IEICE TRANS. INF. & SYST., VOL.E95–D, NO.4 APRIL 2012

PAPER Special Section on Knowledge-Based Software Engineering

Impact Analysis on an Attributed Goal Graph*

Shinpei HAYASHI[†], **Daisuke TANABE**[†], **Haruhiko KAIYA**[††], *Nonmembers*, *and* **Motoshi SAEKI**[†a)], *Member*

SUMMARY Requirements changes frequently occur at any time of a software development process, and their management is a crucial issue to develop software of high quality. Meanwhile, goal-oriented analysis techniques are being put into practice to elicit requirements. In this situation, the change management of goal graphs and its support are necessary. This paper presents a technique related to the change management of goal graphs, realizing impact analysis on a goal graph when its modifications occur. Our impact analysis detects conflicts that arise when a new goal is added, and investigates the achievability of the other goals when an existing goal is deleted. We have implemented a supporting tool for automating the analysis. Two case studies suggested the efficiency of the proposed approach.

key words: *goal-oriented analysis, impact analysis, change management*

1. Introduction

Requirements changes frequently occur after a requirements specification is completed and even during its requirements elicitation step, by various reasons such as changing business goals and improving information technology. In this situation, it is a crucial issue how to manage requirements changes, more concretely to analyze impacts and change propagation for keeping consistency etc.

Meanwhile, goal-oriented or goal-driven analysis methods [2] have been used for requirements elicitation and are putting into practice [3]. In goal-oriented analysis, the customers' and/or users' abstract goals to be achieved are gradually decomposed into more concrete sub-goals. This decomposition process is recorded as a graph whose nodes and edges respectively express the goals and their decomposition relationships. This graph is called a *goal graph*. Thus a goal graph includes dependency relationships among the goals.

Impact analysis is a promising technique for requirements change management, and it can effectively work on a goal graph. Requirements analysts produce a requirements specification documents, e.g., in IEEE 830 standard compliant form [4], based on the derived goals. Although a final product in requirements analysis step is not a goal graph but a specification document, and change management can be done on the document, managing changes on the goal graph is more useful rather than on the document [5]. This is because goal graphs include the information how to derive the goals and why the goals are derived, and these information are useful in order to maintain forward traceability. Based on the process that requirements changes are managed on a goal graph, when a requirements change occurs, a requirements analyst does not modify the specification document directly but rather do the goal graph. Using semi-automated techniques to generate IEEE 830 requirements document from goal graphs [6], the analyst can obtain the latest requirements document from the latest goal graph. Under the situation above, measuring the impacts of requirements changes on a goal graph is useful in order to keep consistency of the managed goal graph.

This paper presents a supporting technique to manage a changes on goal graphs. We use an extended version of goal graph, called *attributed goal graph* [7]. We propose the technique for change management and its supporting tool for an attributed goal graph. Structural changes on attributed goal graphs include addition and deletion of nodes and edges. In this paper, we focus on the support of impact analysis when the structure of a goal graph is changed. When an analyst adds a new goal to a goal graph, conflicts to the other existing goals may occur. The deletion of an existing goal may cause the failure in achieving certain goals. The detection of these conflicts and of the achievement failure should be automated because the detection by manual is not so easy in the case of the more complicated and larger goal graph. We have embedded the functions of impact analysis into the existing tool of Attributed Goal-Oriented Requirements Analysis (AGORA) method, which had been developed by us [7] to measure the quality of requirements being elicited based on IEEE 830.

In our previous papers related to AGORA, in addition to the measurement of requirements quality, we have shown the other beneficial points of attached attributes to a goal graph, e.g., the supports for detecting discordances among stakeholders [8] and for making decisions on requirements selection [9]. The contribution of this paper is to propose the technical support of changing requirements on an attributed goal graph elicited by AGORA method. In particular, we adopt new attribute called *goal characteristics* and show that AGORA's attributes are useful for impact analysis of the graph.

The rest of the paper is organized as follows. We in-

Manuscript received July 1, 2011.
Manuscript revised October 31, 2011.

[†]The authors are with the Department of Computer Science, Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
[††]The author is with the Department of Computer Science, Shinshu University, Nagano-shi, 380–8553 Japan.

*This paper is revised based on [1], which appeared in the proceedings of 16th IEEE International Requirements Engineering Conference, © 2008 IEEE.

a) E-mail: saeki@se.cs.titech.ac.jp

DOI: 10.1587/transinf.E95.D.1012

troduce attributed goal graphs in the next section. Section 3 presents the techniques of impact analysis on an attributed goal graph. The qualitative evaluation of the tool together with case studies is discussed in Sect. 5. Sections 6 and 7 are respectively for related work and concluding remarks.

2. Preliminary: Attributed Goal Graph

In goal-oriented analysis, 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 analysis, a requirements analyst obtains an directed acyclic (cycle-free) 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 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 the other is OR. On the one hand, 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 a goal graph which has been obtained from requirements analysis of a Web account system of high quality. Ovals (nodes) and arrows (directed edges) respectively express goals and decomposition relationships among the goals. The edges attached with an arc outgoing from a parent node show an AND-decomposition. For example, two goals "Easy to register an account" and "Others do not register me" should be achieved in order to achieve their parent goal in the figure. In contrast, either "Password Authentication", "Authentication by SSH", or both are necessary to be achieved for the goal "Others do not register me". The usage of this type of graph, so called AND-OR graph, is a common feature in a family of goal-oriented analysis.

The AGORA method using *attributed goal graphs* is an extended version of the goal-oriented analysis. In AGORA method, a requirements analyst can attach several types of attributes to a goal graph during constructing it [10]: 1) contribution values to edges, 2) preference matrices to nodes, 3) the definitions of additional attribute names, their value types, and their calculating expressions to nodes, and 4) semantic tags to nodes. The contribution values and semantic tags can be utilized to analyze the impact of changes on a goal graph in this paper. The preference matrices and the definitions are respectively used for detecting requirements discordances among stakeholders [8] and for evaluating alternatives of requirements [9]. These two types of attributes are not used in this paper, and we focus on contribution values and semantic tags.

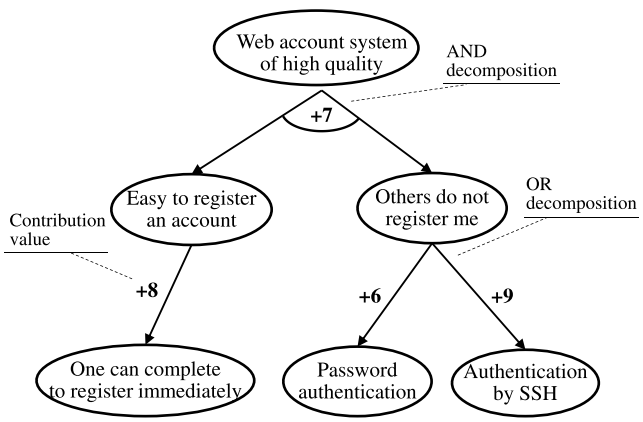


Fig. 1 Example of attributed goal graph for a Web account system.

The contribution value is attached to an edge between a parent goal and its sub-goal. It can be an integer from -10 to $+10$. The value indicates the contribution degree of a sub-goal to the achievement of its parent goal. The higher the value is more contribution the sub-goal provides. Negative values mean that the focused sub-goal blocks the achievement of its parent goal. Analysts can give the different score for each edge in OR-decomposition while they must attach the same value to all of the edges in AND-decomposition because nothing but a complete set of the edges can contribute to the achievement of the parent goal. In the example of Fig. 1, the root goal "Web account system of high quality" has been refined into two sub-goals with AND-decomposition, and the contribution value +7 has been attached to them. Unless both of the sub-goals "Easy to register an account" and "Others do not register me" are achieved, their parent goal cannot be done. If both of them are achieved, the parent goal is achieved with the degree +7. In contrast, the goal "Others do not register me" is further decomposed into two sub-goals with OR-decomposition, and they have the values +6 and +9 respectively. If an analyst selects the goal "Authentication by SSH" to achieve the parent goal, he/she can obtain the contribution degree +9 higher than the other alternative.

Some notations for representing an attributed goal graph are formally defined as follows. We write the contribution value on the edge between a parent goal g_1 and its sub-goal g_2 as $c(g_1, g_2)$. The functions $pa(g)$ and $sub(g)$ respectively denote the sets of all the parent goals and the sub-goals of g. We also define two disjoint sets of parent goals of g as follows:

$$pa^+(g) = \{\, g' \mid g' \in pa(g) \wedge c(g', g) \geq 0 \,\},$$
$$pa^-(g) = \{\, g' \mid g' \in pa(g) \wedge c(g', g) < 0 \,\}.$$

$pa^+(g)$ and $pa^-(g)$ respectively express the parent goals that g positively contributes to and that g prevents.

3. Proposed Technique

Our technique supports the impact analysis to a goal graph when the structure of the graph is changed. We have two kinds of basic modification operations: 1) creating a new

element and 2) deleting an existing element. Any modifications on a goal graph can be represented as their combinations. In this section, we present what impact analysis can and should be considered for each type of basic modification operation, using simple examples.

We focus on the following two typical situations that may cause subsequent changes of a goal graph structure for addition and deletion of goals; 1) an addition of a new goal may cause a conflict with existing goals, and 2) a deletion of a goal may cause a damaging effect on the achievement of root goals. For detecting the former case, we use goal characteristics because they directly indicate the potential conflicts of two goals. For the latter case, we use contribution values. The contribution value-based analysis is suitable for measuring the impact when a goal is deleted because additions of goals do not decrease the achievability of root goals except for the introduction of edges having negative contribution values.

Note that requirements analysts can easily combine the proposed technique and the previous analysis techniques. For example, detecting discordances among stakeholders [8] may lead to structural changes of a goal graph (e.g., deletion of goals) in order to solve the discordance so that they can indirectly obtain the benefits of impact analysis after their subsequent structural changes. Additionally, analysts can apply the techniques of discordance detection and/or alternative evaluation [9] again after the structural changes of a goal graph.

3.1 Adding an Element

When adding a new goal, the resulting modification may cause a conflict between the existing goals and the newly added goal. Suppose that an analyst adds a new goal "Password Authentication" as shown in the left side of Fig. 2. The resulting goal graph shown in the right side of the figure may include a conflict to the existing goal "One can complete to register immediately", because generally requesting a user to input his/her password whenever he/she enters the account system reduces the easiness to use it. To detect this kind of conflicts, we need semantic information of goals. In this example, we used a general property that the improvement of security often causes the deterioration of usability. In this situation, we can consider *Security* and *Usability* as abstract semantic elements, which can represent the meaning of goals from a specific viewpoint. We call these elements *goal characteristics*. Therefore we have the categories of these goal characteristics in advance, and encourage analysts to attach the suitable elements of the goal characteristics to the goals whenever they newly create goals. Inference rules on goal characteristics can detect the possibility of conflicts between goals.

The next issue is what semantic categories as goal characteristics and inference rules on them we should prepare. In this paper, as an example, we focus on quality characteristics of ISO 9126 [11]. We have selected some of them, including the sub-characteristics, closely related to software requirements. Furthermore, we had a set of pairs of the selected characteristics that may often cause conflicts. The pairs are called *conflict pairs*. Suppose that an existing goal g_1 and a newly added goal g_2 respectively have goal characteristics c_1 and c_2. If the pair (c_1, c_2) is a conflict pair, it is suggested to an analyst that g_2 may cause a conflict to g_1. Although we happened to show the example of ISO 9126 characteristics, an analyst can define goal characteristics and their conflict pairs as he/she wants according to his/her aim and situation.

Figure 3 shows an example of the function of detecting conflicts. Our analyst defines six goal characteristics, Security, Performance, Usability, Resource Efficiency, Maintainability, and Portability, and a set of their conflict pair, e.g., (Security, Usability) or (Security, Performance). The analyst has attached *semantic tag* Usability, which represents the goal characteristic of Usability, to the goal "Easy to register an account". Semantic tags are shown in an attributed goal graph like stereotypes in UML class diagrams, such as ≪Usability≫. After adding a new goal "Password authentication", attaching ≪Security≫ as shown in the example, our technique suggests which existing goals can have conflicts to the new one. In the figure, the goal "Easy to register an account" may have a conflict to the new one.

Based on the detected conflicted pairs, the goals affected by the added goal are suggested to users. Users will analyze the relationship and trades-offs between the suggested goal and the added goal, and negotiate with stakeholders if necessary for each suggested goal. For estimating the cost of the analysis, consider that an analyst adds a new goal g to a goal graph having n goals. The number of the detected conflict pairs will be estimated at most $m \times n$,

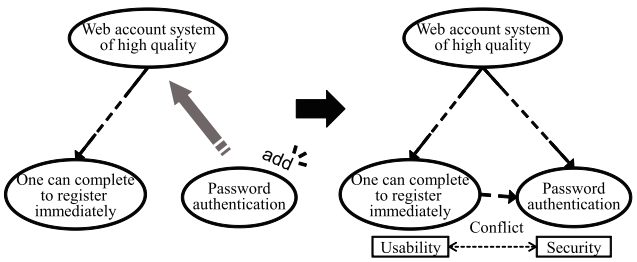


Fig. 2 Adding a goal.

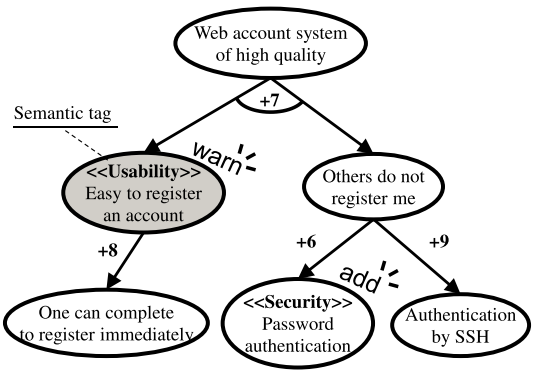


Fig. 3 Detecting goal conflicts.

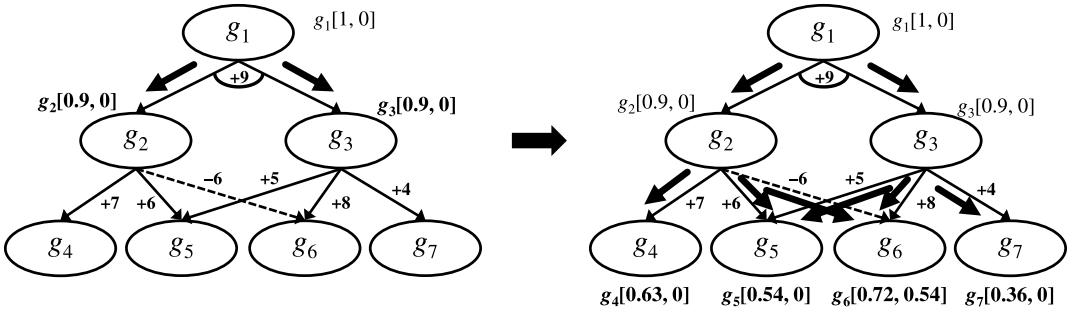


Fig. 4 Calculating achievement and obstruction degrees.

where m is the number of goal characteristics that conflict with the characteristics of g. Basically m is low because we mainly use well-known and limited characteristics such as ones appeared in ISO 9126. The number of goals in a goal graph n is estimated less than hundreds even if the graph is produced in a large industrial project [12]. Thus, we estimate that the number of the detected conflict pairs will be at most several hundreds, and the cost is then acceptable. Moreover, suppose a case that a user starts with a goal graph having only one root goal and then adds n goals at once in the next version. Even though this very extreme case, the estimated number of the suggested conflicts will be at most $m \times n^2$ and is still polynomial instead of exponential.

3.2 Deleting an Element

When deleting a goal from a goal graph, its effects are propagated to its parent goals. For example, if a sub-goal that is generated with AND-decomposition is deleted, its parent goal cannot be logically achieved any longer. In contrast, in the case of OR-decomposition, although the achievement of its parent goal is not changed, the degree of its achievement may be changed. Suppose that a goal g_1 has two sub-goals g_2 and g_3 in OR-decomposition, and their contribution values to g_1 are +10 and +1 respectively. Even if an analyst deletes either of g_2 or g_3, not both, g_1 is still achieved. However, if he/she deletes g_2 having the contribution value +10, the degree of achieving g_1 becomes lower because the remaining goal g_3 has low contribution value +1 to the achievement of g_1. In this case, it can be considered that the deletion of g_2 has the impact on the achievement of g_1 with the degree $10/(10 + 1) = 0.91$. We calculate an impact degree of a goal to the achievement of the root goal in a goal graph, using contribution values. Intuitively, the impact degree of a goal expresses the loss of the achievement of the root goal when it is deleted.

The technique to calculate an impact degree of a goal g to the root goal is as follows. First, we calculate two values called *achievement degree* and *obstruction degree* for g. We respectively write them as $ach(g)$ and $obs(g)$. They are defined as follows:

$$ach(g) = \max_{\substack{g_1 \in pa^+(g) \\ g_2 \in pa^-(g)}} \left\{ ach(g_1)\frac{c(g_1, g)}{10},\ -obs(g_2)\frac{c(g_2, g)}{10} \right\},$$

$$obs(g) = \max_{\substack{g_1 \in pa^+(g) \\ g_2 \in pa^-(g)}} \left\{ obs(g_1)\frac{c(g_1, g)}{10},\ -ach(g_2)\frac{c(g_2, g)}{10} \right\}.$$

The divisions of contribution values by 10 are for putting the values into the range from −1 to 1, because contribution values are attached with 10 scales. For a root goal g_R, which has no parents, we have $ach(g_R) = 1$ and $obs(g_R) = 0$ and start the calculation of the other goals following the above equations in top-down direction in the goal graph. Figure 4 illustrates how their calculation progresses. Here, the notation $g_i[a_i, o_i]$ in the figure means $ach(g_i) = a_i$ and $obs(g_i) = o_i$. Basically, the idea that a goal has a positive or negative value for its achievement is similar to the satisfiability and deniability of the approach by Giorgini *et al.* [13]. However, we adapted it to the calculation of impact degree.

Next, we calculate an impact degree using the above achievement and obstruction degrees. There are three cases of impacts of deleting a goal to the achievement of its parent goal, as follows.

The first case is the occurrences of logically unachievable goals (called *Upward Impact 1*). If one of sub-goals in AND-decomposition is deleted, its parent goal cannot be logically achieved. In the example of Fig. 1, when an analyst deletes the goal "Easy to register an account", its parent goal "Web account system of high quality" cannot be logically achieved any longer, because the goal is connected to its parent goal with AND-decomposition. However, in this example, its deletion leads the damage of the achievement of its parent with $7/10 = 0.7$ degree, based on the contribution value +7.

The second case is the decreasing of sub-goals to achieve its parent goal (called *Upward Impact 2*). Even if an analyst deletes one of the sub-goals that their parent goal is OR-decomposed to, the achievability of the parent goal holds as far as at least one sub-goal remains. However, if he/she deletes a sub-goal of higher contribution value, the achievability of its parent goal may decrease. Suppose that the analyst deletes the goal "Authentication by SSH" shown in Fig. 1. Its parent goal "Others do not register me" is still achievable because the sub-goal "Password Authentication" remains. We estimate the impact of deleting the goal of higher contribution as $9/(6 + 9) = 0.6$, by using a proportional distribution of the contribution values of its sub-goals, as mentioned in the beginning of this subsection.

The last case is the resolution of conflicts (called *Upward Impact 3*). If a goal having an edge whose contribution value is negative is deleted, its parent goal results in having none of the sub-goals that prevent its achievement. Suppose that an analyst deletes the goal g_6 in Fig. 4. For its parent goal g_2, the goal that blocks the achievement of g_2 is deleted, and this deletion allows to increasing the achievement degree of g_2. We can estimate the impact of this deletion as $(-6)/10 = -0.6$. The minus sign of the value means a good impact to the achievement of the parent goal. The example of this category will be shown later using Fig. 5.

To sum up, we define the impact degree $imp(g_1, g_2)$ of the goal g_2 to the root goal through its parent goal g_1 using the achievement and obstruction degrees of g_1, according to the above classification of the edge e between g_1 and g_2, as follows. For Upward Impact 1, if e is an AND-decomposition edge and has a positive contribution value, we have

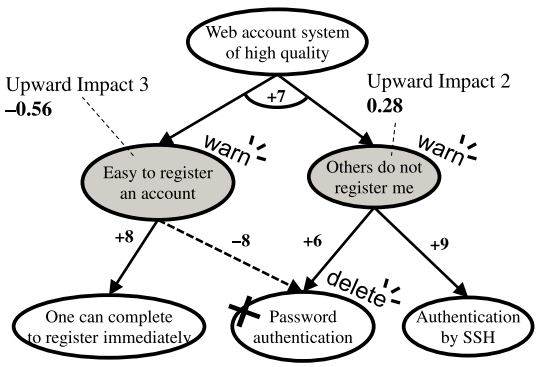
$$imp(g_1, g_2) = \{ach(g_1) - obs(g_1)\}\frac{c(g_1, g_2)}{10}.$$

For Upward Impact 2, if e is an OR-decomposition edge and has a positive contribution value, we have

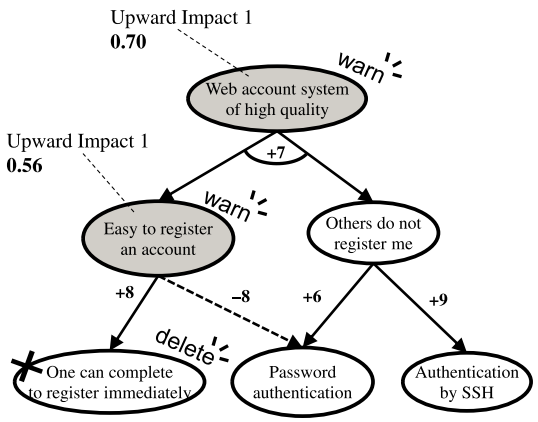
$$imp(g_1, g_2) = \{ach(g_1) - obs(g_2)\}\frac{c(g_1, g_2)}{\sum_{g \in sub(g_1)} c(g_1, g)}.$$

For Upward Impact 3, if e has a negative contribution value, as well as for Upward Impact 1, we have

$$imp(g_1, g_2) = \{ach(g_1) - obs(g_2)\}\frac{c(g_1, g_2)}{10}.$$



(a) Deleting a sub-goal having a negative edge in OR-decomposition.



(b) Deleting a sub-goal in AND-decomposition.

Fig. 5 Examples of calculating impact degrees.

Figure 5 shows two examples to calculate the impact degrees of a specified goal. Note that this example goal graph of a Web account system has a negative edge −8 between "Easy to register an account" and "Password authentication", unlike the example in Fig. 1. In Fig. 5 (a), our analyst selects the goal "Password authentication" to be deleted and calculates its impact degree to the root goal "Web account system of high quality". After the execution of the impact analysis, the analyst obtains two impact degrees: 1) −0.56 to the root goal through the parent goal "Easy to register an account" and its classification of Upward Impact 3, and 2) 0.28 through "Others do not register me" of the classification of Upward Impact 2. As for the first impact degree, the minus value shows a good impact to the achievement of the root goal because "Password authentication" obstructed the achievement of "Easy to register an account", and this obstruction is deleted. Since the sub-goal that the analyst selected for the deletion was in OR-decomposition from "Others do not register me", there are no other impacts to the upper goals on achievement view, and thus our tool stops the further analysis. In contrast, as shown in Fig. 5 (b), the analyst focuses on the goal "One can complete to register immediately". The impact analysis suggests the obstruction through its parent "Easy to register an account" to the root goal, because all of them are connected with AND-decomposition, i.e., Upward Impact 1.

4. Supporting Tool

We have implemented a supporting tool for our impact analysis technique. The tool is implemented as a plug-in of Eclipse. The impact analysis feature is integrated with the goal graph editor of the AGORA tool.

Figure 6 shows a screenshot of our tool. The graph editor has the abilities to attach attributes including semantic tags in addition to build goal graphs. Our tool can suggest the affected goals in contexts of adding or deleting a goal. Analysts can check the impacts of goals in two ways: 1) the color of nodes in a visualized goal graph and 2) a tabular form. After the impact analysis feature turns on, when an analyst selects a goal, the resulting impacts are immediately suggested to the analyst. In Fig. 6, after the selection of the goal "One can complete to register immediately" for deleting it, two upper goals are suggested to the analyst. The goals "Web account system of high quality" and "Easy to register an account" are colored with pink in the visualized graph. The impacts are also listed up in Impact tab sheet of Properties view, including their types, e.g., Upward Impact 1, and degrees. Using the graph view and Impact tab sheet, analysts can easily compare the effects of two or more goals because the analysis and notification are performed immediately after the selection of a goal.

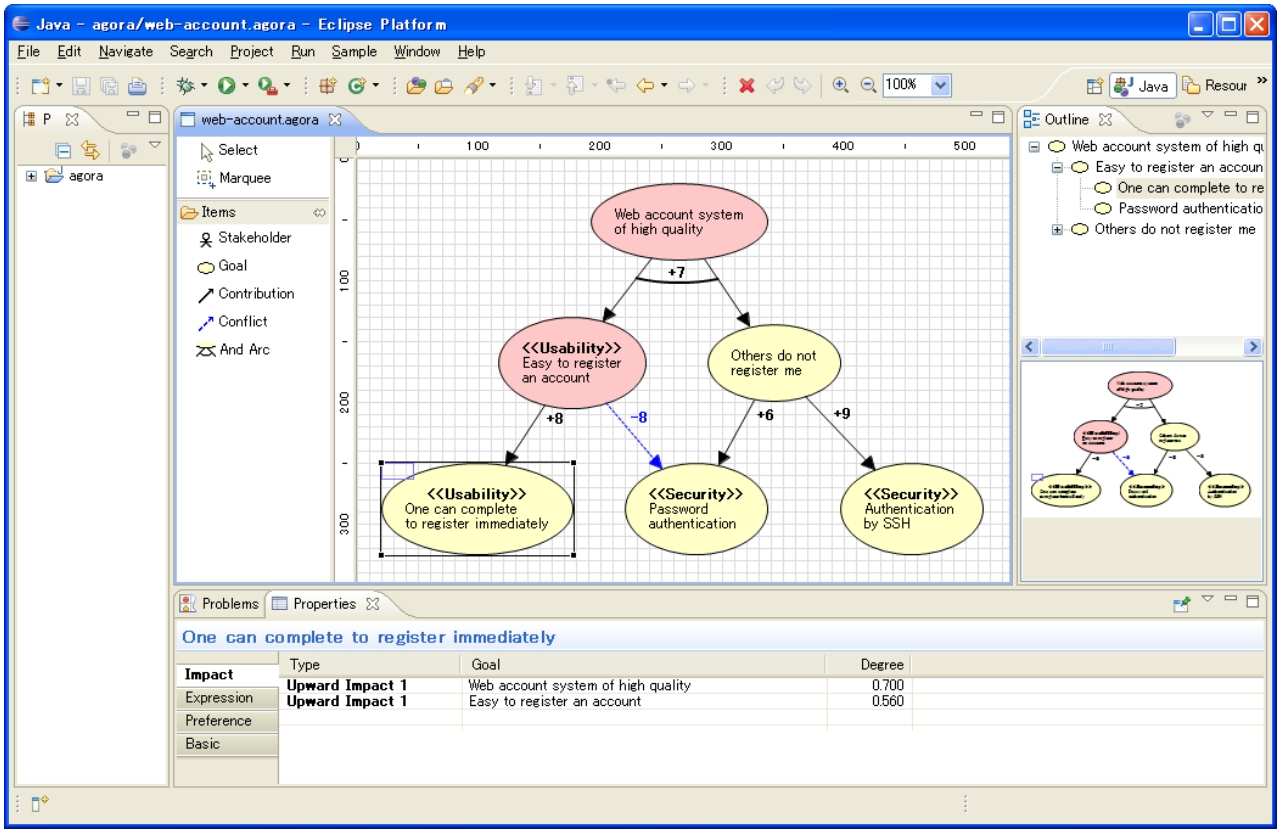


Fig. 6 Screenshot of the supporting tool.

5. Case Studies

In this section, we have a quantitative evaluation of our tool with respect to the ability for detecting conflicts and calculating impact degrees by using two case studies. The aims of our case studies are to check whether our technique can 1) detect all potentials of conflicts when a goal is added and 2) calculate impact degrees of alternatives so as to identify the alternative of more impact when either of two goals is alternatively deleted.

5.1 Case 1: A Seat Reservation System for Trains

Requirements for a seat reservation system for trains are elicited by using our goal-oriented method. First, 29 goals were defined as initial requirements, and each goal is related to several goal characteristics. The analyst used the goal characteristic Cost and a conflict pair (Security, Cost) in addition to (Security, Usability) and (Security, Performance) listed up in Sect. 3.1. Then, our customer requires the new goal "Safety monitor for administrators provided", in addition to the initial requirements, and the goal is added as shown in Fig. 7. The semantic tag ≪Security≫ is attached to this newly added goal.

Our tool tells us nine goals (colored in Fig. 7) could have conflicts with the goal. An expert checked whether these nine goals had really conflicts or not, and the expert decided that four of these nine goals annotated as *actual conflict* with downward triangles in Fig. 7 had actually conflicts to the newly added goal. In addition, the expert did not find conflicts no other than these four goals. Five out of the nine goals ($5 = 9 - 4$) had no conflicts to the added goal, according to the expert. Some of these five goals were the parent goals of the four actually conflict goals. As a result, there is no oversight in finding conflict goals. In other words, conflict goals are completely recalled. Although the precision was not so good ($4/9 = 44.4\%$), to us, it is more significant not to miss conflict goals, i.e., higher recalls, rather than getting higher precision with lower recalls. Therefore, our tool is good enough with respect to conflict detection.

5.2 Case 2: A Meeting Scheduler

Requirements for a meeting scheduler system were also elicited using a goal-oriented method. We used the goal graph shown in [14]. Following five goals were defined as initial requirements to the system: "Schedule meeting" "Minimal effort", "Good quality schedule", "Minimal disturbances", and "Accurate constraints". Based on these five goals, 19 goals are defined as temporary requirements for this system, where some conflict and/or alternative goals exist. Then, we attached attributes of AGORA method to the goal graph. The resulting goal graph is shown in Fig. 8.

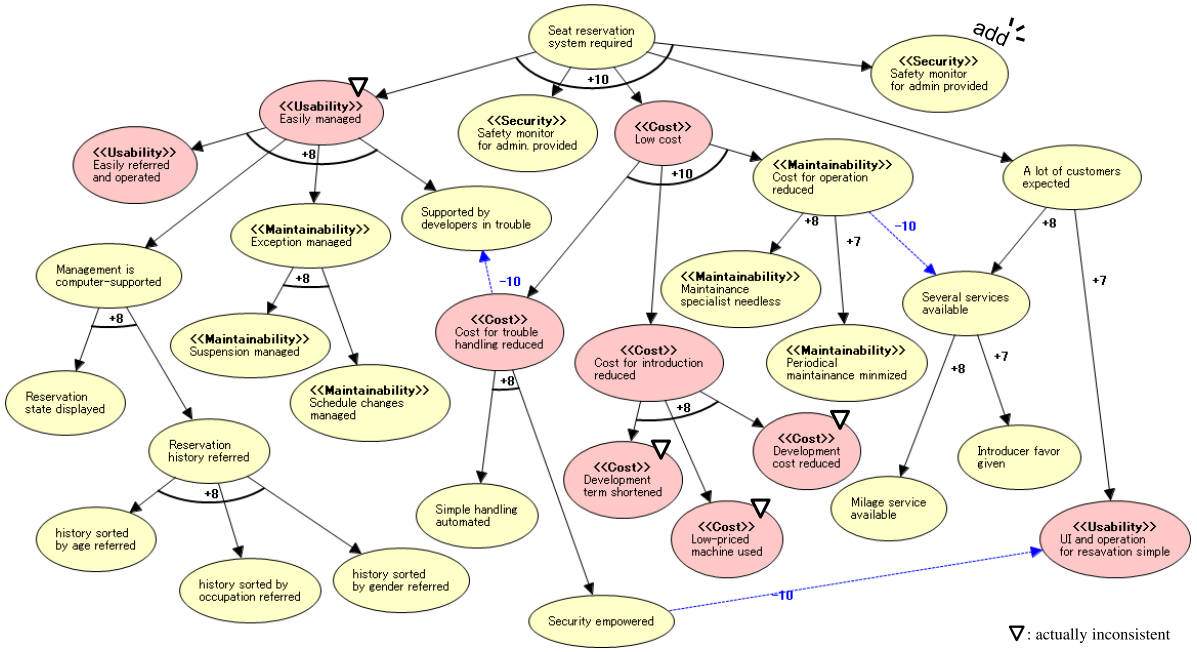


Fig. 7 Attributed goal graph of a seat reservation system.

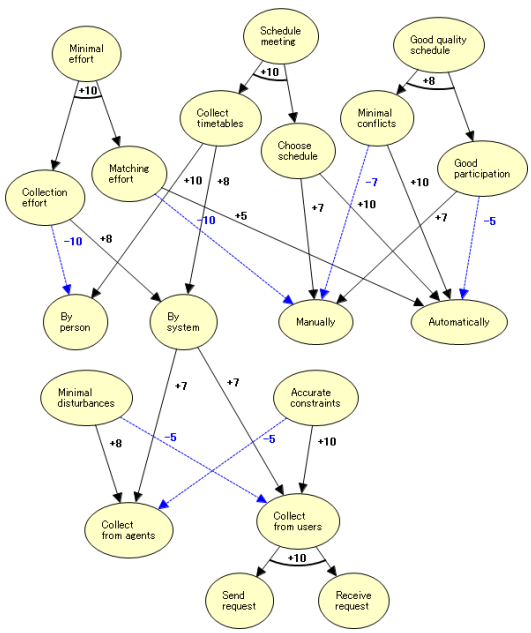


Fig. 8 Attributed goal graph of a meeting scheduler.

We investigate impacts to the root goal "Schedule meeting" by deleting following two goals for each: "(Schedule fixed) Automatically" and "(Schedule fixed) Manually". Table 1 shows the results of impact analysis. Our tool tells us the former goal gives more impacts rather than latter one by calculation results of their impact degrees. This result meets our intuition. Therefore, our tool is useful with respect to impact detection. This kind of impact analysis helps us to make a decision of choosing one from alternatives. Suppose that we should delete either of the above two goals,

Table 1 Suggested upward impacts.

Target	Goal	Type	Impact
Automatically	Choose schedule	Upward Impact 2	0.588
	Good quality schedule	Upward Impact 1	0.8
	Minimal conflicts	Upward Impact 1	0.5
	Good participation	Upward Impact 3	−0.4
	Matching effort	Upward Impact 1	1.0
	Minimal effort	Upward Impact 1	0.8
Manually	Choose schedule	Upward Impact 2	0.412
	Good quality schedule	Upward Impact 1	0.8
	Minimal conflicts	Upward Impact 3	−0.56
	Good participation	Upward Impact 1	0.56
	Matching effort	Upward Impact 3	−1.0

due to reducing development cost. Since the impact degree of "Automatically" is larger to the root goal, it would be better that we would leave it remained and delete the other, i.e., "Manually". Our tool also suggests to us how to change the structure of a goal model so as to minimize bad impacts when some goals are deleted.

6. Related Work

The techniques of impact analysis on requirements are not so popular in contrast to that on source code. In industry, requirements are usually managed and traced using general spreadsheets with retrieval functions by means of attaching attribute values to each requirement, e.g., identification numbers, dates, version number, or authors. Process improvement discipline such as standard process and/or procedure for changes is largely focused.

Quality attributes for each requirement are also used for finding conflicts among requirements [15]. In this research, quality attributes such as efficiency and reliability

are put for each requirement, and by using predefined matrices of attributes, conflicts among requirements are detected. This approach is quite similar to our approach, but they do not apply their approach to goal models yet. We can find another type of interesting contribution on metrics related to the co-evolution of a business process and the information systems that support it [16]. By using the proposed technique, we can measure the misalignment of its information systems when the business process is changed. We can apply this technique to measure stable parts and volatile ones on modifying a goal graph.

Rather than impact analysis, detecting and recovering traceability links between requirements documents and software artifacts are also important. If we can establish traceability links from a requirements document to a destination artifact, and the requirement document is changed, we can be helped to propagate the change to the destination. Jan et al. [17] focused on forward traceability of non-functional requirements, but they do not focus on traceability among requirements. For example, traceability links from a requirements specification to its architectural design are recovered and managed semi-automatically. They generate a probabilistic model based on the frequency of the terms appearing in each document, and traceability links are generated based on the model. This approach can be used for our approach if there is enough documentation for each goal. They also presented an excellent survey and comparison for the techniques to detect traceability links [18].

Contrary to our approach, there are some research topics for analyzing the histories of requirements evolution. For example, the PRINCE model [19] provides a model for requirements evolution focusing on the maturation patterns of requirements elicitation processes. Their aim differs from ours, because our approach focuses a snapshot of a goal graph and its change, and provides the degrees of the change impact.

7. Conclusion and Future Work

This paper presented an impact analysis technique of adding and deleting goals in an attributed goal graph for requirements changes in goal-oriented analysis. Furthermore, we have implemented a supporting tool based on our proposed techniques and assessed it through case studies.

Future work can be listed up as follows.

- We represent a conflict with a binary relation, e.g., a pair of Security and Usability. In more practical setting, more complicated representation of conflicts, e.g., conflicts in a specific context and ternary or more conflict relation, may be required.
- As growth of a goal graph, the technique to construct the goal graph together with attributes becomes a significant problem. The techniques of hierarchical decomposition and grouping of a large goal graph can be considered, and we can attach attribute values to representative goals only and propagate automatically them to the other child goals. The construction of a large goal graph is a common issue in a family of goal-oriented approaches, which should be tackled with in future.
- Our case studies mentioned in Sect. 5 were limited. To argue the generality of our findings, more case studies covering various domains are necessary. In particular, to avoid missing the detection of conflicts, the high recall value of detection should be guaranteed. In addition to refining the category of goal characteristics and conflict representation, the methodological support to attach a goal characteristic to a goal correctly is also necessary.
- Our approach of detecting conflicts is based on explicit dependency among requirements. Thus we can apply it to the other modeling techniques where make requirement dependency explicitly represented, such as the order of tasks [20]. Showing its wide applicability is one of the future issues.
- In our approach, we respectively applied characteristic-based and contribution value-based analyses for checking impacts only when a goal is respectively added and deleted. Although the approach can deal with two typical situations of goal graph modifications, other possibilities still remain. Coping with some other possibilities, e.g., addition of a goal with an edge having negative contribution value, can be a possible future work. Additionally, other types of impacts, e.g., impacts to attributes rather than the structure of goal graphs, are also needed for consideration.
- In AGORA methodology, we can know which goals are selected for achieving the root goals and the others are not by evaluating alternatives of requirements [9]. When detecting conflicts between existing goals and an added goal, we can consider that the conflicts with selected goals are more important than those with non-selected goals. Emphasizing the resulting conflict pairs according to the selection of alternatives is one of the future issues.

Acknowledgements

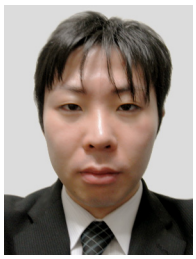
The authors would like to thank Mr. Kohei Uno, Mr. Kinji Akemine, and Mr. Takashi Yoshikawa for implementing the initial version of our tool.

References

[1] D. Tanabe, K. Uno, K. Akemine, T. Yoshikawa, H. Kaiya, and M. Saeki, "Supporting requirements change management in goal oriented analysis," Proc. 16th IEEE International Requirements Engineering Conference (RE'08), pp.3–12, 2008.
[2] E. Yu, "Towards modeling and reasoning support for early-phase requirements engineering," Proc. 3rd IEEE International Symposium on Requirements Engineering (RE'97), pp.226–235, 1997.
[3] C. Rolland, C. Souveyet, and C. Ben Achour, "Guiding goal modeling using scenarios," IEEE Trans. Softw. Eng., vol.24, no.12, pp.1055–1071, 1998.

[4] "IEEE recommended practice for software requirements specifications," tech. rep., IEEE Std. 830-1998, 1998.
[5] A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.
[6] "Objectiver: Sitehomepage." available at http://www.objectiver.com/
[7] H. Kaiya, H. Horai, and M. Saeki, "AGORA: Attributed goal-oriented requirements analysis method," Proc. IEEE Joint International Requirements Engineering Conference (RE'02), pp.13–22, 2002.
[8] H. Kaiya, D. Shinbara, J. Kawano, and M. Saeki, "Improving the detection of requirements discordances among stakeholders," Requirements Engineering, vol.10, no.4, pp.289–303, 2005.
[9] K. Yamamoto and M. Saeki, "Attributed goal-oriented analysis method for selecting alternatives of software requirements," IEICE Trans. Inf. & Syst., vol.E91-D, no.4, pp.921–932, April 2008.
[10] M. Saeki, S. Hayashi, and H. Kaiya, "A tool for attributed goal-oriented requirements analysis," Proc. 24th IEEE/ACM International Conference on Automated Software Engineering (ASE'09), pp.670–672, 2009.
[11] ISO 9126: "Information Technology – Software product evaluation – Quality characteristics and guidelines for their use," 1991.
[12] A. van Lamsweerde, "Goal-oriented requirements enginering: A roundtrip from research to practice," Proc. 12th IEEE International Requirements Engineering Conference (RE'04), pp.4–7, 2004.
[13] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models," Proc. 21st International Conference on Conceptual Modeling (ER'02), LNCS 2503, pp.167–181, 2002.
[14] J. Mylopoulos, "Goal-oriented requirements engineering, Part II," 2006. Presented at 14th IEEE International Requiements Engineering Conference (RE'06). available at http://www.ifi.uzh.ch/req/events/RE06/ConferenceProgram/RE06_slides_Mylopoulos.pdf
[15] A. Egyed and P. Grunbacher, "Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help," IEEE Softw., vol.21, no.6, pp.50–58, 2004.
[16] T. Bodhuin, R. Esposito, C. Pacelli, and M. Tortorella, "Impact analysis for supporting the co-evolution of business processes and supporting software systems," Workshop Proc. 16th International Conference on Advanced Information Systems Engineering (CAiSE'04), pp.146–150, 2004.
[17] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina, "Goal-centric traceability for managing non-functional requirements," Proc. 27th International Conference on Software Engineering (ICSE'05), pp.362–371, 2005.
[18] J. Cleland-Huang, "Toward improved traceability of non-functional requirements," Proc. 3rd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'05), pp.14–19, 2005.
[19] T. Nakatani, S. Hori, M. Tsuda, M. Inoki, K. Katamine, and M. Hashimoto, "Towards a strategic requirements elicitation — A proposal of the PRINCE model," Proc. 4th International Conference on Software and Data Technologies (ICSOFT'09), pp.145–150, 2009.
[20] S. Liaskos, S.A. McIlraith, S. Sohrabi, and J. Mylopoulos, "Integrating preferences into goal models for requirements engineering," Proc. 18th IEEE International Requirements Engineering Conference (RE'10), pp.135–144, 2010.

Shinpei Hayashi received a BE degree in information engineering from Hokkaido University in 2004. He also respectively recieved ME and DE degrees in computer science from Tokyo Institute of Technology in 2006 and 2008. He is currently an assistant professor of computer science at Tokyo Institute of Technology. His research interests include software changes and software development environment. He is the maintainer of the current AGORA tool.



Daisuke Tanabe received a BE degree in computer science from Tokyo Institute of Technology in 2007. He is currently working at AGREX INC. His research interests include requirements engineering and change impact analysis.

Haruhiko Kaiya is an associate professor of software engineering in Shinshu University. He is also a visiting associate professor in National Institute of Informatics (NII).

Motoshi Saeki respectively received a BE degree in electrical and electronic engineering, and ME and DE degrees in computer science from Tokyo Institute of Technology, in 1978, 1980, and 1983. 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).