A Web-Based Requirements Analysis Tool

Annie I. Antón
Eugene Liang
Roy A. Rodenstein
{anton,eugene,royrod@cc.gatech.edu}
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract

The Goal Based Requirements Analysis Tool (GBRAT) is designed to support goal-based requirements analysis. The tool provides procedural support for the identification, elaboration, refinement and organization of goals to specify the requirements for software based information systems. GBRAT employs interactive Web browser technology to support the collaborative nature of requirements engineering.

1 Introduction

The World-Wide-Web (WWW) has emerged in recent years as a standard medium to display information. The ability to support the collaborative nature of requirements engineering using interactive WWW technologies led us to develop our Web-based Goal-Based Requirements Analysis Tool (GBRAT). Using GBRAT, project members can work collaboratively to specify goals for software systems. The specified goals can be viewed and modified by other project members located anywhere around the world.

This paper discusses some aspects of GBRAT in the context of its use in a real study involving the specification of requirements for Web-based applications. The tool is useful to identify, elaborate and organize goals for requirements specification. Section 2 provides a brief overview of our previous work. An overview of the tool and its use on a real study are provided in Section 3. A summary and conclusions are discussed in Section 4.

2 Previous Work

In a previous paper [1] we cite the need for strategies for the initial identification and construction of goals. We discussed goals from the perspective of two themes: goal analysis and goal evolution. That paper provides an overview of our Goal-Based Requirements Analysis Method (GBRAM) and summarizes our experiences in applying the method to a relatively large example. Our experiences with scenario analysis [4] demonstrated that scenarios are useful for uncovering and elaborating requirements, and for answering questions that are not easily answered using other techniques. These studies prompted us to further develop and validate our strategies to identify and construct goals. We are currently prototyping GBRAT to support the process of identifying and capturing goals, responsible agents, stakeholders, constraints, goal obstacles and scenarios as well as for specifying relationships between goals and subgoals. We are applying the method (GBRAM) and tool (GBRAT) to electronic commerce applications as we discuss in this paper.

Goal Analysis

Goals are high level objectives of the business, organization or desired system. They are a logical mechanism for identifying, organizing and justifying software requirements [1]. Using the Goal-Based Requirements Analysis Method [1], we identified the functional modules for GBRAT and determined the functional requirements by operationalizing the goals (Operationalization is the process of defining a goal with enough detail so that its subgoals have an operational or functional definition). The result of this analysis is best observed in the GBRAT prototype.

Current goal-based methods do not provide analysts with sufficient strategies for knowing how to initially identify goals or how to extract goals from the available information sources. Furthermore, tool support is lacking to support these methods. The objective of the underlying research is to develop a catalog of heuristics and questions to guide analysts as they identify and specify system and enterprise goals. The objective of GBRAT is to provide analysts with the procedural support they need to be able to analyze and refine goals. GBRAT will support and guide analysts as they identify, capture and structure require-

ments information in the form of goals.

3 GBRAT

GBRAT supports goal-based requirements analysis. The tool serves as a medium for project team members, working from different locations, to participate in the decision-making processes which permeate requirements engineering. Team members are able to work collaboratively on new ideas, discuss issues, and make decisions about system goals despite their geographic and time differences.

3.1 Users

The typical GBRAT user is an experienced requirements engineer with a considerable working understanding of the goal-based method, the WWW and Web-based applications. We assume that GBRAT users will work from existing diagrams, textual statements of need and/or additional sources of information, such as transcripts of interviews with stakeholders to identify and specify the goals of the desired system. After the analyst/elicitor has gathered all available information about the desired system he or she can then extract goals from these information sources and specify them using GBRAT as described below.

3.2 System Features

GBRAT features enable users to create project repositories and, specify goals, view goals from several perspectives and order goals. The examples provided in this paper to illustrate these features are part of an ongoing requirements reengineering of a Web server that supports various consortium member organizations participating in electronic commerce. The Web server must support secure payment and transactions, different access levels, membership and seminar registrations, as well as project and proposal status tracking. Several examples from this study are employed to demonstrate how GBRAT enables us to easily identify synonymous goals and manage traceability via the Web. Although, we are currently using GBRAT to establish the requirements for Web-based applications, its use is certainly not limited to this kind of system. It can be used for a more general range of information systems involving multiple stakeholders.

The method (GBRAM), though not the tool (GBRAT), has been used on a range of problems, including a business process reenginering project for an Air Force Base [1] and the requirements specification for GBRAT [2] [3].

Project Repositories

Goals concerning a given system are stored in a project repository. Each project repository has a specified project name and description as well as the name of the analysts working on a given project. From within a specific project repository, we can create new goals or view the previously specified goals using three filters: the maintenance and achievement goal filter, the agent filter, and the total order filter. The following sections discuss how goals are created and how the ability to view the specified goals via the different available filters is helpful to analysts.

Figure 1: Project Repositories

Creating Goals

Goal creation requires users to complete a form, as shown in Figure 2, to specify the goal name, classification and responsible agent(s). Users must specify a goal name, such as, "`AVOID duplicate purchase`" as shown in Figure 2. Goals are named in a standardized subset of natural language in which the first word is a verb that describes the kind of goal being named. For example, `AVOID` denotes one kind of goal. Goals of this kind are satisfied for as long as their target conditions remain false. Each goal is classified either as an achievement or as a maintenance goal. Achievement goals are objectives of the system and are named by the verbs `MAKE` and `KNOW`. For example, a seminar registration system may need to satisfy the goal of enrolling consortium members in the seminar before the actual seminar begins. The object of the goal is seminar registration and so the goal would be named `MAKE member registered`. Maintenance goals are those goals that are satisfied while their target condition remains true and are therefore named using the verbs `MAINTAIN`, `KEEP`, `AVOID` and `ENSURE`. They tend to be operationalized as actions that prevent certain states from being reached. The goals in Table 1 are examples of maintenance goals.

Figure 2: GBRAT Form to Create Goals

Goal	Classification	Agent
ENSURE secure transaction	Maintenance	Server
ENSURE information updates managed	Maintenance	Server

Table 1: Maintenance Goal Example

Goal Traceability

Hypertext links enable traceability to take various forms in GBRAT. When a user creates a new goal, the user must specify the name of the information *source* from which each goal was identified. In Figure 2, the source is an email message from Kenji Takahashi dated 9 Feb This ensures that each goal can be traced back to its place (i.e. document) of origin. It also enables analysts to easily identify goals which may have been extracted from more than one information source so that any similarities and differences can be immediately reconciled. Goals may also be traced back to the *responsible agents*. Further enhancements to GBRAT will include traceability among obstacles and scenarios as well as pre-conditions and post-conditions.

Viewing Goals by Name

Goals may be viewed by various filters in GBRAT. When viewed by *name*, the goals are listed alphabetically and displayed in a tabular format, as shown in Figure 3. GBRAT allows users to view either achievement or maintenance goals by name. The goals in Figure 3 are achievement goals. By clicking on the bullet (labeled **M**) in the left hand column, users can modify the selected goal. The second column (labeled **C**) notifies the users of the existence of any constraints on each goal. A constraint places a condition on the achievement of a goal. For example, in Figure 2, `Member must be able to ascertain if product was previously purchased` places a condition on the achievement of the goal `AVOID duplicate purchase`. In the third column (labeled **Name**) the user can select a goal by name in order to view all the properties of the goal (This view is shown in Figure **??** and discussed in more detail below). The next three columns list the agents responsible for each goal, the goal obstacles and scenarios (respectively). Goal obstacles are behaviors that prevent or block the achievement of a given goal. Scenarios are behavioral descriptions of a system and its environment. The bullets in the far right column (labeled **D**) allows users to delete goals.

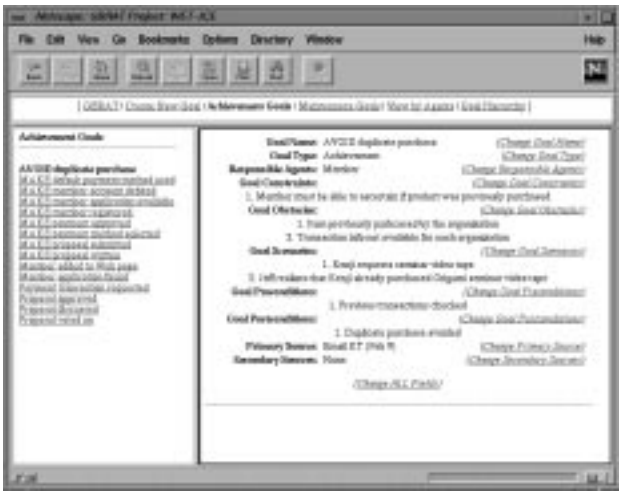

Figure 3: Viewing Goals by Name

Viewing Goals by Agent

Many times analysts need the ability to look at all of the goals for which a particular agent is responsible. For each agent, the relevant goals he or she is responsible for are displayed in the same format as described above. However, a different table is created for each agent. For example, Figure 4 shows a few of the achievement goals which a consortium member is responsible for in the electronic commerce web-server system. More than one agent can be associated with a goal. Our experience with GBRAT has shown that when the same goal is identified from two different sources, the only difference between the two goals is often the responsible agents. GBRAT notifies the user when this occurs and allows the user to merge the two goals into one goal with multiple responsible agents.

Figure 4: Viewing Goals by Agent

Viewing Goals by Precedence Relation

All achievement goals are related in some way to the other goals in the system. A *precedence* relation exists between goals G_1 and G_2, when goal G_1 must be completed before goal G_2. Our main interest in organizing achievement goals according to their precedence relations is to enable analysts to envisage goal operationalizations and refinements. GBRAT enables users to specify precedence relations among achievement goals so that a total ordering can be produced for the system goals. Once the user has specified the precedence relations, GBRAT assigns a number to each goal and displays the goals according to that ordering. Figure 5 shows the ordering produced by GBRAT for goals based on the ordering specified by the user. This view of the goals is helpful. The easy identification of synonymous goals in clusters facilitates an analyst's ability to recognize those goals which need to be reconciled, merged or elaborated.

Viewing a Goal's Properties

Goals have 7 properties, as shown in Figure ??. A link is provided from each goal in a table to the goal record itself. Figure shows this view for the goal `AVOID duplicate purchases` it includes, for each goal, its name, responsible agent(s), constraints, obstacles, scenarios, as well as any pre- and post-conditions.

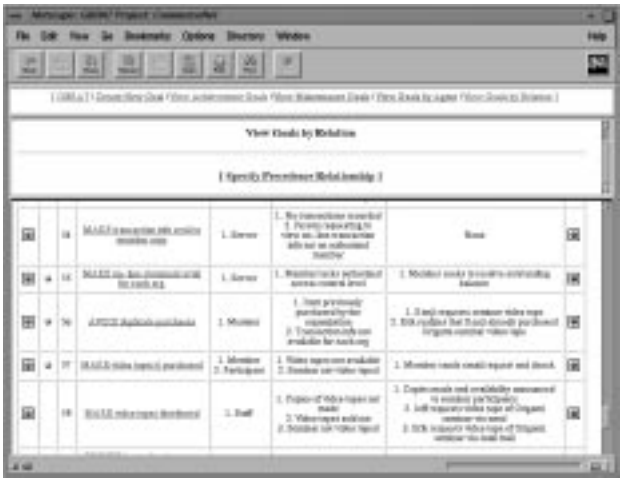

Figure 5: Viewing Goals by Precedence Relation

Implementation

Having described *what* GBRAT does, we now describe *how* it is implemented. The WWW provides a consistent user interface and the ability to incorporate a wide range of technologies and document types. Web browsers allow multiple users in different physical locations to access information via the Web. These characteristics played a role in the decision to develop GBRAT as a Web-based application. GBRAT allows analysts working in different locations to easily access the same documents. Netscape offers a consistent interface across different platforms and nonstandard HTML tags. It has built in security capabilities that enable us to limit access to registered GBRAT users. GBRAT is compliant with Web browsers; the capability to establish clearly visible links from one document to another as well as within documents is supported via hypertext links.

The cacheing capability of WWW browsers requires repeated reloading of modified pages. However, by using PERL scripts to retrieve information from the goal database, pages are dynamically generated and the user is assured that all information displayed is always up-to-date. Goals and goal properties are entered in natural language fragments. GBRAT easily manipulates and scans large amounts of text as evidenced by the ability to display goals via different filters as shown in Figures 2, 3, 4 and 5.

Viewing Goal Hierarchy Relationships

Drag-and-drop seemed the most intuitive way to build a visualization of the relationships among goals.

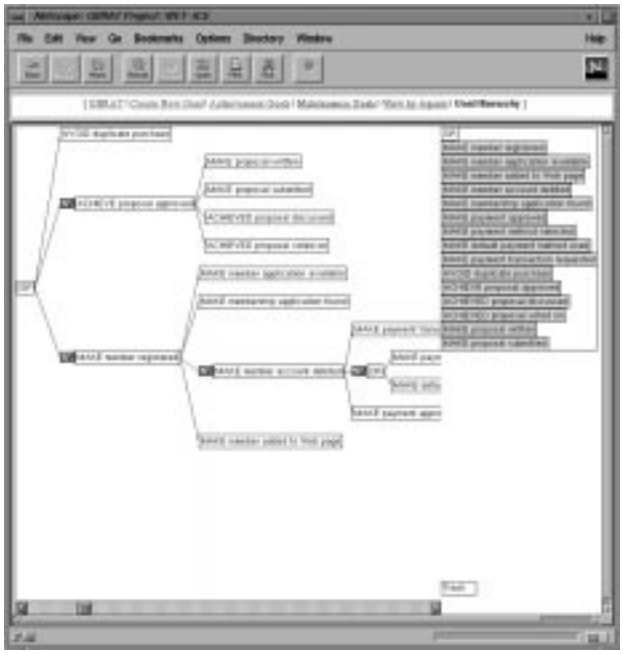

Figure 6: Goal Hierarchy

In order to show these relationships, an n-ary tree structure can be constructed, with the root being the name of the project (repository) and any number of children at each level in the tree. This allows for showing that a particular goal is a subgoal of another, and must be completed as a prerequisite of its parent. The goal list, at right, includes an item labelled 'OR' so that OR relationships among goals may be created- this goal depends on 'goal A' OR 'goal B' being completed. By default, an AND relationship is assumed for subgoals, so hierarchies that reflect more complex requirements may be built using these AND and OR relationships.

In order to make the visualization as malleable as possible, to simplify its use and be forgiving of mistakes, it was necessary to allow the dragging not only of goals onto the tree but also leaves and entire subtrees. This allows repositioning of the goals to reflect changes in the project's structure or subtasks.

Feedback is provided to the user in several ways. First, naturally, the goal is displayed as it is dragged, so that the user can tell they are taking the action of adding a goal to the hierarchy. Second, the goal is highlighted while it is over an area where it can be dropped, which helps the user ascertain that they are adding the new goal exactly where they want to. Further, the name of the goal the the user is over (i.e. the goal that the user would add a subgoal to, where (s)he to release the drop at that point) is displayed in the status bar at the bottom of the screen. After a particular goal has been inserted into the hierarchy, it is highlighted in the goal list at right, so that the user will know that goal has been used at least once. Lastly, when the user is not dragging a goal over a subtree, the status bar displays the name of the last goal that the user added a subgoal to, so that if the hierarchy becomes large and/or dense the user will not lose h(is)(er) place.

Each subtree in the hierarchy can be collapsed to save space, as well as to abstract away its subgoals. An arrow appears to the left of each subtree, i.e. leaf nodes do not display an arrow, as they have no children. The arrow changes orientation depending on the state of the subtree- when it is open and showing its subgoals the arrow points down; when it is closed, the arrow points to the right. The arrow is also the place the user should click in order to open and close the hierarchy.

By dragging the label for a subtree or leaf, the user can reposition it. A subtle drop shadow is provided in these cases, as feedback that the structure is out of the tree and being dragged. This feedback also occurs in the tree, as it redisplays itself to show the hierarchy without the goal being dragged.

If a goal from the goal list at right is dropped on an invalid area, such as empty space or the trash area, it simply goes away (though the original remains in the list so that it can be used again; this is because certain goals, such as 'training,' may be needed more than once. Ideally, though, a separate goal would be created for each situation, to preserve the specificity of requirements). If a subtree or leaf from the tree is dropped in the trash area, it also goes away, in effect removing it from the hierarchy. As before, the goals remain in the goal list at right so that they may be used again. If a subtree or leaf is dropped onto empty space, it returns to the level it was originall dragged out of, but is inserted as the first child on that level. If a subtree or leaf is dropped onto the same parent it was dragged off of, it is inserted as the last child at that level. These two positioning features allow goals to be arranged in a particular order within subtrees, to show the order that subgoals need to be completed in, for example.

The tree exists within a Panner object that turns scrollbars on and off as appropriate, so that the user may scroll to any section of the tree if the latter grows large. This feature, along with collapsibility, allow for effective use of limited space.

4 Summary and Conclusions

Goal-oriented methods are attracting research interest, but there exists little work in the form of tool support for these methods. GBRAT supports the collaborative nature of requirements engineering using interactive WWW technologies and allows project members located anywhere around the world to work col-

laboratively. Ultimately, GBRAT will support the two stages of our goal-based approach (GBRAM) [1]:

1. Goal Analysis: The process of exploring gathered information and then identifying, classifying and organizing goals.

2. Goal Refinement and Decomposition: The process of refining the classified goals and decomposing them into functional requirements.

GBRAT currently provides a mechanism for analysts to capture, classify and organize goal information. The ability to specify goal hierarchies offers an initial approach to goal refinement. Future work will further support the goal refinement and decomposition process. GBRAM calls for the ability to assign precedence relations to each goal in order to organize the goals accordingly. In its current state, the tool allows users to reorder goals based on their precedence relations resulting in a total ordering of the specified goals. Future extensions will enable users to refine goals by specifying subgoals with precedence relations assigned to each goal's subgoals. The use of Web technology for collaborative requirements engineering is still in its infancy. GBRAT is offers some minimal support for defining and specifying goal repositories. By employing the tool in our work on electronic commerce applications we are seeking to answer how goals are used to identify and refine system requirements and how the method's strategies are used in reengineering efforts involving a team of analysts.

Problems

The use of the WWW introduces problems linked to cooperation, information sharing, different viewpoints and networked enterprises.

Acknowledgements

The authors would like to thank NTT for supporting this research, Dr. Peter Freeman who directed the development of GBRAT, as well as, Colin Potts, Kenji Takahashi, Jeff Smith and Erik Bataller.

References

[1] Antón, A.I., "Goal-Based Requirements Analysis," *2nd IEEE International Conference on Requirements Engineering (ICRE '96)*, Colorado Springs, Colorado, 15-18 April 1996, pp. 136-144.

[2] Antón, A.I., "Goal-Based Requirements Analysis Tool (GBRAT): Requirements Document," Version 0.3, Georgia Institute of Technology Web Page, http://www.cc.gatech.edu/ computing/ SW_Eng/ Project/ reqts_doc.html, 14 November 1995.

[3] Liang, E, "Goal-Based Requirements Analysis Tool (GBRAT): Design Document," Version 0.3, Georgia Institute of Technology Web Page, http:// www.cc.gatech.edu/ computing/ SW_Eng/ Project/ design_doc.html, 14 November 1995.

[4] Potts, C., K. Takahashi and A.I. Antón, "Inquiry-Based Requirements Analysis," *IEEE Software*, 11(2), pp. 21-32, March 1994.