An Optimizing Algorithm for Modifying a Goal Graph

Wen-Kui Chang [1], Shih-Peng Su[2] and Fuw-Yi Yang[3]

[1] Dept. of Computer Sciences & Information Engineering, Tunghai University, Taichung, Taiwan 40767
wkc@thu.edu.tw
[2] Library of Chien Kuo Technology University, Taiwan
lib6@ctu.edu.tw
[3] Dept. of Electronic Engineering, Chien Kuo Technology University, Changhua, Taiwan
flyyou@ctu.edu.tw

Abstract

This research presents the application of the Goal Graph in the methodology of generating software requirements specification, in order to improve the modifiability of the customization the Goal Graph. We propose an algorithm that optimizes the quality factor of modifiability for the Goal Graph to enhance the customization of the Goal Graph. Besides, we demonstrate the real application of the proposed algorithm. An empirical application for generating the customized requirements specification for a practical library project is also presented. In addition, using two attributes, priority-values and cost-values, we further build the selection factors which as a screening criterion. The Goal Graph, having attribute values and their weights, can generate a customized requirements specification.

Keywords: Goal-Oriented Analysis, Modifiability, Quality Metrics, Tree Structure.

1. Introduction

In the life cycle of software development, the requirements specification is usually changeable and incompletely defined. As the result, the subsequent phases such as system design, implementation, testing etc., exist lots of uncertainty to face with the possible changes in the later time of a project. Therefore, how to confront the continuous changes of customer requirements more effectively and efficiently as well plays an important role for performing a successful project.

In this research, we explore the goal-oriented approach to formulate a more widely applicable specification-generation process by extending the Goal-Oriented Requirements Analysis method (GORA) [Dardenne, 1993; Anton and Potts, 1998] to support the task of requirements elicitation. A high-quality requirement specification will contribute to customer satisfaction. Literature such as: Davis [Davis, 1993] and IEEE std-830 [IEEE Computer Society, 1998] analyze quality of requirement specification. In particular, Kaiya et al. proposes a method of Attribute Goal-Oriented Requirement Analysis (AGORA) with attribute values to provide quantitative analysis on goals.

In the following, after discussing related studies, we depict, in Section 3, a case study from a realistic library-loan system for analyzing quality of a goal graph denoted by the AGORA. In Section 4, we propose an algorithm to optimize the quality factor of modifiability for a goal graph to enhance its customization. A demonstration of applying the proposed algorithm on the concerned case study is also illustrated.

2. Related Studies

Several researchers or institutes such as Mylopoulos[Mylopoulos, et al. 1999], Lamsweerde [Van Lamsweerde, 2001], Oshiro [Oshiro, et al. 2003], etc., have proposed the Goal-Oriented Requirement Analysis method to support the task of requirements elicitation. In principle, the method starts to refine the customers requirements into goals, and then represent them in a goal graph using the AND-OR elements, which may be used to identify any conflicts among these initial goals and analyze the corresponding impacts.

Several analysts analyze the potential impacts when requirements are changed and provide quantitative analysis techniques on goals. Kaiya et al. propose a method of Attribute Goal-Oriented Requirement Analysis (AGORA) with attribute values. In fact, AGORA is an extended version of the Goal-Oriented requirement analysis method. The attribute values can help the procedure of requirements analysis in detecting and resolving these possible conflicts

among the goals. The attribute and structure of a goal graph can help analysts to evaluate quality factors of requirements specification, such as correctness, unambiguity, completeness, etc. But above research are not consider how to improve the quality factors of modifiability. This paper extends the implied framework and proposes an algorithm to determine quality factors of modifiability in a goal graph to enhance its customization.

3. Quality Analysis of the Goal Graph

3.1. An illustrated case of the Goal Graph

For the demonstration purpose, a subsystem is sampling from a maintaining project of our university library-loan system by revising some user requirements. The librarian who hopes to improve the rate of the return books intend to implement more real time of "*overdue note subsystem*" to substitute for the traditional written notice. The analyst who wants to identify the requirement specification of the subsystem employs the method of Goal-Oriented Requirement Analysis to first construct a goal graph as shown in Fig. 1.

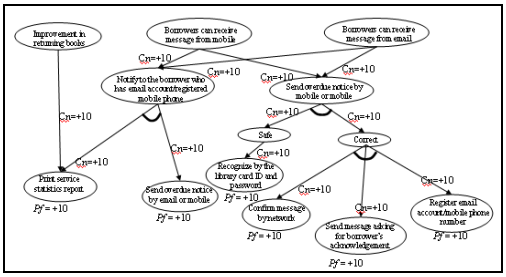

Fig. 1. The refined Goal Graph.

Analysts and librarians discuss and identify the three initial goals: "*Improvement in returning books*," "*Borrowers can receive message from mobile*" and "*Borrowers can receive message from email*," which are put on the root of the goal graph. Then they adopt the Goal-Oriented Requirement Analysis to decompose initial goals from top to bottom, step by step. Finally, they get six final goals that are used as the requirement specification which has the concrete operation description as illustrated in Figure 1.

3.2. Quality measurements for requirements specification

Davis [Davis, 1993] and IEEE std-830 [IEEE Computer Society, 1998] provide a list of quality factors such as: correctness, unambiguity, completeness, consistency, verifiability, modifiability, traceability, etc. and ranks them as the most important in the requirements specification. Later, Kaiya et al. apply the measurement method to analyze these quality factors. For each requirements analysis method, the goal graph proposes specific quality metrics of requirements specification, which are then used to describe various quality factors of requirement specification [Kaiya, et al. 2002].

For the sake of deliberate discussion in the following, we focus on the quality factor of the modifiability of the goal graph. By the definition of IEEE std-830, "modifiability" means the structure and style of requirements documents are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style. Therefore Kaiya and Mario etc. lay claim to the fact that modifiability included in a goal graph is closed to a tree structure [Kaiya, et al. 2002; Alejandra and Mario, 2004]. When there are many incoming edges to a goal, the goal contributes to an achievement of many goals. In consequence, these incoming goals should be first considered while we change the goal in the future. Kaiya defines the symbols for measuring the goal graph produced by AGORA as follows:

- *incoming_edges(g,e)* denotes that edge *e* comes in the goal *g*.
- *RefinedGoals* are derived goals from the initial goals in the goal graph.
- *InitialGoals* are considered as the needs of the customers, and at first an analyst puts them on the root nodes of the goal graph.
- *Goals* are the total goals in the goal graph.

Expressed precisely, the formal definition of modifiability may be computed as follows:

$$Modifiability = \frac{\#\{g \in RefinedGoal \mid \#\{e \mid incoming\_edge(g,e)\} = 1\}}{\#\,RefinalGoal} \quad \cdots \quad (1)$$

$$RefinedGoals = Goals - InitialGoals \quad \cdots \quad (2)$$

As an example, the value of modifiability in Fig. 1 is calculated as:

$RefinedGoals$ = 13-3 = 10

$Modifiability$ = (10-3)/10 = 0.7

4. Improving Modifiability of a Goal Graph

4.1. The algorithm for improving the modifiability of the Goal Graph

As mentioned above, for optimizing the modifiability of a goal graph, its structure must be transformed to the one same as to a complete tree in which each node has only one incoming edge. Therefore, we propose an algorithm for extending the goal graph to a complete tree. We adopt the method of Post-order Traversal to identify any node that is with two or more incoming edges. We will extend the goal graph by copying and adopting the identified node. Therefore we define the *copied node* and the *genuine*

node. If a node has two or more incoming edges, we repeat the following steps until all nodes have only one incoming edge:

1. Copying (incoming edges -1) nodes of node together with its sub-trees,
2. Removing all incoming edges of *copied nodes*
3. Redistributing all incoming edges of *genuine node* to all *nodes* (*genuine node* and *copied nodes*) such that each node with one incoming edge.

As an illustration, since there are three incoming edges for the node D in Fig. 2, we first copy the node D to two new sets of the node D_1, D_2 together with their sub-trees. Next step, we remove all incoming edges of the *copied node* D, and redistribute all incoming edges of nodes D, D_1 and D_2 to the node A, B and C.

If expressed in a structured-statement manner, we may adopt the following algorithm to optimize the modifiability of a goal graph:

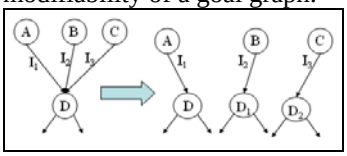

Fig. 2. Redistributing all incoming edges of a *copied node*

```
For node = Post-order traverse from leaf_node to root_node
    If 1 < node (incoming_edges)
            Copy (incoming_edges -1) nodes of node
            together with its sub-trees
            Remove all incoming edges of genuine node
            Redistribute all incoming edges of genuine node
            to all nodes (genuine node and copied nodes)
            such that each node with one incoming edge
    Endif
Endfor
```

4.2. Complexity of the proposed algorithm

In this subsection, we may derive the complexity of the modifiability algorithm is in the order of n^2, as shown in the following proof.

Theorem: The complexity of the modifiability algorithm is $O(n^2)$.

Proof:

Assuming that there are "*n*" nodes arranged in *3* levels, each level containing at most "*m*" nodes. It is clear that the incoming edges to both the root node (the fist level) and these nodes at the second level are no more than one edge. Thus the number of incoming edges of the nodes at the third level is at most *m*.

Let *Derived_tree_nodes* denote the sum of *genuine nodes* and *copied nodes*.

$$Derived\_tree\_nodes\ (m) = (n - m - 1)\bullet m + (m + 1) = 1 + n\bullet m - m^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The maximal quantity of *Derived_tree_nodes* is then ($n^2 / 4 + 1$) and occurs when $m = n / 2$ as shown in Fig. 3.

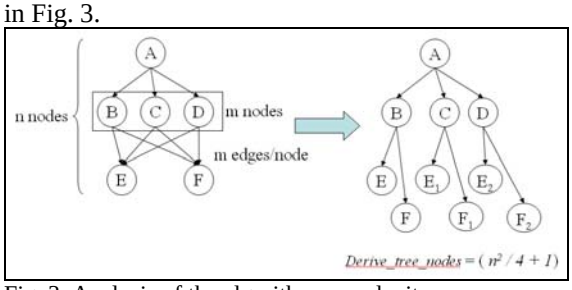

Fig. 3. Analysis of the algorithm complexity

Now, if there are "*n*" node in the goal graph then we amounting ($1 + n / 2$) nodes to locate at the first level and second level. We repeat the above process for the third level, fourth level, and so on. Thus the following equation is obtained.

$$Derived\_tree\_nodes < (n^2/4+1) + ((n/2)^2/4+1) + ((n/4)^2/4+1) + \ldots + 1$$

$$\leq log_2(n) + (n^2/4)[1 + (1/2)^2 + (1/2)^3 + \ldots + (1/2)^{2\cdot(\log_2(n)-1)}] \quad \ldots.(4)$$

$$\leq log_2(n) + (n^2/3) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$Derived\_tree\_nodes \in O(n^2)$$

From the above formula, we can conclude that this algorithm may reach the astringency. That is, the complexity of the modifiability algorithm is $O(n^2)$. *Q.E.D.*

4.3. Empirical demonstration

To optimize modifiability of a goal graph, we apply the modifiability algorithm presented in the previous section to re-adjust its structure. As in the case study of a library project introduced in Section 3.1, the goal "*Print service statistics report*" has two incoming edges. We then copy it to generate to two new goals "*Print service statistics report*" and "*Print email and mobile service statistics report.*" Then we remove all of the incoming edges on the goal "*Print service statistics report*" and then redistribute all incoming edges. Thus, the incoming edge of the goal "*Print service statistics report*" connects to its parent goal "*Improvement in returning books*" and the incoming edge of goal "*Print email and mobile service statistics report*" connects to the parent goal "*Notify to the borrower who has email account/registered mobile phone*."

Finally, the sub-tree root of "*Web register for email account or mobile phone number*" is further copied to two new sub-trees of "*Web register for mobile phone number*" root and "*Web register for email account*" root. Then we remove all of the incoming edges, and redistribute all incoming edges. Incoming edge of the sub-tree "*Web register for email account*" connects to the parent goal "*Borrowers can receive message from email*" and incoming edge of sub-tree "*Web register for mobile phone number*" connects to parent goal "*Borrowers can receive message from mobile.*" Through the previous process we get three independent trees as shown in Fig. 4. In

all, there are twelve refined command-style statements that constitute the preliminary goals for the demonstration example. Note that these goals are described starting by a predicate verb such as: *Print*, *Send*, *Confirm*, etc.

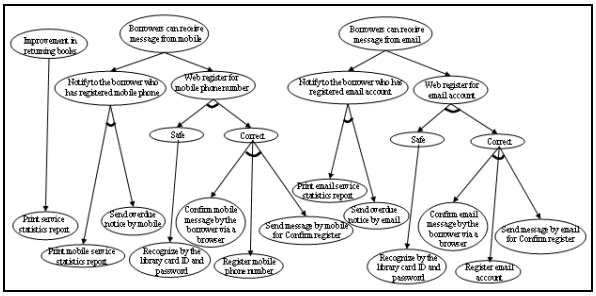

Fig. 4. The resultant tree structure of the Goal Graph

5. Conclusion

The requirements engineering process plays a vital stage in a project development. In this research, the goal-oriented analysis is investigated to solve the issue encountered in the effort of requirements elicitation. We adopt the process flow of AGORA to generate preliminary goals and then produce the derived goals with our proposed algorithm for optimizing the modifiability of the goal graph. In this paper, we propose an algorithm to improve the modifiability of the goal graph. The suggested algorithm can optimize the modifiability of the goal graph and help the requirements of change in the future. We only have to consider the relation around the goals, and do not need to consider other possibility goal cause changes of influence.

For a large system, our approach establishes a big goal graph. We recommend that sub-systems be divided under the system to construct the goal graph for each sub-system.

In the future, we'll explore how to further resolve the dependency problem that we likely encounters among some preliminary goals when applying the screening factor. In addition, quality analysis using common quality metrics such as verifiability, usability on the generated specification is the next task in this research.

6. References

[1] Alejandra C., and Mario P., "Challenges Setting a Process to Manage COTS Component Selection" *International Workshop on Models and Processes for the Evaluation of COTS Components*, (MPEC'04) 25 May, 2004.

[2] Anton, A. "Goal Identification and Refinement in the Specification of Software-Based Information Systems," *Ph.D. thesis, Department of Computer Science, Georgia Institute of Technology*, June 1997.

[3] Bourque, Pierre and Robert Dupuis, "Guide to the Software Engineering Body of Knowledge," *2004 Version, IEEE Computer Society*, 2004.

[4] Dardenne, A., Van Lamsweerde, A. and Fickas, S."Goal-directed Requirements Acquisition." *Science of Computer Programming 20*, pp. 3–50 1993.

[5] Davis, A., "Software Requirements: Objects, Functions, and States (Second Edition)," *Englewood Cliffs, New Jersey: Prentice-Hall*, ISBN: 0-13-562174-7, 1993.

[6] Hui Bowen, Liaskos Sotirios and Mylopoulos John, "Requirements Analysis for Customizable Software: A Goals Skills Preferences Framework" *John Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pp. 117-126, 2003.

[7] IEEE "Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998, IEEE*, New York, 1998.

[8] Kaiya, H., Horai, H., and Saeki, M. "AGORA: Attributed Goal-Oriented Requirements Analysis Method. " *In Proc. of the 10th IEEE International Requirements Engineering Conference (RE'02)*, pp. 13–22, 2002.

[9] Mylopoulos, J., Chung, L. and Yu, E. "From Object-Oriented to Goal-Oriented Requirements Analysis." *Communications of the ACM*, 42(1), pp. 31–37, 1999.

[10] Oshiro, K.; Watahiki, K., and Saeki, M . "Goal-oriented idea generation method for requirements elicitation" *Requirements Engineering Conference, 2003. Proceedings.* 11th IEEE International, 8-12 Sept, (2003).

[11] Shinbara, D., Kawano, J., Kaiya, H., and Saeki, M. "Identifying Requirements Gaps among Stakeholders by using Goal Oriented Analysis." *in REFSQ04 Post Workshop Proceedings, Riga*, Latvia 7 - 11 June, pp. 219-234, 2004.

[12] Sutcliffe, A., "Scenario-Based Requirements Analysis." *Requirements Engineering 3*, pp. 48–65, 1998.

[13] Takeda, N., Shiomi, A., Kawai, K. and Ohiwa, H., "Requirements Analysis by the KJ Editor." *In Proc. of 1st IEEE International Symposium on Requirements Engineering*, (RE'93), pp. 98–101, 1993.

[14] Van Lamsweerde, A., "Goal-oriented requirements engineering: a guided tour." *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, 27-31 Aug, pp. 249 – 262, 2001.