. The main difference between PERSUADER and our approach in the trade-off analysis is that in aggregating individual requirements, requirements are compensated to each other based on their relationships in our approach; whereas, PERSUADER needs to modify user's utility if no solution can be obtained.
- Luria [27] proposes a commonsense planner called KIP (Knowledge Intensive Planner) which is developed for the Unix Consultant system. KIP uses goals conflict concerns to deal with potential goal conflicts. Luria classifies goal conflict concerns into six types: default concerns, violated-default concerns, intended effects concerns, unintended effects concerns, expressed goal conflict concerns and effect goal conflict concerns. After KIP detects the goals of the user, it selects a potential plan. KIP then checks for violated defaults goal conflict concern. KIP next proceeds the intended effect of the selected plan about user goal. Finally, KIP evaluates the degree of those concerns. If the degree of concern is low, KIP disregards the concern. If the degree of concern is high, KIP elevates the concern to a source of plan failure and pass it to goal conflict resolution. Conflict resolution may occur by either modifying the plan, or choosing a new plan.

## 5.3 Multiperspectives Specifications

Work on multiple perspectives has been investigated along several directions. Feather [14] suggests using many parallel evolutionary transformations for constructing specifications, which may then be merged by replaying them sequently. Finkelstein and Fuks [18] develop a framework to support the construction of formal specifications and reasoning about the process of specifications from multiple viewpoints. Their model has two parts: an underlying viewpoint architecture and a dialogue scheme, which combines the dialogue logics with cooperation and negotiation approaches. Dialogues are used to perform viewpoints negotiation, to establish responsibilities of participants, and to construct an overall specification in a cooperative manner among the participants. The viewpoint architecture includes viewpoint, commitment store, working area, event store and dialogue kernel. A viewpoint is a logical participant in the dialogue. A physical participant in a dialogue may present many logical viewpoints. Each viewpoint has a commitment store which holds it's commitments within the dialogue. The dialogue scheme is presented in terms of three constructs: acts, events, and commitments.

Easterbrook [12] proposes a framework for representing conflicting viewpoints in a domain model. A viewpoint in his framework is a self-consistent description of an area of knowledge representing the context in which a role is performed. In evolving viewpoints, a new viewpoint will need to be split if it causes inconsistency. The new viewpoint and its negation are placed in different descendants of the original viewpoint, so that each remains self-consistent individually. The detection of conflict might be based on detection of logical inconsistencies. Thus, a hierarchy of viewpoints is established as the elicitation proceeds. The inheritance structure implies that the higher an item in the hierarchy, the more widely agreed it is. One of the aims of using viewpoints is to reduce the need for consistency checks. This approach allows many viewpoints to be combined into a single domain model without necessary resolving conflicts between them. Conflicts are treated as an important part of the domain, and are to be represented and understood.

Robinson [33] and Robinson and Fickas [34] propose an approach, called Oz, to requirements negotiation. There are three steps involved in Oz: conflict detection, resolution generation, and resolution selection. The conflict detector of Oz does a pairwise comparison across all specifications. It does so by matching up design goals from perspectives and by comparing their plans. The specifications and conflicts will be passed to conflict resolver which will provide analytic compromise and heuristic compensation and dissolution for each conflict. Compensation is to add similar but conflict free requirements to negotiations, while, dissolution is to replace conflicting items potentially less contentious items. Finally, the resolver will provide guidance for search control by choosing intermediate alternatives and automated negotiation methods. Each method can be applied in any sequence to derive resolutions. The nonconflicting specifications are jointed into a single specification by merger of Oz.

Furthermore, our notion of conflicting and cooperative degrees can be interpreted as taking the distance-wise view between any two individual requirements; whereas, Yen et al. [43] addressed the same issue from the probability aspects of conflicting and cooperative relationships, that is, computing the ratio between the total number of after state pairs and those of conflicting or cooperative pairs. No irrelevant pair is considered.

## 6 An Example: ISPBEX Expert System

To convey a sense of how RTA performs its trade-off analysis, let's consider the expert system ISPBEX [13] that helps Forest Service personnel to make decisions about the Southern Pine Beetle spots by providing treatment recommendations. The requirements for the treatment recommendation generated by the system represented using their canonical forms are described below:

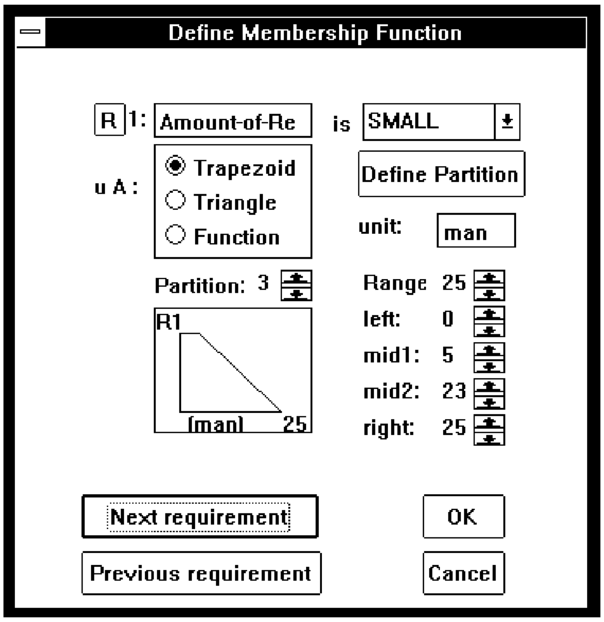
$R_1$: Amount-of-Resource (Treatment Recommendation) is SMALL;
$R_2$: Cost-of-implementation (Treatment Recommendation) is LOW;
$R_3$: Responsive-Time (Treatment Recommendation) is SHORT;
$R_4$: Negative-impact (endangered-species (Treatment Recommendation)) is LOW;
$R_5$: Profit (Treatment Recommendation) is HIGH;

where SMALL, LOW, SHORT, and HIGH are fuzzy subsets and serve as elastic constraints on a treatment recommendation.

The input parameters for ISPBEX are treatment month, type of forest (general forest, wilderness), colony-condition (active, not active), breeding season (spring, summer, fall, winter), priority (A, B, C, D), type of trees (loblobby or short leaf pine), characteristics of trees (age, size), and growth distance (in degrees). The output is a proper treatment recommendation. Each situation represents a set of given conditions corresponding to the input parameters. For our example, we consider the situation under which the treatment month is March, the type of forest is wilderness, SPB spot is active, endangered species are breeding, priority is high, SPB is going to impact the endangered species colony within 60 days, and so on. Four main recommendations are possible, including Cut and Leave (C/L), Cut and Remove (C/R), Pile and Burn (P/B), and Monitor (Mon). A before state corresponds to the input of situation, while an after state corresponds to a possible solution (recommendation) given for the situation. The criticalities of requirements are ($R_1$, 0.3), ($R_2$, 0.1), ($R_3$, 0.1), ($R_4$, 0.2), *and* ($R_5$, 0.3), which are derived by Saaty's pairwise comparison approach.

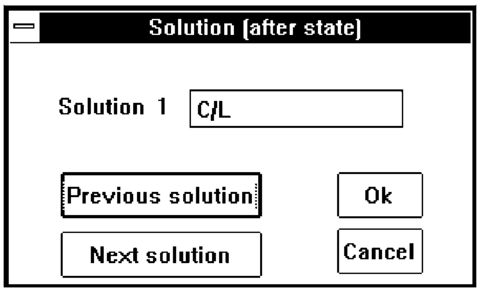
To begin with, in Fig. 2, we first input requirements in their canonical forms, and define their corresponding membership functions by selecting a type of function (e.g., trapezoid, triangle and user defined), the number of partitions, and their related fuzzy terms.

The next step is to describe the possible solutions corresponding to the requirements (see Fig. 3). Experts will then be requested to fill in the value for each solution w.r.t every



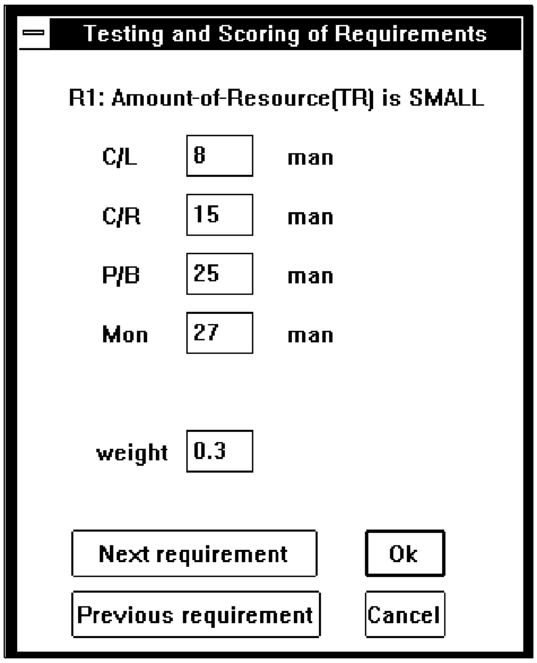
| | |
|---|---|
| R 1: | Amount-of-Re |
| is | SMALL |
| u A: | ● Trapezoid ○ Triangle ○ Function |
| unit: | man |
| Partition: | 3 |
| Range | 25 |
| left: | 0 |
| mid1: | 5 |
| mid2: | 23 |
| right: | 25 |

Fig. 2. Define membership functions.



Solution 1 C/L

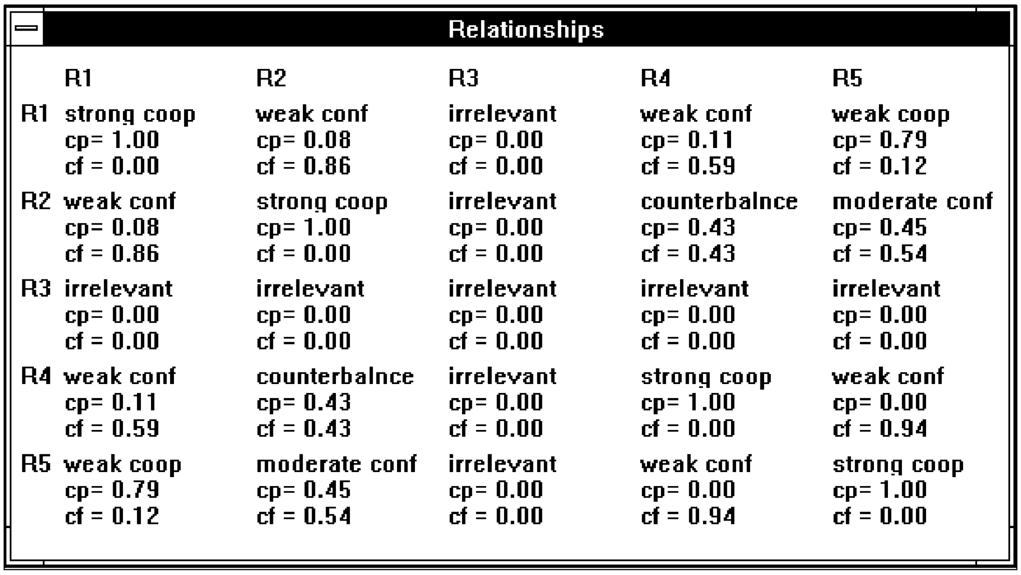
Fig. 3. Solutions description.



R1: Amount-of-Resource(TR) is SMALL

| | | |
|---|---|---|
| C/L | 8 | man |
| C/R | 15 | man |
| P/B | 25 | man |
| Mon | 27 | man |
| weight | 0.3 | |

Fig. 4. Values assignment.


Relationships

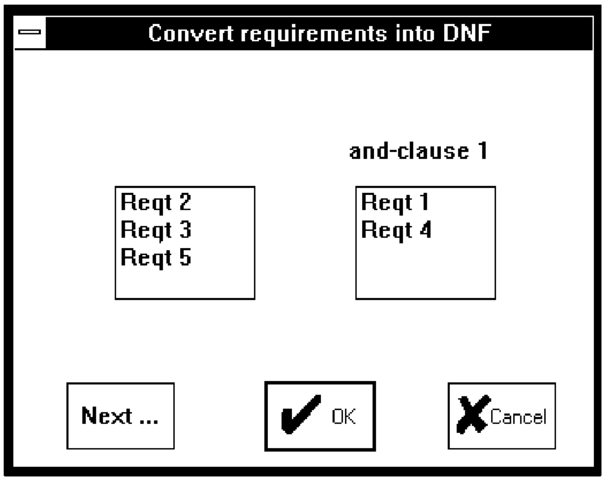
| | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| R1 | strong coop cp= 1.00 cf = 0.00 | weak conf cp= 0.08 cf = 0.86 | irrelevant cp= 0.00 cf = 0.00 | weak conf cp= 0.11 cf = 0.59 | weak coop cp= 0.79 cf = 0.12 |
| R2 | weak conf cp= 0.08 cf = 0.86 | strong coop cp= 1.00 cf = 0.00 | irrelevant cp= 0.00 cf = 0.00 | counterbalnce cp= 0.43 cf = 0.43 | moderate conf cp= 0.45 cf = 0.54 |
| R3 | irrelevant cp= 0.00 cf = 0.00 | irrelevant cp= 0.00 cf = 0.00 | irrelevant cp= 0.00 cf = 0.00 | irrelevant cp= 0.00 cf = 0.00 | irrelevant cp= 0.00 cf = 0.00 |
| R4 | weak conf cp= 0.11 cf = 0.59 | counterbalnce cp= 0.43 cf = 0.43 | irrelevant cp= 0.00 cf = 0.00 | strong coop cp= 1.00 cf = 0.00 | weak conf cp= 0.00 cf = 0.94 |
| R5 | weak coop cp= 0.79 cf = 0.12 | moderate conf cp= 0.45 cf = 0.54 | irrelevant cp= 0.00 cf = 0.00 | weak conf cp= 0.00 cf = 0.94 | strong coop cp= 1.00 cf = 0.00 |

Fig. 5. Display relationships.

requirement. The criticalities (i.e., weights) of requirements are also derived. For example (see Fig. 4), the expert needs to assign the amount-of-resource in terms of man/month required for each solution such as C/L, C/R, etc. After the inputs, the conflicting and cooperative degrees will then be calculated, which are used to derive the relationships among the requirements.

Fig. 5 shows the relationships for our example. It is obvious that $R_1$ and $R_2$ are weak conflicting (i.e., cp < cf and cp + cf < 1), and $R_1$ and $R_5$ are weak cooperative because cf < cp and cp + cf < 1. Note that $R_2$ and $R_4$ are counterbalance due to cp = cf = 0.43; meanwhile $R_1$ and $R_3$ are irrelevant because cp = cf = 0.

Finally, to aggregate those requirements to form an overall requirement, we first convert those requirements into their DNF forms and build up the extended hierarchical aggregation structure. Based on the discussion in Section 4, the *fuzzy and* operator is applied to aggregate the requirements connected by conjunction, and the *fuzzy or* operator to aggregate the requirements connected by disjunction. The criticalities of requirements are also aggregated using the *fuzzy and* or *fuzzy or* operator. In the example (see Fig. 6), we assume that $R_1$ and $R_4$ are connected by the conjunctive operator, and $R_2$, $R_3$, and $R_5$ are connected by the conjunctive operator. The two aggregated sets are connected by the disjunctive operator.


Convert requirements into DNF

and-clause 1

| | |
|---|---|
| Reqt 2<br>Reqt 3<br>Reqt 5 | Reqt 1<br>Reqt 4 |

Fig. 6. Convert requirements into DNF.

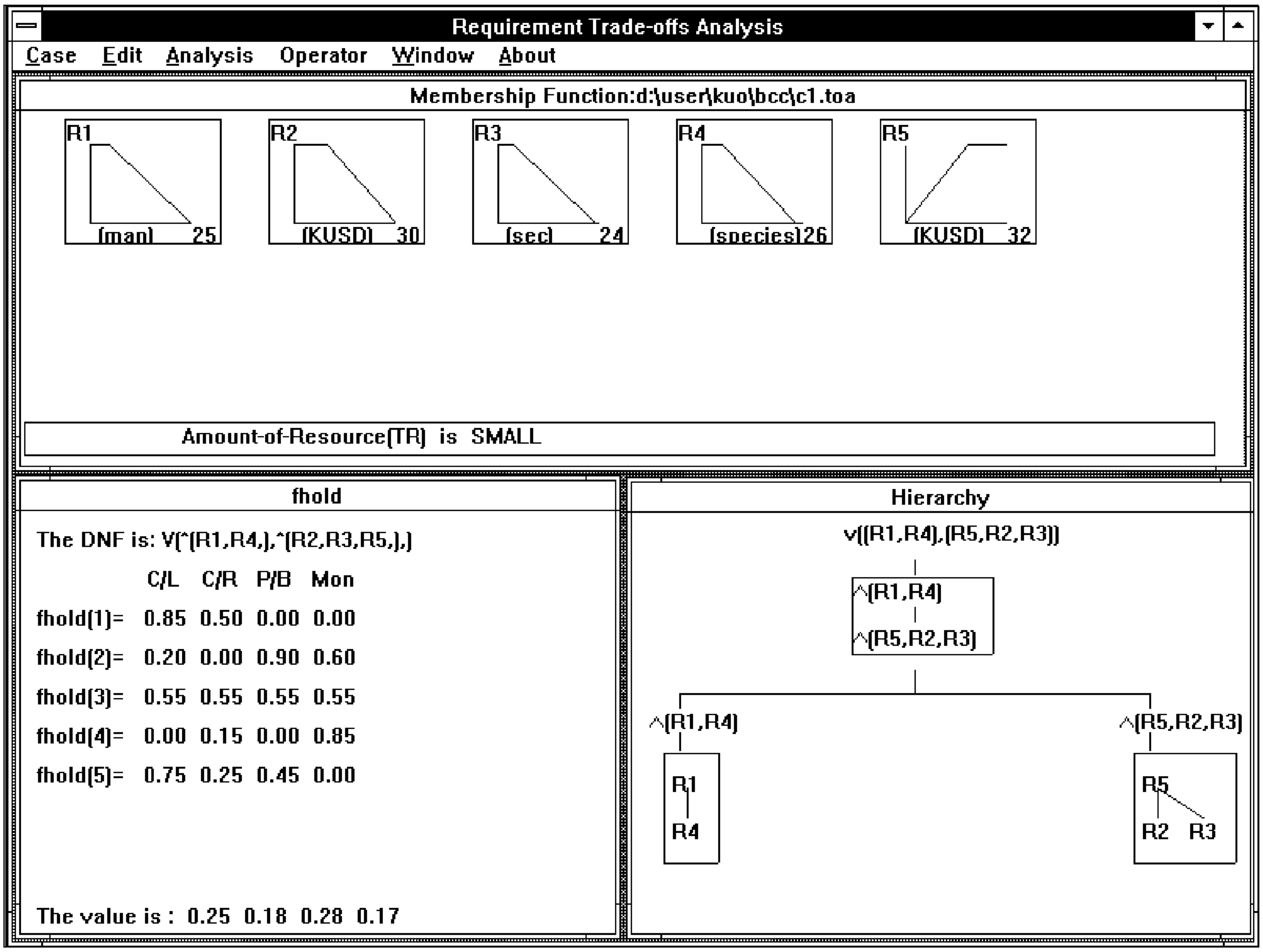
Fig. 7 illustrates the result of the trade-off analysis, including membership functions for each requirement, the requirements hierarchy, and the final evaluation of the degree of satisfaction w.r.t. each solution. The degrees of satisfaction of the overall requirement for the four solutions are, respectively, (0.25, 0.18, 0.28, 0.17). Therefore, RTA will recommend the selection of the treatment P/B in our example.

## 7 Conclusion

In this paper, a Requirements Trade-off Analysis technique (RTA) is proposed to formally model vague requirements. Conflicting and cooperative degrees between any two individual requirements are first formulated. Relationships between individual requirements are identified based upon their conflicting and cooperative degrees. Requirements are converted into the disjunctive normal form to obtain a uniform representation of the requirements, and then arranged into an extended hierarchical aggregation structure, where requirements in each disjunction are expanded to form a requirements hierarchy. A requirements hierarchy is established based on the notion of criticality and the cooperative degree. Parameterized aggregation operators, *fuzzy and/or*, are selected to combine individual requirements. A compromise overall requirement can be obtained through the aggregation of individual requirements based on the hierarchical requirements.

RTA offers several important benefits:

- RTA provides a framework for formally analyzing and modeling conflicts between requirements, and for users to better understand their requirements.



```
Amount-of-Resource(TR) is SMALL

The DNF is: V(^(R1,R4,),^(R2,R3,R5,),)

            C/L   C/R   P/B   Mon
fhold(1)=   0.85  0.50  0.00  0.00
fhold(2)=   0.20  0.00  0.90  0.60
fhold(3)=   0.55  0.55  0.55  0.55
fhold(4)=   0.00  0.15  0.00  0.85
fhold(5)=   0.75  0.25  0.45  0.00

The value is :  0.25  0.18  0.28  0.17
```

Fig. 7. The result of requirements trade-off analysis.

- Incorporating parameterized fuzzy and/or operators in the extended hierarchical aggregation structure helps in reflecting the intended relationships between requirements.
- The extended hierarchical structure makes easy the obtaining of a compromise overall requirement.

RTA can be considered as one of the attempts towards the formulation of what we called *Trade-Off Requirements Engineering*. In order to further extend this research area, our future work consists of several tasks:

1) to investigate the possibility of integrating RTA with the traditional requirements analysis and design approaches to complement each other, and
2) to formally model nonfunctional requirements based on RTA.

## ACKNOWLEDGMENTS

We thank Professor W.T. Huang and Professor John Yen for their invaluable comments on an early draft of this paper. This research is supported by the National Science Council of Taiwan, Republic of China, under Grants No. NSC84-2213-E-008-016 and No. NSC85-2213-E-008-007.

## REFERENCES

[1] R. Balzer, N. Goldman, and D. Wile, “Informality in Program Specifications,” *IEEE Trans. Software Eng.*, vol. 4, no. 2, pp. 94–103, 1978.

[2] B.I. Blum, “Representing Open Requirements with a Fragment-Based Specification,” *IEEE Trans. Software Eng.*, vol. 23, no. 3, pp. 724–736, 1993.

[3] A. Borgida, S. Greenspan, and J. Mylopoulos, “Knowledge Representation as the Basis for Requirements Specification,” *Computer*, pp. 82–91, Apr. 1985.

[4] C. Carlsson, “On Optimization with Interdependent Multiple Criteria,” R. Lowen and M. Roubens, eds., *Fuzzy Logic: State of the Art*, pp. 287–300, Kluwer, the Netherlands, 1993.

[5] C. Carlsson and R. Fuller, “Interdependence in Fuzzy Multiple Objective Programming,” *Fuzzy Sets and Systems*, vol. 65, pp. 19–29, 1994.

[6] V. Cutello and J. Montero, “A Characterization of Rational Amalgamation Operations,” *Int’l J. Approximate Reasoning*, vol. 8, pp. 325–344, 1993.

[7] V. Cutello and J. Montero, “The Associativity Problem for Owa Operators,” *Proc. Sixth Int’l Fuzzy Systems Assn. World Congress*, vol. I, pp. 149–152, 1995.

[8] A. Dardenne, A. van Lamsweerde, and S. Fickas, “Goal-Directed Requirements Acquisition,” *Science of Computer Programming*, vol. 20, pp. 3–50, 1993.

[9] *Software Requirements: Analysis and Design*, A.M. Davis, ed., Englewood Cliffs, N.J.: Prentice Hall, 1990.

[10] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum, New York, 1988.

[11] D. Dubois and H. Prade, "An Introduction to Possibilistic and Fuzzy Logics," G. Shafer and J. Pearl, eds., *Readings in Uncertain Reasoning*, pp. 742–761, Morgan Kaufmann, San Mateo, Calif., 1990.

[12] S. Easterbrook, "Domain Modelling with Hierarchies of Alternative Viewpoints," *Proc. IEEE Int'l Symp. Requirements Eng.*, pp. 65–72, 1993.

[13] R.O. Flamm et al., "The Integrated Southern Pine Beetle Expert Systems," *Expert Systems With Applications*, vol. 2, pp. 97–105, 1991.

[14] M.S. Feather, "Constructing Specifications by Combining Parallel Elaboration," *IEEE Trans. Software Eng.*, vol. 15, no. 2, pp. 198–208, 1989.

[15] R. Felix, "Relationships Between Goals in Multiple Attribute Decision Making," *Fuzzy Sets and Systems*, vol. 67, pp. 47–52, 1994.

[16] R. Felix, "Fuzzy Decision Making Based on Relationships Between Goals Compared with the Analytic Hierarchy Process," *Proc. Sixth Int'l Fuzzy Systems Assn. World Congress*, pp. 253–256, 1995.

[17] R. Felix, S. Reddig, and A. Adelhof, "Multiple Attribute Decision Making Based on Fuzzy Relationships Between Objectives and Its Application in Metal Forming," *Proc. Second IEEE Int'l Conf. Fuzzy Systems*, pp. 378–383, 1993.

[18] A. Finkelstein and H. Fuks, "Multi-Party Specification," *Proc. Int'l Workshop Software Specifications and Design*, pp. 185–195, 1989.

[19] B. N. Freeman-Benson, J. Maloney, and A. Borning, "An Incremental Constraint Solver," *Comm. ACM*, vol. 33, no. 1, pp. 54–63, Jan. 1990.

[20] R.G. Babb III, R. Kieburtz, K. Orr, A. Mili, S. Gearhart, and N. Martin, "Workshop on Models and Languages for Software Specification and Design," *Computer*, pp. 103–108, Mar. 1985.

[21] W.L. Johnson, "Knowledge-Based Software Engineering," A. Kent and J.G. Williams, eds., *Encyclopedia of Computer Science and Technology*, vol. 31, pp. 173–225, Marcel Dekker, New York, 1994.

[22] U. Kaymak and H.R. van Nauta Lemke, "Selecting an Aggregation Operator for Fuzzy Decision Making," *Proc. Third IEEE Int'l Conf. Fuzzy Systems*, pp. 1,418–1,422, IEEE CS Press, 1994.

[23] G.J. Klir, "Where Do We Stand on Measures of Uncertainty, Ambiguity, Fuzziness, and the Like?" *Fuzzy Sets and Systems*, vol. 24, pp. 141–160, 1987.

[24] J. Lee, J.Y. Kuo, and W.T. Huang, "Classifying, Analyzing, and Representing Informal Requirements," *Proc. Sixth Int'l Fuzzy Systems Assn. World Congress*, vol. I, pp. 645–648, July 1995.

[25] J. Lee, L.F. Lai, and W.T. Huang, "Task-Based Specifications Through Conceptual Graphs," *IEEE Expert*, vol. 11, no. 4, pp. 60–70, Aug. 1996.

[26] *Automating Software Design*, M.R. Lowry and R.D. McCartney, eds., AAAI Press, Menlo Park, Calif., 1991.

[27] M. Luria, "Goal Conflict Concerns," *Proc. 12th Int'l Joint Conf. Artificial Intelligence*, pp. 1,025–1,031, 1987.

[28] J. McCarthy and P. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," *Machine Intelligence 4*, pp. 463–502, Edinburgh Univ. Press, Scotland, 1969.

[29] C. Niskier, T. Maibaum, and D. Schwabe, "A Look Through Prisma: Towards Pluralistic Knowledge-Based Environments for Software Specification Acquisition," *Proc. Fifth Int'l Workshop Software Specification and Design*, pp. 128–136, IEEE CS Press, 1989.

[30] B. Nuseibeh, J. Kramer, and A. Finkelstein, "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification," *IEEE Trans. Software Eng.*, vol. 20, no. 10, pp. 760–773, Oct. 1994.

[31] R. Prieto-Diaz and P. Freeman, "Classifying Software for Reusability," *IEEE Software*, vol. 4, pp. 6–16, Jan. 1987.

[32] H.B. Reubenstein and R.C. Waters, "The Requirements Apprentice: Automated Assistance for Requirements Acquisition," *IEEE Trans. Software Eng.*, vol. 17, no. 3, pp. 226–240, 1991.

[33] W.N. Robinson, "Negotiation Behavior During Requirement Specification," *Proc. Int'l Conf. Software Eng.*, pp. 268–276, 1990.

[34] W.N. Robinson and S. Fickas, "Supporting Multi-Perspective Requirements Engineering," *Proc. First Int'l Conf. Requirement Eng.*, pp. 206–215, IEEE CS Press, 1994.

[35] G.C. Roman, "A Taxonomy of Current Issues in Requirements Engineering," *Computer*, vol. 18, no. 4, pp. 14–21, Apr. 1985.

[36] J. Rushby, "Quality Measures and Assurance for AI Software," Technical Report NASA CR-4187, NASA Langley Research Center, 1989.

[37] T.L. Saaty, *Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World*, Lifetime Learning, Atlanta, 1982.

[38] K. Sycara, "Resolving Goal Conflicts Via Negotiation," *Proc. Sixth Nat'l Conf. Artificial Intelligence*, pp. 245–250, Cambridge, Mass., MIT Press, 1988.

[39] B.M. Werner, "Aggregation Models in Mathematical Programming," G. Mitra, ed., *Math. Models for Decision Support*, pp. 295–305, Springer-Verlag, Berlin, 1988.

[40] R.R. Yager and D.P. Filev, "On the Extension of Owa Operators," *Proc. Sixth Int'l Fuzzy Systems Assn. World Congress*, vol. II, pp. 161–163, 1995.

[41] J. Yen and J. Lee, "Fuzzy Logic as a Basis for Specifying Imprecise Requirements," *Proc. Second Int'l Conf. Fuzzy Systems*, pp. 745–749, 1993.

[42] J. Yen and J. Lee, "A Task-Based Methodology for Specifying Expert Systems," *IEEE Expert*, vol. 8, no. 1, pp. 8–15, Feb. 1993.

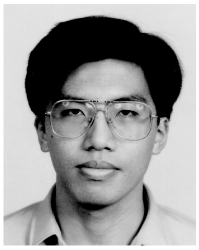
[43] J. Yen, X. Liu, and S.H. Teh, "A Fuzzy Logic-Based Methodology for the Acquisition and Analysis of Imprecise Requirements," *Concurrent Eng.: Research and Applications*, pp. 265–277, 1994.

[44] L.A. Zadeh, "Fuzzy Set as a Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

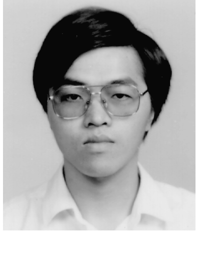
[45] L.A. Zadeh, "Test-Score Semantics as a Basis for a Computational Approach to the Representation of Meaning," *Literacy Linguistic Computing*, vol. 1, pp. 24–35, 1986.

[46] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications*, Kluwer, Boston, 1991.

[47] H.-J. Zimmermann and P. Zysno, "Decisions and Evaluations by Hierarchical Aggregation of Information," *Fuzzy Sets and Systems*, vol. 10, pp. 243–260, 1983.



**Jonathan Lee** received his PhD degree in 1993 in computer science from Texas A&M University. He is now an associate professor in the Software Engineering Laboratory of the Department of Computer Science and Information Engineering at the National Central University in Taiwan, Republic of China. His research interests include requirements engineering, methods integration, knowledge-based software engineering, and fuzzy logic. He is a member of the IEEE Computer Society, the ACM, and the AAAI.



**Jong-Yih Kuo** received his BS degree from National Tsing Hua University, Taiwan, Republic of China, in 1991, and his MS degree from the National Central University, Taiwan, in 1993. He is now a PhD student in the Software Engineering Laboratory of the Department of Computer Science and Information Engineering at the National Central University, Taiwan. His research interests include requirement engineering and fuzzy logic.