Using Attributed Goal Graphs for Software Component Selection: An Application of Goal-Oriented Analysis to Decision Making

Kazuma Yamamoto and Motoshi Saeki

Dept. of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152, Japan
Email: saeki@se.cs.titech.ac.jp

Abstract

During software requirements analysis and design steps, developers and stakeholders have many alternatives of artifacts such as software component selection and should make decisions to select best alternatives out of them. There are two significant points to be considered for supporting these decision making processes; 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 model multi-criteria and to evaluate the alternatives with them.

1 Introduction

During software requirements analysis and design steps, analysts, designers and stakeholders have many alternatives of artifacts and should make decisions to select best alternatives 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 an requirements analyst elicits requirements from stakeholders using goal-oriented requirements analysis, one of the promising methodology for requirements elicitation [5, 16, 12]. In this methodology, stakeholders' needs are modeled as goals to be achieved by software-intensive systems, 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 software 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 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. Thus, the edges outgoing in OR decomposition express the alternatives to achieve the parent node. To sum up, goal graphs can represent both dependencies among decomposed sub-goals using edges and alternatives with the occurrences of OR decomposition of nodes.

There are two significant points to support the decision making processes in requirements analysis and design; 1) dependencies among alternatives and 2) multi-criteria for evaluation and their trade-off. In the first point, the selection of an alternative X may force a software engineer to adopt the alternative 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 characteristics (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.

In this paper, we illustrate how to solve the above two issues on attributed goal graphs, an extended version of goal graphs, by taking as an example the topic of selecting alternatives of software components to implement stakeholders' needs. A node and an edge on a graph represent parts of software components and their dependency relationships, respectively.

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 the rules for calculating their values are associated with edges. In an attributed goal graph, the attributes attached to the goals express the criteria to evaluate the alternatives and the technique of integrating the attribute values is necessary. It means that the way to derive from the various results of calculating attribute values a single value that can grade the alternatives is necessary. Suppose that the software engineer 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 problem is how to combine these two values into a single grade. To solve this issue, we combine a decision making technique for multi-criteria called TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [15], with the attributed goal graph technique.

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

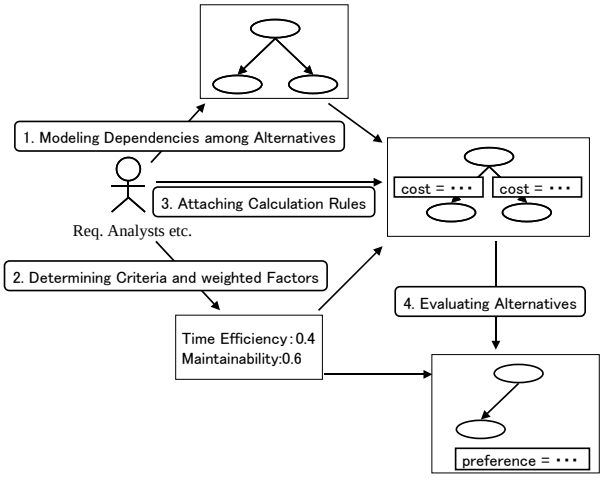

Figure 1: Process for Grading the Alternatives

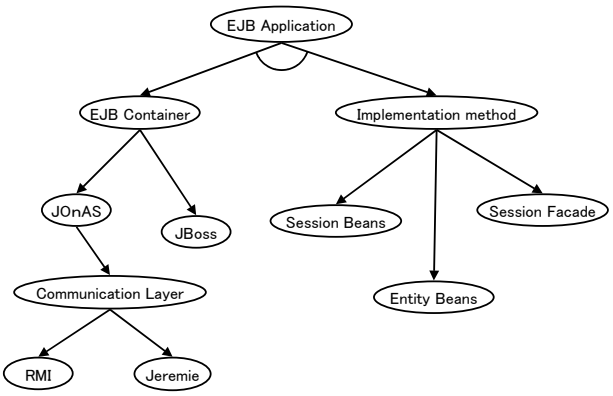

Figure 2: A Goal Graph of Implementing an Internet Auction System using Software Components

2 Using Attributed Goal Graphs and TOPSIS Method

2.1 Overview of Our Approach

As mentioned in the last section, the contribution of this paper is the combination of an attributed goal graph technique and a multi-criteria decision making method to apply the support of selecting alternatives. In this section, we explain the details of our technique using a simple illustration.

Figure 1 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 criteria are attached as attributes of the nodes in the goal graph. The calculation rules to evaluate the attached attribute values are associated with the edges of the goal graph in the step 3. Finally, in the step 4, the alternatives are automatically evaluated based on TOPSIS method according to the calculation rules and the weighted factors. The details of these steps will be mentioned in the next sub section.

Throughout this section, we use an example of implementing an Internet Auction System like eBay [3], which will be developed as an application of EJB. This example is in a software design step or later, after finishing a requirements analysis step, and its essential point is to select EJB (Enterprise Java Beans[1]) software components satisfying the elicited requirements. There are three points of decision making to be required; selections of 1) an implementation method, 2) an EJB container and 3) a communication layer, and there exist some dependencies among these three selections, as shown in Figure 2.

2.2 Decision Making Process

Step 1. Modeling the Dependencies among Alternatives :

By using goal-oriented analysis, the dependencies among the alternatives, i.e. relationships among the software components in this example, are modeled and represented with an AND-OR graph. Figure 2 illustrates an AND-OR graph of the alternatives of this example. The node "EJB application" is decomposed into EJB container and Implementation Method in AND decomposition, and it means that a developer of the example should address both issues on EJB container and on Implementation Method. EJB container has two alternatives; JOnAS (Java Open Application Server) and JBoss (Java Application Server developed by JBoss Inc.), which are open source Java EE application servers, as shown in OR decomposition from EJB Container node in the figure. Only if the developer selects JOnAS, he or she should consider Communication Layer, which has the two alternatives RMI (Remote Method Invocation of Java) and Jeremie for providing and implementing a communication protocol for distributed objects on a network. There are none of dependencies between EJB Container and Implementation Method.

Step 2. Determining Evaluation Criteria and Weighted Factors

We frequently pick up non-functional requirements and quality characteristics that the alternatives being selected affect. Since the criteria for evaluating alternatives are the attributes attached to the nodes in our approach, we can use any criteria if they can be quantified. In this example, we adopt Time Efficiency and Maintainability as criteria. Weighted factors are also adopted and we give 0.8 and 0.2 to Time Efficiency and Maintainability respectively. The sum of the weight factors over attributes shall be 1.

Step 3. Attaching Calculation Rules The attributes and the rules to calculate the attribute values are defined and attached to nodes and edges in the graph created in the step 1. Figure 3 shows the calculation rules attached to the edges of Figure 2. For example, the node A "EJB Application" has the attribute "throughput" to express Time Efficiency, and it can be calculated from the attribute value of the node H. The rule "A.throughput = H.throughput"[2] is attached to the edges outgoing from node A, and specifies how to calculate the throughput value of A. The calculation rules are similar to ones in Attribute Grammar. When some attribute values are instantiated, following the rules, new values are derived or propagated. The attributes like throughput of A that can be calculated from the values of its child nodes are called *synthesized attributes*, while the attributes that are calculated from its parent node and/or brother ones are *inherited attributes*. The attributes "cont" and "comm" of node H are the examples of inherited attributes, and are used for calculating the throughput value of Implementation Method from the selected container and the selected communication technique.

The attribute "comm" is attached to any sub-nodes

[1] A kind of software architecture for Java software components to construct enterprise information systems such as Web application.

[2] Rightfully, we should have written "EJB Application.throughput = Implementation Method.throughput". However, for simplicity, we use the simple labels A, B, C, ・・・ instead of real goal names written inside ovals in the figure.

from EJB container and its values are calculated along bottom-up direction in the graph. For example, if the developer selects RMI (Remote Method Invocation) technique for Communication Layer, the gained throughput value is 1 unit, while Jeremie is 1.4 units. Its value is propagated to the nodes C and then to B following the attached rules "C.comm = E.comm" and "B.comm = C.comm". The throughput value of Implementation Method is calculated from the values of cont and comm of its own, and the calculation rule depends on the selected Implementation Method. If the developer selects Entity Beans, the throughput value of Implementation Method can be obtained by the calculation of 600 × H.cont × H.comm. It is the example that the alternative affects the attribute values. Note that, as for the node EJB Application decomposed in AND form, the rules for calculating the three attribute values from all of its sub-nodes B and H are attached to the set of the edges to B and H, because both B and H should be simultaneously selected for the achievement of A. On the other hand, calculation rules can be attached to each edge and/or the combinations of the edges in the case of OR decomposition. As for another attribute Maintainability, we attach the attributes and the calculation rules in the similar way, in order to calculate Maintainability value.

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. This evaluation consists of two stages; one is to evaluate the attribute values based on the calculation rules and another is to integrate the resulting attribute values into preference degrees over the alternatives.

Step 4-1 Evaluating Attributes :

For each alternative, the attribute values of a root node of the graph are calculated. In our example, we have 9 alternatives and the throughput values of EJB Application for each of them are calculated. Similar to the attribute evaluation technique of Attribute Grammar, following the calculation rules, the attributes possible to be calculated are calculated at first. Figure 4 illustrates the calculation result of the alternative adopting the components JBoss and Entity Beans. In this example, we can get the throughput vale of A as 780 and it is the value of Time Efficiency. We can also combine several attribute values, e.g. computing a weighted linear average value from several values like McCall's approach [2], to express the attributes that we will use in the next step of TOPSIS Method.

Step 4-2 Integrating the Resulting Attribute Values

After calculating all of the attribute values of a root node for each alternative, we apply TOPSIS Method to get a preference degree from them. This step is mentioned as follows.

1. Composing a matrix
   For each alternative, we compose a matrix D whose elements are the resulting attribute values of the root node. 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.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{M} x_{ij}^2}}$$

   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 better. We have two vectors A^* and A^- in the following:

$$\begin{aligned} A^* &= < (\max v_{ij} | j \in J), (\min v_{ij} | j \in J') | i = 1, 2, ..., M > \\ &= < v_{1*}, v_{2*}, ..., v_{N*} > \\ A^- &= < (\min v_{ij} | j \in J), (\max v_{ij} | j \in J') | i = 1, 2, ..., M > \\ &= < v_{1-}, v_{2-}, ..., v_{N-} > \end{aligned}$$

   where N is the number of the attributes used as evaluation criteria. Each of the elements in A^*, say v_{j*} is the best value of the j-th attribute in the alternatives.

5. Calculating the Distances and Preference Degrees
   For i-th alternative, we have a vector $<v_{i1}, v_{i2}, ..., v_{iN}>$ from the matrix V in N dimensional space, and calculate its distances S_{i^*} from A^* and S_{i^-} from A^- as follows;

$$S_{i^*} = \sqrt{\sum_{i=1}^{N} (v_{ij} - v_{j^*})^2}\ ,\ S_{i^-} = \sqrt{\sum_{i=1}^{N} (v_{ij} - v_{j^-})^2}$$

   Finally we can get

$$C_{i^*} = \frac{S_{i^-}}{S_{i^*} + S_{i^-}}$$

   We use the above value as a preference degree for the i-th alternative. Figure 5 illustrates the principle of TOPSIS method. Suppose that we use the two attributes Time efficiency and Cost attributes as criteria to evaluate alternatives. As for Time Efficiency, 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 Time Efficiency and Cost respectively. The alternatives A_1 and A_2 are the best and worst one of Cost attribute respectively. The worst alternative A^- can be plotted as a vertex of the rectangular box surrounded with the worst values and the best ones of the attributes, and all of its elements are the worst. The preference value C_i of the alternative i is calculated from the distances from the worst alternative and from the best one, as shown in the figure.

Table 1 shows the results of evaluating the alternatives of our example following the above steps. The column "TOPSIS" express the preference values of the alternatives. According to these results, we select the alternative of the component set {Session Facade, JOnAS, Jeremies}, which have the most preferable value from the viewpoints of both Time Efficiency and Maintainability.

3 Support Tool

The tool to support decision making processes following our approach has been implemented on our attributed goal-oriented tool AGORA [9]. 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

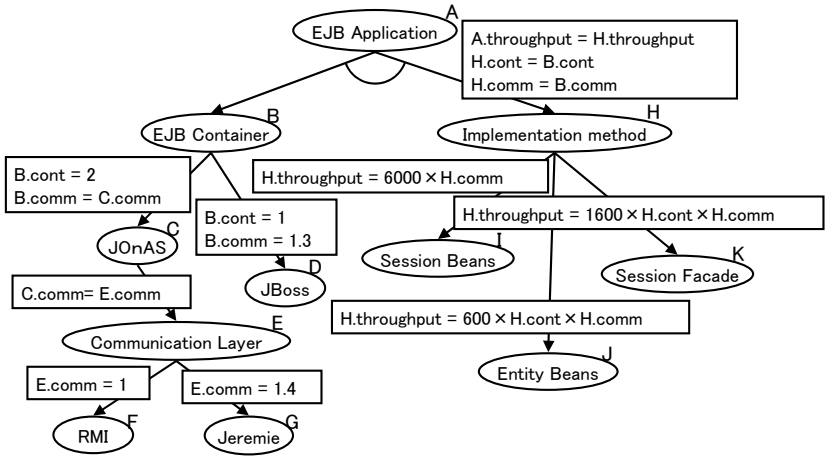

Figure 3: A Graph and the Attached Calculation Rules

Table 1: Evaluation Results

Alternatives			Evaluation value		
Implementation Method	**EJB Container**	**Communication Layer**	**Time Efficiency**	**Maintainability**	**TOPSIS**
Session Beans	JBoss	–	7800	2	0.5173
	JOnAS	RMI	6000	2	0.3048
		Jeremie	8400	2	0.7662
Entity Beans	JBoss	–	780	6	0.1050
	JOnAS	RMI	1200	5	0.1121
		Jeremie	1680	8	0.2161
Session Facade	JBoss	–	2080	9	0.3324
	JOnAS	RMI	3200	5	0.5376
		Jeremie	4480	8	0.8417

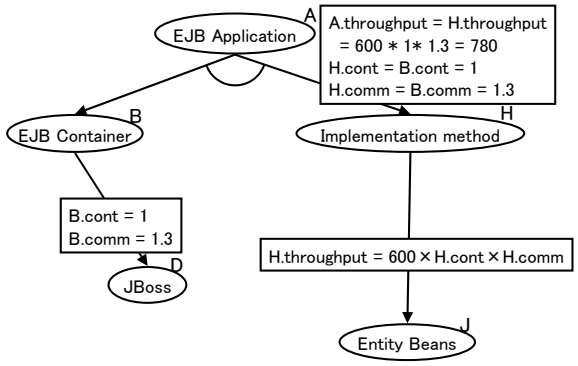

Figure 4: Attribute Evaluation

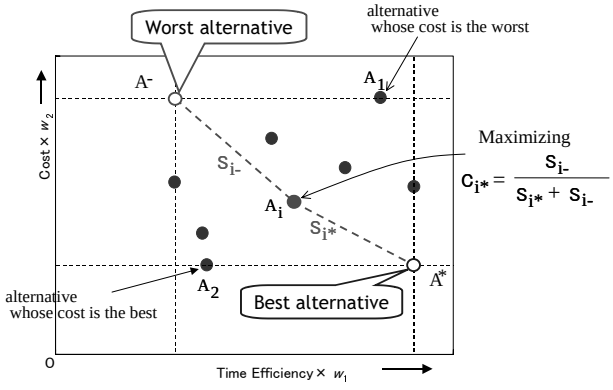

Figure 5: TOPSIS Method for Multi-Criteria Evaluation

values following the calculation rules, and 4) calculating automatically a preference degree for each alternative and showing it to the users. Figure 6 illustrates the screenshot of the supporting tool. The window AGORA Editor is for inputting and editing a goal graph. After completing a goal graph, the user clicks a node to input calculation rules for its attributes. The calculation rules are attached to the edges outgoing from the clicked node. In the figure, the reversed oval, the goal "EJB Application" is selected for editing the calculation rules. The sub window Expression Editor shows a list of the registered calculation rules. The tool checks the syntax of the rules and extracts the attributes from the rules. For example, it identifies the attribute "throughput" from the term "EJB Application.throughput" appearing in the first row in the window. In addition, the top line "<EJB Container><Implementation Method>" of Expression Editor shows that the rules displayed below rules has been attached to a set of two edges to EJB container and Implementation Method sub goals. The tool also checks whether attribute dependencies are cyclic-free or not, so that it can really execute the attribute evaluation.

4 Discussion

Through the case study mentioned in section 3, we have got the following findings;

- Extracting alternatives:
  In the example of the 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. 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.

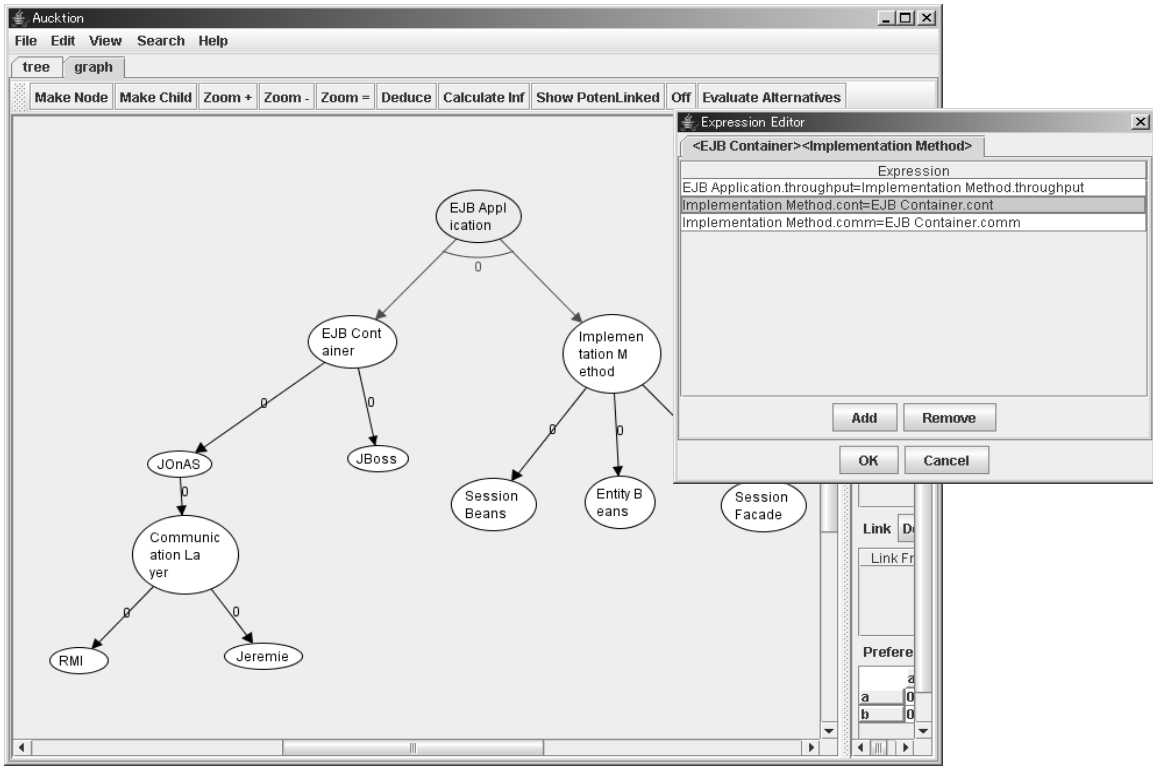

Figure 6: A Screenshot of the Supporting Tool

As the system to be developed becomes more complicated and larger, this benefit is more effective.

- Multi-criteria decision making:
  Although our case studies restricted to two attributes for evaluation, we can attach various types of the attributes that can be quantified. The results of the examples suggested that the TOPSIS method could select reasonable alternatives. In fact, the selected alternative had a better value to a certain extent for every attribute, and this experience showed that the TOPSIS method seemed to work well in the case of a small number of the attributes to be evaluated. However, when the attributes to be evaluated increases, we have to consider the other established methods for dealing with the larger number of criteria [14].

- 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 ISO9126 [7] 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 our examples, we adopted simple calculations consisting of four fundamental arithmetic operators ($+$, $-$, $\times$, $/$) only. In some cases, more complicated rules such as probability density or distribution functions may be necessary in order to model attributes based on statistics. The supports to identify and to construct complicated calculation rules, e.g. making a catalogue of the rule patterns that were used before and will be useful, may be necessary.

5 Related Work

There are some extended versions of goal graphs having attributes in nodes and/or edges. In AGORA [8], 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 quality of a goal graph such as completeness and to detect conflicts of interests among stakeholders [9]. In [6], 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. A qualitative analysis technique of contribution grades from sub-goals to their parent goals was discussed in [11]. The four types of labels denoting positive and negative contributions can be attached to the edges of a goal graph, and following the pre-defined label propagation rules, we can analyze qualitatively the contribution of a specific set of sub-goals to a root goal. In any of them, however 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. 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 a software engineer wants to estimate the development cost from a goal graph. She or he attaches to a goal an attribute denoting the cost necessary to achieve it, and the rules how to calculate the development cost from the attached attribute values of the goals.

NFR framework [4] 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 Independency Graph (SIG), and using this graph, we can estimate the degree of satisfying the non-functional requirements appearing in the SIG. However it does not include the support of integrating these satisfaction degrees into a preference degree for selecting alternatives. The analysts should simultaneously examine the satisfaction degrees of the multiple non-functional requirements.

AHP (Analytic Hierarchy Process) [13] has been used to prioritize the elicited requirements [10]. In AHP, criteria and alternatives are evaluated independently by pair-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 useful for 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 selection supports of 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.

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. 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 [9], where stakeholders can give preference values of their own independently, can be applied.

3. Application of the other techniques except for TOPSIS method
   As mentioned in the last section, we adopted the TOPSIS method to integrate the attribute evaluations into a single preference value. 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.

4. 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.

References

[1] J. Akker, Brinkkemper, G. Diepen, and J. Versendaal. Determination of the Next Release of a Software Product: an Approach using Integer Linear Programming. In *Proc. of CAiSE'05 Forum*, pages 119–124, 2005.

[2] J. P. Cavano and J. A. McCall. A Framework for the Measurement of Software Quality. In *Proc. of ACM Software Quality Assurance Workshop*, pages 133–139, 1978.

[3] E. Cecchet, J. Marguerite, and W. Zwaenepoel. Performance and Scalability of EJB Applications. In *Proc. of the 2002 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA2002)*, pages 246–261, 2002.

[4] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Academic Publishers, 1999.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed Requirements Acquisition. *Science of Computer Programming*, 20:3–50, 1993.

[6] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with goal models. In *Lecture Notes in Computer Science (ER2002)*, volume 2503, pages 167–181, 2002.

[7] ISO. Information Technology – Software product evaluation – Quality characteristics and guidelines for their use, 1991.

[8] Haruhiko Kaiya, Hisayuki Horai, and Motoshi Saeki. AGORA: Attributed Goal-Oriented Requirements Analysis Method. In *IEEE Joint International Requirements Engineering Conference, RE'02*, pages 13–22, Sep. 2002.

[9] Haruhiko Kaiya, Daisuke Shinbara, Jinichi Kawano, and Motoshi Saeki. Improving the detection of requirements discordances among stakeholders. *Requirements Engineering*, 10(4):289 – 303, Dec. 2005.

[10] N. Maiden, P. Pavan, A. Gizikis, O. Clause, H. Kim, and X. Zhu. Making Decisions with Requirements: Integrating I* Goal Modelling and AHP. In *Proc. of REFSQ'2002 Workshop*, pages 24–35, 2002.

[11] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using non-functional requirements : A process-oriented approach. *IEEE Trans. on Soft. Eng.*, 6(4):489–497, 1992.

[12] John Mylopoulos, Lawrence Chung, and Eric Yu. From Object-Oriented to Goal-Oriented Requirements Analysis. *Communications of the ACM*, 42(1):31–37, Jan. 1999.

[13] T. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[14] E. Triantaphyllou. *Multi-Criteria Decision Making Methods : A Comparative Study*. Kluwer Academic, 2000.

[15] E. Triantaphyllou, B. Shu, S. Sanchez, and T. Ray. Multi-criteria Decision Making: An Operations Research Approach. In *Encyclopedia of Electrical and Electronics Engineering*, volume 15, pages 175–186, 1998.

[16] Axel van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *RE'01*, pages 249–263, Aug. 2001.