Attributed Goal-Oriented Analysis Method for Selecting Alternatives of Software Requirements

Kazuma YAMAMOTO*, Nonmember and Motoshi SAEKI†, Member

SUMMARY During software requirements analysis, developers and stakeholders have many alternatives of requirements to be achieved and should make decisions to select an alternative out of them. There are two significant points to be considered for supporting this decision making processes in requirements analysis; 1) dependencies among alternatives and 2) evaluation based on multi-criteria and their trade-off. This paper proposes the technique to address the above two issues by using an extended version of goal-oriented analysis. In goal-oriented analysis, elicited goals and their dependencies are represented with an AND-OR acyclic directed graph. We use this technique to model the dependencies of the alternatives. Furthermore, we associate attribute values and their propagation rules with nodes and edges in a goal graph in order to evaluate the alternatives with them. The attributes and their calculation rules greatly depend on the characteristics of a development project. Thus, in our approach, we select and use the attributes and their rules that can be appropriate for the project. TOPSIS method is adopted to show alternatives and their resulting attribute values.

key words: goal-oriented analysis, requirements analysis, attribute grammar, decision making, TOPSIS method, attributed graph

1. Introduction

During software requirements analysis and design steps, analysts, designers and stakeholders have many alternatives of artifacts and should make decisions to select an alternative out of them. For example, in requirements analysis, the analysts select the requirements that will be implemented as software, while the designers choose suitable software components to implement the software if they adopt component technology.

As an example, suppose that a requirements analyst elicits requirements from stakeholders using goal-oriented requirements analysis, one of the promising methodology for requirements elicitation [4], [12], [15]. In this methodology, customers' needs are modeled as goals to be achieved by software-intensive systems that will be developed, and the goals are decomposed and refined into a set of more concrete sub-goals. After finishing goal-oriented requirements analysis, the analyst obtains an acyclic (cycle-free) directed graph called goal graph. Its nodes express goals to be achieved by the system that will be developed and its edges represent logical dependency relationships between the connected goals. More concretely, a goal can be decomposed into a sub-goals, and the achievement of the sub-goals contributes to its achievement. We have two types of goal decomposition; one is AND decomposition and another is OR. In AND decomposition, if all of the sub-goals are achieved, their parent goal can be achieved or satisfied. On the other hand, in OR decomposition, the achievement of at least one sub-goal leads to the achievement of its parent goal. Figure 1 illustrates a part of the goal graph which has been obtained from requirements analysis of a support system of Internet Auction System like eBay. The edges to which a node attaches show an AND decomposition, and for example, two goals B (Retrieving Goods) and C (High Usability for Various Users) should be achieved in order to achieve their parent goal A in the figure. On the other hand, either D, E or both are necessary to be achieved for the goal B. Thus the analyst and the stakeholders should decide which is adopted from the three alternatives D, E or both of D and E, when the software system will be implemented. To get a final product of high quality, the supports of decision making for these alternatives are necessary.

There are two significant points to support the decision making processes in goal-oriented requirements analysis; 1) dependencies among goals and 2) multi-criteria for evaluation and their trade-off. In the first point, the selection of a goal X may force the analyst to adopt the goal Y in the case that X and Y has a dependency. As for the second point, we have several criteria to evaluate alternatives such as quality attributes (performance, security, reliability, ..., cost etc.), and some of them may cause conflicts, e.g. the higher reliability may decrease performance because the system should execute many error checking operations.

Fig. 1 An example of a goal graph for Internet auction system.
In this paper, we illustrate how to solve the above issues on goal graphs. Our idea is the usage of the extended version of goal graph and of a decision making technique for multi-criteria. As for the first issue mentioned above, the logical dependencies can be expressed as edges of a goal graph. To deal with multi-criteria for evaluation, we use attributed goal graphs, where specific attributes are attached to nodes (goals) and rules for calculating their values are associated with edges. The attached attributes are considered as evaluation criteria and the attribute values calculated with the attached rules are used for evaluating an alternative. It greatly depends on the characteristics of a development project which attributes are appropriate for usage of evaluation. For example, suppose that you have a profit-oriented project. The attributes of “costs spent in the project” and “needs of the products in a market” may be more significant to evaluate the alternatives included in the project, because the products should be completed with cheaper cost and much more customers should buy them with higher prices to get more profit. From this example, we should deal with wide varieties of attributes, not only with quality attributes and non-functional requirements, and the appropriate attributes vary on the characteristics of the project such as its aim. The attributes and their calculation rules can be defined according to a development project in our approach.

After evaluating the attributes, for each alternative we get a vector whose elements are the attribute values, and the users of our approach select an alternative by considering the values of the vector. To support this process, it is useful to sort the alternatives with these values, in particular in the case where they have many evaluated attributes. And when we select an alternative by taking every attribute value into consideration, we may sometimes give higher priority to specific attributes than the others. Appropriate priority greatly depends on the characteristics of a project, and we should specify it project by project. We adopt a multi-criteria decision making (MCDM) method called TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) to sort the alternatives. This method produces from multiple values a single value to grade the alternatives. The reason why we adopt TOPSIS is as follows. Suppose that the analyst adopts the attributes “development cost” and “the number of functions” to select their alternatives. In the case of development cost, the smaller is the better, while the larger is the better in the number of functions. In addition, their value scales and units are different. The TOPSIS method can solve these problems. Furthermore, weighted factors of evaluation criteria in TOPSIS can give priority to specific attributes for evaluation.

Note that we don’t force the users to select the alternative that is graded as the best one with TOPSIS. The users of our approach are only persons that can make a final decision by investigating the attributes values and the result of TOPSIS. And we don’t propose a universal and definitely new MCDM technique to output the “true” best alternative in any development project, but a technique adaptable for the characteristics of projects so that it can suggest to stakeholders the basis of selecting the alternative. The attribute values of goals obtained from calculation may be good basis to provide the reasons why the alternative should be selected.

The rest of the paper is organized as follows. The overview of our approach including the process to evaluate alternatives is presented in the next section. We discuss a supporting tool that has been developed in Sect. 3. We had two experiences in the application of our approach and the tool, and discuss its feasibility and findings obtained from these experiences in Sect. 4. Sections 5 and 6 present related work and conclusion respectively.

2. Overview of Our Approach

2.1 Attributed Goal Graph

As mentioned in Sect. 1, the contribution of this paper is the application of an attribute goal graph technique to a multi-criteria decision making (MCDM) for supporting alternative selection in requirements analysis. We start with the discussion on the attributed goal graph technique.

Figure 2 illustrates a part of an attributed goal graph that expresses the Internet Auction System mentioned in Fig. 1. It includes several attributes, for example, the goal E (Retrieving using Rating of Sellers) has 4 attributes “development period (simply, period in the figure)”, “total of development period (t_period)”, “needs of a market (needs)”, and “number of competitors (competitors)”. Note that, for simplicity, we omit not only several nodes and edges of the graph but also attributes and their calculation rules that are not necessary to explain the mechanism of attribute calculation.

To achieve the Internet Auction System, we should achieve both of its sub-goals “Retrieving goods” and “High Usability for Various Users” in this development project example, because the goal A is refined into B and C in AND decomposition. The goal B is further decomposed to D and E, and this decomposition specifies that we should achieve...
either D, E or both. Attributes are attached to goals, i.e. nodes in the graph, and we can attach any attributes to goals, according to the characteristics of the development project and to aims of evaluating a goal graph. In the example of the figure, we attached “period” and “t_period” to the goal A, while “period”, “t_period”, “competitors” and “needs” were attached to E. As shown in this example, the attributes are not restricted to quality attributes like ones listed up in ISO9126 [6] or types of non-functional requirements such as NFR types [3]. In this example, we adopted the attribute “period” for A to estimate the development period of Internet Auction System from the decomposed sub-goals. On the other hand, in order to evaluate the needs of a market of auction software on the function “Retrieving using Rating of Sellers”, we attached “needs” to the goal E.

The attribute value is represented with its attribute name prefixed with a goal name. For example, E.period stands for the values of the attribute “period” of the goal E, i.e. the development cost of achieving E.

In Fig. 2, the development period of achieving the goal B can be estimated as the summation of the values of D and E, because D and E are necessary for the achievement of C and they are independently and sequentially developed in this example project. In other words, the period of C is derived from those of D and E, and we call this attribute derived attribute of the goal C. The calculation rules are in the form of equation and a term of left hand side in an equation is instantiated with the value of the expression of a right hand side. The rules are attached to the edges outgoing to the goal. In the figure, each rule is enclosed in a rectangle. Furthermore the rules represent a kind of dependencies among goals from a view of derivation of their attribute values. When the concrete values of the development periods of D and E, e.g. 5 and 4 units are given, the value of C is calculated, e.g. to 9 (5 + 4) units. In a real development project, a problem may be the way to get the concrete attribute values such as 5 and 4. Of course, the way how to estimate these values greatly depends on the used attributes. In the case of development period and cost, we can apply the established estimation methods such as Function Point and COCOMO to the goals that are refined as operational ones. If we already have existing estimation methods applicable to the goals, we can do. However, we cannot restrict or anticipate what attributes users of our approach really use in their project. For each attribute, the topic on the methods to estimate its concrete attribute values with constants is out of scope of this paper.

Generally, our attribute calculation starts with the attributes that can be actually calculated and all of the attribute values are decided if possible. The calculation procedure constructs an attribute dependency graph before starting calculation. In the case where the calculation is not possible to start, e.g. when we did not give the constant values to “period.D” and “period.E”, the procedure explores the attribute dependency graph and detects which attributes should be instantiated for starting the calculation. This calculation mechanism is the same as attribute evaluation of Attribute Grammar [9].

The attribute “period” of the goal B corresponds to a synthesized attribute of Attribute Grammar, which is derived from the attribute values of its child nodes. Like Attribute Grammar, we can use inherited attributes and they are derived from the attribute values of its brothers and/or a parent node. The attribute “t_period” of goal E is an example of inherited attributes. It results from the calculation of the total development period of Internet Auction System and is propagated from the root goal A. This attribute value is used to calculate the attribute “needs” of E (needs of a market to achieve the goal E). Suppose that we try to estimate the needs of a market to implement the function of the goal E. Intuitively speaking, the delay of its development causes a decrease of its market needs because the other competitors may complete its implementation and release it to the market earlier. Thus, in this example, we attach the calculation rule such as E.needs = 10/(t_period.E×(competitors.E+1)), where the maximum needs of a market (in the case where there are no competitors) can be estimated as 10 unit. This expression specifies that the more competitors can complete the development if the longer development period is spent, and that as a result, the need of a market decreases.

The case of attaching the rules to the goal B is more complicated. Since B is decomposed into D and E in OR style, we have three alternatives to achieve A, i.e. the achievement of D, E, or both. So, we can attach three different calculation rules for these alternatives. For example, if the analyst selects D, the development period of B is the same as D because he or she will implement D only. On the other hand, if he or she selects both D and E, the development period of B is calculated from D and E, as shown in Fig. 2.

The significant points in our approach are as follows.

1. The attributes to evaluate goals greatly depend on the characteristics of a project. In our approach, we don’t restrict the evaluation criteria to non-functional requirements like NFR Types of NFR Framework or to quality attributes of ISO9126, but we can attach any attributes according to project characteristics, e.g. “the number of competitors who are achieving the goal” to evaluate needs from a market of software systems to be developed. We can use any data types of attributes, not only integer and float types but also string and compound types such as vector and record types, similar to usual programming languages. For example, when we adopt an attribute whose values vary on stakeholders, we can use a vector value whose element is the attribute value of each stakeholder.

2. The calculation rules of attribute values are defined by users of our approach. We can attach any expres-
visions according to the characteristics of a development project. We can use any computable expressions that include functions and variables in addition to arithmetic operators and constants, to specify complicated relationships among attributes.

As mentioned above, we have applied an AND-OR graph of goal-oriented requirements analysis methods to describe dependencies among alternatives, including dependencies among attribute values. The attributes of goals are calculated along the logical structure of the goal graph, using calculation rules attached to edges.

Note that the example of Fig. 2 in this section is for the explanation of attributes and attribute calculation only. These attributes and rules in this example were not from a real project, and we don’t require that the attributes and rules of the example, e.g. “needs” and “E.needs = 10/(E.t.period x (E.competitors + 1))”, should be always used in any development projects. The users of our approach should define the attributes and rules appropriate for their projects and their aims.

2.2 Decision Making Process

Figure 3 depicts the process of decision making of alternatives in our approach. In the first step, we apply a goal-oriented analysis to model the alternatives and their dependencies, and obtain a goal graph whose nodes represent the alternatives. The evaluation criteria and their weighted factors are determined in the step 2, and the evaluation criteria are attached as attributes of the nodes in the goal graph. The calculation rules to evaluate the attribute values are attached to the edges of the goal graph in the step 3. In the step 3, we can get as vectors the resulting values of evaluation criteria for alternatives. The analysts and stakeholders make a final decision by considering these vector values. To support their final decision, it is helpful to sort the alternatives based on the vector values, especially in the case where there exist many alternatives. Multiple criteria decision making (MCDM) methods can be applied to arrange them in order or to prioritize them. Note that we don’t force our users to select the alternative to which our MCDM gives the highest priority. It provides them with useful information on their making a final decision. For this aim, we adopted TOPSIS method from a family of MCDMs. Finally, in the step 4, the alternatives are automatically evaluated based on TOPSIS method according to the calculation rules and the weighted factors, and sorted.

Throughout this section, we use an example of requirements to an Internet Auction System [2], whose goal graph was shown in Fig. 1.

Step 1. Modeling the Dependencies among Alternatives

By using goal-oriented analysis, the dependencies among alternatives are modeled and represented with an AND-OR graph. See Fig. 1. Our step starts after a requirements analyst completes a goal graph. The node “Internet Auction System” is decomposed into “Retrieving Goods”, and “High Usability for Various Users” in AND decomposition, and it means that a developer of the example should achieve both issues on retrieving goods and high usability. The former sub-goal is for users to look for the goods that they want. In the latter sub-goal, the analyst considers wide varieties of users, e.g. skills and knowledge about personal computers and auction processes. The goal “Retrieving Goods” has two alternatives; “Retrieving by Categories” and “Retrieving using Rating of Sellers”, as shown in OR decomposition from Internet Auction System node in the figure. Only if the analyst selects “Retrieving using Rating of Sellers”, he or she should consider the technique or function to rate the sellers of auctions and then decomposes this goal into the sub-goal “Rating Sellers by Voting”. He or she elicits the technique to provide rating of a seller by voting of the users experienced in buying from this seller before.

Step 2. Determining Evaluation Criteria and Weighted Factors

Since the evaluation criteria are based on the attributes attached to the nodes, we can use any criteria if they can be quantified. In this example, we adopt Functionality and (Development) Cost as criteria. Weighted factors are also adopted and we give 0.5 and 0.5 to Functionality and Cost respectively. The sum of the weight factors over attributes shall be 1.0. These weighted factors depend on the characteristic of the project and we define weights appropriate for the project.

Step 3. Attaching Calculation Rules

The rules of calculating the attribute values are defined and attached to the graph created in the step 1. When some attribute values are instantiated, following the rules, new values are derived or propagated, as mentioned in Sect. 2.1. Figure 4 shows the calculation rules attached to the graph of Fig. 1. For simplicity, we attach to the graph the attribute “cost” and its calculation rules only.

For example, suppose that the analyst selects “Retrieving by Categories” only for the achievement of “Retriev-
YAMAMOTO and SAEKI: ATTRIBUTED GOAL-ORIENTED ANALYSIS METHOD FOR SELECTING ALTERNATIVES OF SOFTWARE REQUIREMENTS

Fig. 4 A graph and the attached calculation rules.

Fig. 5 Attribute evaluation.

Step 4. Evaluating the alternatives

In this step, the preference degree of each alternative is evaluated based on the attached calculation rules and weighted factors.

Step 4-1 Evaluating Attributes

For each alternative, the attribute values of each node of the graph are calculated. In our example, we select a root node “Internet Auction System” as an evaluated goal. We can select any nodes and attributes included in alternatives for calculating their evaluation values, according to the characteristics of a development project. In this example, we have 6 alternatives and the cost values of the root node for each of them are selected for calculation. Similar to the attribute evaluation technique of Attribute Grammar, following the calculation rules, the attributes possible to be calculated are calculated at first. Figure 5 illustrates the calculation result of the alternative adopting the requirements “Rating Sellers by Voting” and “Alert Functions” (having alert functions), and as a result we get the cost 13 units (A.cost = 13) to achieve the alternative in Internet Auction System. Gray colored nodes and edges in the figure are not selected as the alternative of this example.

Step 4-2 Integrating the Resulting Attribute Values

After calculating all of the selected attribute values of the evaluated node for each alternative, we apply TOPSIS method to get an order to show our users. This step is shown as follows.

1. Composing a matrix
   For each alternative, we compose a matrix D whose elements are the resulting attribute values that were selected for evaluation use. Let the matrix element $x_{ij}$ be the value of j-th attribute for the i-th alternative.

2. Normalizing the matrix
   We normalize the matrix D and get the matrix R. The element $r_{ij}$ of the matrix R is as follows:
   $$ r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{M} x_{ij}^2}} $$
   where $M$ is the number of the alternatives.

3. Weighting the matrix
   The weighted factors that have been decided in step 2 are multiplied to the matrix R. Let $w_j$ be a weighted factor for the j-th attribute. The element $v_{ij}$ of the weighted matrix V is
   $$ v_{ij} = w_j \times r_{ij} $$

4. Deciding the Best and the Worst Values
   We classify the attributes into two sets $J$ and $J'$ where $J$ consists of the attributes whose larger values are the better, and the smaller is the better for the attributes in $J'$. For example, the attribute “development cost” belongs to $J'$ because the lower cost is the better. We have two vectors $A^*$ and $A^-$ in the following:
   $$ A^* = \langle \max v_{ij} | j \in J \rangle, \langle \min v_{ij} | j \in J' \rangle $$
   $$ |i = 1, 2, \ldots, M > $$
where \( N \) is the number of the attributes used for evaluation criteria and is the sum of the elements of \( J \) and \( J' \). Each of the elements in \( A^* \) is the best value of the attribute resulting from the alternatives.

5. Calculating the Distances
For \( i \)-th alternative, we have a vector \(<v_{i1}, v_{i2}, \ldots, v_{IN}>\) from the matrix \( V \) in \( n \) dimensional space, and calculate its distances \( S_p \) from \( A^* \) and \( S_f \) from \( A^- \) as follows:

\[
S_p = \sqrt{\sum_{i=1}^{N} (v_{ij} - v_p)^2}, \quad S_f = \sqrt{\sum_{i=1}^{N} (v_{ij} - v_f)^2}
\]

Finally we can get

\[
C_p = \frac{S_f}{S_p + S_f}
\]

We use the above value as a preference degree for the \( i \)-th alternative. Figure 6 illustrates the principle of TOPSIS method. Suppose that we use the two attributes Functionality and Cost attributes as criteria to evaluate alternatives. As for Functionality, its larger value is better, while the smaller value of Cost is more preferable. In the figure, each alternative is plotted as a black dot on the two dimensional plane, whose horizontal and vertical axes are Functionality and Cost respectively. The alternatives \( A_1 \) and \( A_2 \) are the best value and the worst one of Cost attribute respectively. The worst alternative has the attribute values that are the worst case. The preference value \( C_p \) of the alternative \( i \) is calculated from the distances from the worst case and from the best one, as shown in the figure.

Table 1 shows the results of evaluating the alternatives of our example shown in Fig. 7 following the above steps. We picked up a root node “Internet Auction System” as a node to be evaluated, and thus the column “cost” stands for the cost of the root node. The column “TOPSIS” expresses the preference values of the alternatives. According to these results, we select the alternative of the requirements set [Classifying Goods into Pre-defined Categories, Rating Sellers by Voting], which has the higher grades of both Functionality and Cost. Note that from the criteria of Cost, adopting either “Classifying Goods into Pre-defined Categories” or “Rating Sellers by Voting” having the score 9 is the best selection, while selecting three requirements (the bottom row of the table) has the highest score 19 from the viewpoint of Functionality. However, the result of applying TOPSIS method suggests that we should select both “Classifying Goods into Pre-defined Categories” and “Rating Sellers by Voting” only from trade-off between Cost and Functionality. Of course, we are not forced to select invariably the alternative that TOPSIS decided as the best one. The result of applying TOPSIS method is just helpful to show orderly many alternatives to our users so that they can make a final decision easily from the attribute values.

3. Supporting Tool

The tool to support decision making processes following our approach has been implemented on our attributed goal-oriented tool AGORA [8]. It has the functions to create and edit attributed goal graphs in graphical form. We have added the four functions: 1) extracting from a goal graph alternatives by analyzing the combinations of OR decomposition and showing them to users, 2) inputting and editing attributes, their values and their calculation rules, 3) evaluating automatically the attribute values following the calculation rules based on the evaluation mechanism mentioned in Sect. 2.2, and 4) calculating automatically a preference degree for each alternative and showing to the users a sorted list of alternatives and their attribute values. Figure 7 illustrates the screenshot of the supporting tool. The window net.auction is for inputting and editing a goal graph cooperated with stakeholders during a requirements elicitation step.

After completing a goal graph, the user clicks a node to input calculation rules for its attributes. In the figure, the grayed oval, the goal “Retrieving Goods” is selected for editing the calculation rules, and it has three alternatives 1) “Retrieving using Rating of Sellers”, 2) “Retrieving by Categories” and 3) both of “Retrieving using Rating of Sellers” and “Retrieving by Categories”. The sub-window Expression Editor shows a list of the registered calculation rules for the above third alternative, as shown in the top tab of the window. These rules are being applied when selecting the alternative. The tool checks the syntax of the rules and extracts the attributes from the rules. For example, it identifies the attribute “cost” from the term “Retrieving Goods”.cost appearing in the first row in the window. The tool user currently edits the expression for calculating “Retrieving Goods”.func which is displayed in the second row. When the user clicks the button “Evaluate Alternatives” on the third line from the top of the main window, the tool con-
Constructs an attribute dependency graph using the technique of Attribute Grammar [9] and it also checks whether attribute dependencies are cyclic-free or not so as to really execute the attribute evaluation.

After that, as shown in Fig. 8, a window appears for the user to select attributes that are used in TOPSIS calculation and to specify weighted factors of the selected attributes. The upper window in the figure shows that the user selects the attribute cost of the goal “Internet Auction System”, and in bottom window, he or she has provided the weighted factors of the two attributes that he or she selected for evaluation. The radio button “Greater” expresses that the larger the attribute value is the better. The tool checks if constant values should be given to some attributes to calculate the selected attributes. For example, suppose that H.cost is not instantiated with a constant, 4 yet in Fig. 5. Using the attribute dependency graph, the tool finds that the concrete value of H.cost is necessary to calculate the A.cost (“Inter-

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieving Goods</td>
<td>Cost</td>
</tr>
<tr>
<td>Retrieving by Categories</td>
<td>9.0</td>
</tr>
<tr>
<td>Retrieving using Rating of Sellers</td>
<td>14.0</td>
</tr>
<tr>
<td>Retrieving by Categories + using Rating of Sellers</td>
<td>9.0</td>
</tr>
<tr>
<td>Retrieving using Rating</td>
<td>13.0</td>
</tr>
<tr>
<td>Retrieving by Categories + Rating</td>
<td>13.2</td>
</tr>
<tr>
<td>Retrieving by Categories + Rating + Alert</td>
<td>18.2</td>
</tr>
</tbody>
</table>

Abbreviations:
- Classifying Goods, Classifying: Classifying Goods into Pre-defined Categories
- Alert: Alert Function
- Rating: Rating Sellers by Voting
Fig. 8  Selecting attributes and setting weighted factors.

net Auction.cost”, which has been selected in Fig. 8). It generates a warning message saying that the user should input H.cost value. This function is useful to find trade-off points, when the user iteratively inputs concrete data by changing its value little by little to investigate the changes of the results of attribute evaluation and of TOPSIS.
4. Discussion

We applied our approach and the supporting tool to select alternatives in a requirements analysis step of our complete version of the Internet Auction System. We used the attribute “time efficiency” in addition to “cost” and “functionality” for selecting the alternatives.

Furthermore, we picked up the problem on component selection, which is the different type of application of our approach, not requirements elicitation. We considered the Internet Auction System would be implemented as EJB application having three layers: Implementation Method, EJB Container and Communication Layer. By applying goal-oriented analysis, we decomposed EJB application into these three layers and proceeded the decomposition and dependency analysis on software components belonging to these layers such as Session Beans for Implementation Layer, JBoss for EJB Container, Jeremie for Communication Layer, etc. The dependencies of their combinations were modeled in a goal graph. We attached to the goal graph the attributes “Time Efficiency” and “Maintainability” and their calculation rules, and by using the supporting tool, we selected the combinations of these components.

In our case study, we didn’t show that our technique can always select the “true” best alternative. In fact, to show that the selection result was truly the best alternative, we have to implement all of the alternatives under the same environment and to decide the “true” best implementation. It would be unrealistic to do in any case. Rather, we assessed the usability of our approach and tool by investigating additional efforts required by users. And through these experiences, we clarify the advantages and the shortcomings of the process of our approach.

The sizes of our two examples can be summarized in Table 2. In the table, for example, Internet Auction System had 675 alternatives if we did not consider the dependencies among requirements, while they could be filtered out into 36 using the dependencies of goals in the goal graph.

Through these two experiences, we can get the following findings:

- Extracting alternatives:
  In the example of implementing the Internet Auction System, the combinations of goals necessary to achieve the root goal were 12. However, three of them were not feasible because of dependencies among the goals. In another example, we had 675 alternatives totally, and 36 of them were obtained as the alternatives to be considered because of the dependencies. It means that a goal graph was useful to specify the dependencies so as to reduce the number of the alternatives that we should consider. As the system to be developed becomes more complicated and larger, this benefit is more effective.

- Adopted multi-criteria decision making:
  Although our case studies restricted to two or three attributes for evaluation, we can attach various types of attributes that can be quantified. The results of the examples suggested that the TOPSIS method could select alternatives by reasonable efforts of the users. In fact, the selected alternatives had a better value to a certain extent over the adopted attributes, and these experiences showed that the TOPSIS method seemed to require less efforts of the users in the case of a small number of the attributes to be evaluated. However, when the attributes to be evaluated increase, the users have more efforts which combinations of the attributes could be useful for decision making.

- Attributes and calculation rules:
  In our examples, we did not discuss how to identify the attributes and calculation rules for evaluation. The identification or selection of suitable attributes is a crucial issue. One of the solutions to this issue is the reuse of the attributes that have been frequently used, e.g. cost, performance and so on. The popular quality characteristics such as time efficiency and reliability catalogued in [3] are one of the promising candidates. In addition, weighted factors have also great effects on the selection of the alternatives. By using automatic evaluation and editing functions which are supplied by our tools, we can change attribute values and weight factors stepwise and investigate how the preference degrees are being changed. That is to say, we can tune up the attributes and weighted factors. It leads to qualitative analysis so that we can discover which attributes and weighted factors can have more affects on the selection rather than the others.

In some cases, more complicated attribute calculation rules such as probability density or distribution functions may be necessary in order to model attributes based on statistics for simulating the effects of attributes. The supports to identify and to construct complicated calculation rules, e.g. making a catalogue of

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Case studies.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives without dependencies</td>
<td>Internet Auction System</td>
</tr>
<tr>
<td>Alternatives by goal graphs</td>
<td>675</td>
</tr>
<tr>
<td></td>
<td>36</td>
</tr>
</tbody>
</table>
the rule patterns that were used and useful, may be necessary. In fact, in NFR framework, a Correlation Catalogue where the qualitative contribution relationships between NFR types were stored was proposed, and any experienced persons can submit newly found contribution relationships to it through Internet.

5. Related Work

NFR framework [3] is also a goal-oriented analysis for non-functional requirements such as performance, security etc. The dependencies among the non-functional requirements such as positive and negative contributions are modeled as Soft goal Independence Graph (SIG). A qualitative analysis technique of contribution grades from sub-goals to their parent goals was discussed in [11], and the technique to embed it to NFR framework was proposed. The four types of labels denoting positive and negative contributions, i.e. +++, + for positive contribution and --, - for negative one, can be attached to the edges of a goal graph, and following the pre-defined label propagation rules on a SIG, we can analyze qualitatively the contribution of a specific set of sub-goals to a root goal. However, NFR framework has no attribute concepts. The cataloged types called NFR types of non-functional requirements such as Security and Performance can be used as soft goal to evaluate the qualitative contribution of alternatives to a root goal only. In fact, NFR types are not attributes, for example attribute “the number of competitors” or its value constraint cannot be inherently a non-functional requirement. In addition, NFR Framework adopted not quantitative evaluation but qualitative one. The users of NFR Framework can attach either of four labels +++, +, - or -- to each edge to specify the contribution of the sub-goal to its parent one, and the fixed label propagation rule evaluates the achievement degree of a root goal quantitatively, i.e. in four levels.

In AGORA [7], nodes and edges can have attributes “preference” and “contribution” respectively, the former expresses the preference degrees of stakeholders for achieving the goal and the latter denotes how many degrees a sub-goal contributes to the achievement of its parent goal. It includes the calculation mechanisms to evaluate the contribution of sub-goals to a root goal and to detect conflicts of interests among stakeholders using preference matrices [8]. However, the attributes attached to goals are only two types: contribution value and preference matrix, and the calculation rules of contribution values based on Fuzzy logic cannot be changed according to development projects.

In [5], its authors adopted the approach similar to “contribution” of AGORA and provided the rules of calculating satisfaction and denial degrees of a goal from its sub-goals. It has the same shortcoming as AGORA’s contribution evaluation mechanism. Additionally, in any of NFR framework, AGORA, [5] and [11], the attributes and their calculating rules are tightly embedded into their framework and also support tools, and it is impossible to attach the other types of attributes and/or to use the calculation of attributes for the other purposes, according to the characteristics of development projects and the aims of evaluation. Thus, in order to get benefits from the techniques of attributed goal graphs, we should be able to define attributes and their calculation rules, according to our purposes. For example, suppose that an analyst wants to estimate the needs of a market from a goal graph. She or he attaches to a goal an attribute denoting the needs of a market, and the rules how to calculate the needs of a market from the attached attribute values of the goals, as shown in Fig. 2.

AHP (Analytic Hierarchy Process) [13] has been used to prioritize the elicited requirements [10]. In AHP, criteria and alternatives are evaluated independently by pairwise comparison. By normalizing the grades of pair-comparison results, weighted factors of the criteria and preference degrees of the alternative are calculated. One of the benefits of AHP is that we can select criteria without any restriction. However, the evaluation of the alternatives is done by fixed calculation rules and it does not reflect the dependencies among the alternatives. It may be applicable to deciding the weighted factors of attributes in our approach.

There are some mathematical techniques to find maximum or minimum evaluation values under constraints such as Linear Programming [1]. Although we used TOPSIS method because of its simplicity, we can consider the application of these techniques to multi-criteria decision making in order to get more sophisticated results suitable for specific situation, i.e. situational multi-criteria decision making.

6. Conclusion

This paper presents the application of attributed goal oriented analysis to supports of selecting alternatives from multi-criteria views. We have developed a support tool to construct attributed goal graphs and to evaluate the identified alternatives based on the attached attribute values and their evaluation results. Furthermore, we had two case studies; one is the selection of the requirements that will be implemented in a final product and another is the selection of software component sets for implementation.

The future research agenda can be listed up below.

1. Case studies to construct catalogues of useful attributes and calculation rules
   We will have more complicated and practical case studies to extract useful attributes and their calculation rules as well as to assess our approach and support tool.

2. Visualization of evaluation results
   Global features such as distribution of attribute values are sometimes useful to support a final decision making. Instead of table form where each attribute value is displayed like Fig. 9, we should develop some graphical representation forms so that global features of the attribute values such as their distribution can be comprehensively and intuitively shown to requirements analysts.
3. Symbolic analysis of attributes and iterative evaluation
   In the current version of the tool, to identify some boundary values of attributes where the order of alternatives could be changed, we have to iteratively execute attribute calculation by changing the attribute values stepwise. The functions of analyzing or executing symbolically calculation rules without any iterative execution may be helpful.

4. Goals vs. attributes, and identifying situational attributes
   As mentioned in Related Work, some attributes such as quality attributes can be dealt with goals. We should develop some guidelines suggesting which way would be suitable in this case. In our approach, its users should identify appropriate attributes for the situation of a development project, so called situational attribute, and their calculation rules. Like the approach of Correlation Catalogue of NFR framework [3], assets resulting from experiences should be stored as a knowledge base so that we can retrieve situational attributes and rules from the knowledge base.

5. Evaluation from multiple stakeholders
   Wide varieties of stakeholders participate in requirements analysis processes, and they have criteria and priorities of their own. The problem is how to deal with the evaluations varying on the stakeholders. The approach of preference matrix in AGORA [8], where stakeholders can give preference values of their own independently, can be applied.

6. Application of the other techniques except for TOPSIS method
   As mentioned in the last section, we adopted the TOPSIS method to calculate a single preference value from selected attribute values. Its benefits and shortcomings should be clarified and we exploit a suitable technique for the features of the used criteria. The other methods such as ELECTRE [14] should be explored.

7. Embedding impact analysis
   The changes of the selected requirements may propagate to the other requirements, and as a result, the preference of the alternatives may be changed. It is interesting and significant to develop this impact analysis by combining qualitative analysis with our quantitative approach.

8. Relationship among alternatives
   Relationships among alternatives such as overlaps on two alternatives should be explored. Although a sub-graph corresponding to each alternative is independently constructed and the attribute values are calculated for each sub-graph, the information on the relationships among alternatives could be useful to reduce computational costs by avoiding the multiple occurrences of the same calculations. In addition, if the stakeholders did not select the first candidate from the results of TOPSIS method, the relationships among alternatives could be useful to look for next candidates. For example, we may be able to use the inclusion relationships among the alternatives as goal graphs (sub-graph relationship) to show the next candidates of alternatives following the inclusion relationship.

References

Motoshi Saeki received a B.Eng. in electrical & electronic engineering, and M.Eng. and a PhD in computer science from Tokyo Institute of Technology, in 1978, 1980 and 1983 respectively. He is currently a professor of computer science at Tokyo Institute of Technology. His research interests include requirements engineering, software design methods, software process modeling, and computer supported cooperative work (CSCW).