CHAPTER 4

The Goal-Based Requirements Analysis Method

When people are unaware of their own goals, they are often attracted to the seemingly glamorous goals of others.

Harry Palmer

This thesis addresses the critical nature of the discovery process in goal analysis. The process of identifying high-level goals is fundamental to the requirements analysis and specification process. Existing goal-based methods usually fail to address the initial identification and origin of goals, taking previous documentation of the goals for granted [7, 24,90]. This chapter introduces the Goal-Based Requirements Analysis Method (GBRAM) which assumes that goals have not been previously documented or explicitly elicited from the stakeholders and that the analyst must work from existing diagrams of, for instance, processes or information flows, textual statements of need, and/or additional sources of information such as transcripts of interviews with stakeholders to determine the goals of the desired system.

Several approaches, surveyed in Chapter 2, exist for refining goals once they have been identified (e.g. [24, 90]). In contrast to other approaches, GBRAM focuses on the initial identification and abstraction of goals from all available sources of information, regardless of the scope of the knowledge base. It also supports the elaboration of goals to represent

the desired system.

This chapter introduces the Goal-Based Requirements Analysis Method in detail. An overview of the method is provided in Section 4.1. This overview discusses the activities an analyst is involved with while employing the method, differentiating between the goal analysis and goal refinement phases. Section 4.2 illustrates the method in more detail by presenting a reasonable sequence of steps to progress from initial identification of goals to translation of those goals in operational requirements.

4.1 Overview of GBRAM

Figure 4.1 on page 69 shows the activities with which an analyst is intimately involved when applying the GBRAM. The ovals located within the dotted box on the upper right corner of the figure denote the *goal analysis** activities. The goal analysis activities may be summarized as follows:

- *Explore* activities entail the examination of the 'inputs'.
- *Identify* activities entail extracting goals and their responsible agents from the available documentation.
- *Organize* activities involve the classification of goals and organization of those goals according to goal dependency relations.

Goal analysis concerns the exploration of documentation for goal identification followed by the organization and classification of goals.

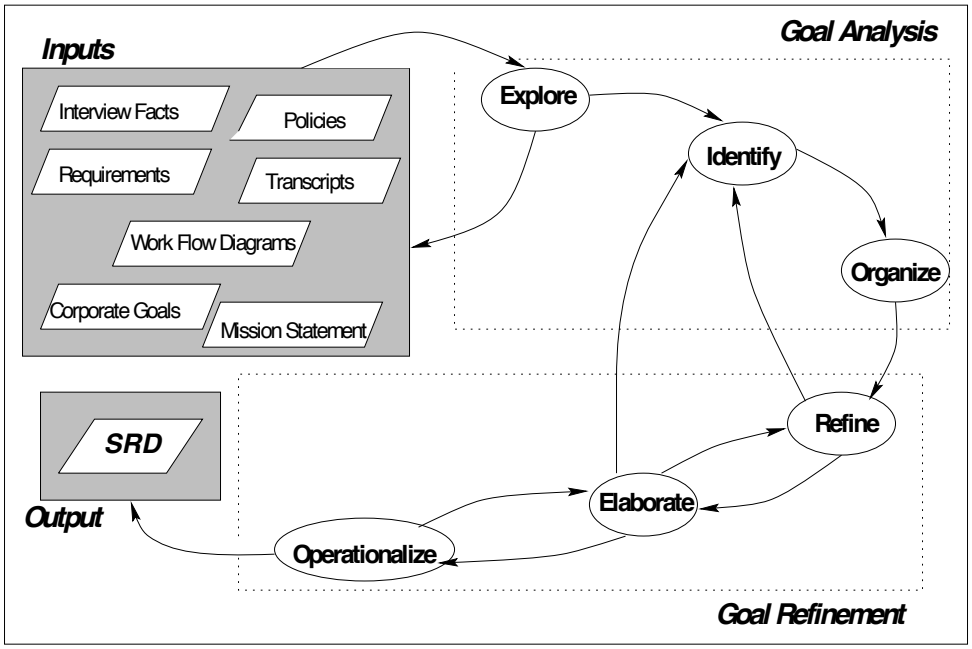

Figure 4.1. Overview of GBRAM Activities

The ovals within the dotted box on the lower half of the figure denote the activities that take place during *goal refinement**. The goal refinement activities may be summarized as follows:

- *Refine* activities entail the actual pruning of the goal set.
- *Elaborate* refers to the process of analyzing the goal set by considering possible goal obstacles and constructing scenarios to uncover hidden goals and requirements.
- *Operationalize* refers to translating goals into operational requirements for the final requirements specification.

* *Goal refinement* concerns the evolution of goals from the moment they are first identified to the moment they are translated into operational requirements for the system specification.

The box in the top left corner of Figure 4.1 contains the possible *inputs*, which may vary in accordance with the documentation initially available to analysts. The *output* of GBRAM (as shown in Figure 4.1) is always a software requirements document*. The SRD includes the functional† and nonfunctional‡ requirements and should be very specific with regard to the external behavior of the system. A generalized summary of the inputs and output of each GBRAM activity is presented in Table 4.1.

The level of detail or generality of the requirements stated in a SRD is at the discretion of the authors of the document. Stakeholders or potential customers tend to express requirements in general terms; while analysts or developers tend to express requirements in greater detail, relying on formal methods and notations. In using formal representations to help reduce the level of ambiguity, analysts (who may not be familiar with the application domain) produce requirements documents which can be intimidating to stakeholders unfamiliar with these notations, making the requirements difficult to understand. In contrast, stakeholders are typically experts in the application domain but are not trained in formal analysis methods. Requirements expressed by stakeholders tend to be ambiguous (as discussed in Chapter 1). The SRD produced using GBRAM addresses the recurring problem of ambiguity and yet does not rely on formal or mathematical notations. GBRAM allows the requirements to be expressed in natural language prose so they may be easily understood by non-computer experts, thus encouraging the active involvement of the stakeholders throughout the process.

---

*A *software requirements document* (SRD) is a document that contains a complete description of *what* the software will do without describing *how* it will do it [26].

†*Functional requirements* describe the behavioral aspects of a system.

‡*Nonfunctional requirements* describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate.

Table 4.1. Inputs and Outputs of GBRAM Activities

Activity	Inputs	Outputs
Explore	Requirements	Organized artifacts
	Interview transcripts	Goals
	Work flow diagrams	
	Corporate goals	
	Policies	
	Interview facts	
	Mission statement	
Identify	Requirements	Goals
	Interview transcripts	Stakeholders
	Work flow diagrams	Agents
	Corporate goals	
	Mission statement	
Organize	Goals	Achievement goals
		Maintenance goals
		Dependency relations
		Reduced goal set
		Goal topography
Refine	Goal set	Goal obstacles
		Scenarios
		Constraints
Operationalize	Goal set	Requirements
		Goal schemas
		Action definitions
		Software Requirements Document

The SRD produced by GBRAM provides a means of:

- communication among stakeholders, users, analysts, and developers;
- supporting requirements validation; and
- supporting requirements evolution.

Since a SRD must serve as a primary means for communication among the stakeholders and analysts, it must be well-written and unambiguous. As discussed in Chapter 1, it is critical to address any misinterpretations and disagreements among stakeholders as early as possible. The SRD is also a basis for requirements validation, clarifying the stakeholders' intentions. Additionally, SRDs help control the evolution of the software system by tracking rationale associated with decisions and assumptions made regarding the requirements.

This chapter introduces the activities involved in the goal-based requirements analysis method and presents techniques for accomplishing these activities. Section 4.2 discusses the goal analysis activities and Section 4.3 discusses the goal refinement activities. A generalized summary of the activities is presented in Table 4.2.

4.2 Goal Analysis Activities

This section discusses the activities which an analyst must perform in GBRAM during goal analysis. Techniques are provided to execute these activities. As shown in Figure 4.1 on page 69, goal analysis concerns three specific activities which may be summarized as follows:

- *Exploration* of existing documentation for the initial identification of goals;
- *Identification* of goals, stakeholders and responsible agents; and
- *Organization* of goals according to dependency relations and classification of goals according to target conditions.

Table 4.2. Overview of GBRAM Analysis Activities

Activity	Description
Explore	The main objective of exploring is to extract and identify all goals from the existing sources of information. Goals are not always explicitly stated; however, analysts may abstract goals by observing that the statements which describe the purpose of a system or process generally provide insight into what the goals are.
Identify	Goal identification techniques are applied to extract and identify goals from the available documentation to initially specify them for further elaboration. Stakeholders are identified by considering who or what claims an interest in each goal. In addition to goals and stakeholders, the responsible agents must be identified thereby allocating responsibility assignments to each goal.
Organize	The main objective of organizing is to classify the goals are classified according to their target conditions and organize the goals according to their dependencies. Relationships among goals are determined by considering the dependency relations. When the goals have been refined and ordered according to their precedence relations and the goal hierarchy is constructed.

The remainder of this section provides a discussion of each of these activities within the context of the associated activities shown in Figure 4.1. Examples of the method's applicability, taken from the case studies presented in Chapter 3, are provided throughout the remainder of this chapter. It should be noted that these activities need not be performed sequentially; rather, they may be performed concurrently with occasional interleaving and

iteration, as illustrated by the arrows in Figure 4.1's overview of the GBRAM.

Exploration of Existing Documentation

Ross [71] observed that goals drive the identification of supporting requirements. Goals also restrict acquisition of requirements information to facts which are relevant to the purpose of a system. This tenet forms the basis for exploring any existing documentation to identify goals, their responsible agents, and any stakeholders that claim an interest in those goals.

It should be noted that GBRAM is not dependent upon any specific representations. Instead, GBRAM recognizes the typical challenge of working with different sources of knowledge that are represented in different forms. It provides a goal identification strategy that may be applied to various types of informal descriptions of the desired system. An overview of the GBRAM activities is provided in Table 4.2. The following section discusses how to analyze various descriptions of the desired system in order to extract and identify goals.

Identifying Goals and Objectives

Using GBRAM, analysts must first explore the available information to identify and extract goals from these sources (Table 4.2). It is good practice to gather as much relevant information as possible to understand the design implications of goals; thus, in GBRAM analysts explore these information sources to identify and extract goals. These information sources or descriptions may be provided in such diverse formats as textual statements, transcripts of interviews, charts, diagrams (e.g. Entity Relationship Diagrams), process

descriptions, or even explicitly stated goals (i.e., an organization mission statement or a strategic plan). The level of detail involved will vary greatly depending upon whether a completely new system is desired or a current system is already in place but in need of modification. GBRAM aids analysts in making the best possible use of the information available to them. The remainder of this subsection discusses techniques to support this identification process.

To identify goals, each statement (or piece of information) is analyzed by asking, *"What goal(s) does this statement/fragment exemplify?"* and/or *"What goal(s) does this statement block or obstruct?"* In order to operationalize goals for specification, analysts must be able to reason about any preconditions and postconditions on the goals and the corresponding system operations. It is for this reason that the identified goals are worded to emphasize the state that is true, or the condition that holds true, when the goal is realized. Table 4.2 summarizes the identification process.

Examples 4.1 and 4.2 demonstrate two techniques for identifying and extracting goals from textual descriptions of the desired system by looking for statements which guide decisions and action words. For the remainder of this thesis, such textual descriptions are referred to as Natural Language Descriptions (NLDs).

Example 4.1 As a general rule, statements which seem to guide design decisions at various levels within the system or organization point to possible goals. Consider the following NLD for an Air Force Base career training system (Career Track Training System; Chapter 3.2).

> **NLD #1:** *Congress has mandated that government acquisition professionals in the Department of Defense (DoD) must improve their acquisition skills so that they may spend tax payers' money allocated for weapons systems more effectively*

and efficiently. A DoD-wide program that includes positions and qualifying training was established to provide career tracks for these acquisition professionals.

Recall that in Table 4.1, one of the initial inputs is policies; NLD #1 is a prescriptive organizational/policy level description of the desired system which delineates the objectives of the organization. By examining each statement in NLD #1 and asking *"What goal does this fragment exemplify?"* several goals become evident from the description: `Skills Improved`, `Position training provided`, `Qualifying training provided`, `Career tracks provided`, and `Tax payer money spent efficiently`.

All action words are possible candidates for goals in the proposed system. Goals may thus also be identified by searching for action words which point to some state that is, or can be, achieved within the system once the action is completed. This is an extension of previously supported techniques (Abbott [2] in OOD, Booch [10], Rumbaugh [75]; Rumbaugh takes it further by also circling verbs). Certain types of verbs such as *allocated*, *completed*, *achieved*, *found out*, and *satisfied* intimate possible goals (this is discussed further in Chapter 5).

Example 4.2 To demonstrate the 'action word identification' approach, consider NLD #2 , which provides a start-to-finish explanation of the training acquisition process designed for an employee enrolling in a training course (Career Track Training System; Chapter 3.2).

NLD #2: *TSD (Training System Directorate) uses the information in the database to arrange and coordinate training, to track progress of professionals endeavoring to improve their qualifications, and to ensure that TS professionals meet the APDP requirements of their respective positions.*

Several action words (verbs) may be found in NLD #2: *arrange*, *coordinate*, *track*, *improve*, and *ensure*. These action words serve as indicators for the goals: `Training coordinated`, `Progress tracked`, and `Qualifications improved`.

Goals are thus identified using inquiry-driven and traditional action word location techniques. These techniques are not limited simply to the initial goal identification phase; they may be applied throughout the analysis effort.

Although goal identification is discussed prior to discussion of stakeholder identification in this chapter, the activities do not preclude each other. Stakeholders must often be identified before any goals can be specified. Analysts must understand who the stakeholders are before they can even begin to develop an understanding of the goals. However, this thesis, assumes that the analysts already posses an understanding of general stakeholders for the system prior to goal identification. In GBRAM, stakeholder analysis is a vehicle for considering multiple viewpoints and potentially affected parties for various goals within the system. Stakeholder conflicts may be detected early by considering the needs of different stakeholders throughout the analysis process. The following section discusses stakeholder identification.

Identifying Stakeholders

As discussed in Chapter 2, it is important to capture stakeholders'* viewpoints so that conflicts may be surfaced early. Identifying stakeholders determines who or what claims an interest in each goal so that an understanding may be gained regarding the different view-

*A *stakeholder* is anyone who claims an interest in the enterprise or system.

points and stipulations contributing to the system, allowing for future conflict resolution. Multiple stakeholders may be associated with one goal; a stakeholder is thus not simply a system 'user' in the classical sense, but rather, any representative affected by the achievement or prevention of a particular goal. For clarification, the difference between an agent and a stakeholder should be noted. As shown in Figure 4.2, some agents are stakeholders and some stakeholders are agents; that is, $Agents \cap Stakeholders$. A stakeholders may be a customer[†], actor[‡], owner[§], [15] or representatives of organizations. The agents that lie outside of the intersection of *Agents* and *Stakeholders* are not stakeholders; instead, they are system-specific agents.

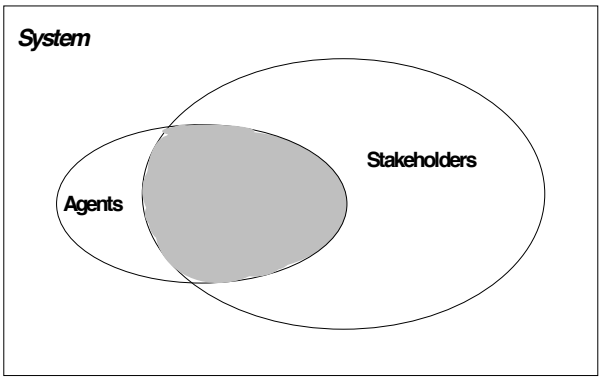

Figure 4.2. Venn Diagram Distinguishing Agents and Stakeholders

[†]A *customer* is a perceived beneficiary of the system.
[‡]An *actor* is someone who actually performs functions in the system.
[§]An *owner* is a customer in the contractual sense.

The stakeholders for each goal are determined by asking *"Who or what claims a stake in this goal?"* and *"Who or what stands to gain or lose by the completion or prevention of this goal?"* Examples 4.3 and 4.4 demonstrate how to consider stakeholders during goal analysis.

Example 4.3 G_7 (`Skills improved`) in Table 4.3 shows that both the employees and the organization, which in this example is the Air Force Base (AFB), have an interest or stake in improving employee skill levels. However, the AFB claims the sole stake in the goal of providing career tracks for *all* its employees (G_1 : `Career tracks provided`) while each employee is ultimately interested in their own *individual* training acquisition status and skill level.

Table 4.3. Stakeholder Analysis Example

Goals	Agent	Stakeholders
G_1 : Career tracks provided	AFB	AFB
G_5 : Approval granted	AFB	employee
G_7 : Skills improved	employee	AFB, employee
G_8 : Certification granted	AFB	AFB, employee

Example 4.4 G_8 (`Certification granted`) in Table 4.3 is of interest to both the AFB and each employee seeking certification. Both stakeholders claim a mutual stake in this goal. However, the AFB, not the employee, is ultimately *responsible* for granting certification to the employee. Thus, responsibility is allocated to one agent, the AFB. Agent identification is discussed further in the following section. This highlights the potential for a goal to

prevent another agent from achieving a goal. For example, if G_5 (`Approval granted`) is not achieved, then an employee is unable to achieve G_8 (`Certification granted`), which may in turn prevent an employee from enrolling in any subsequent courses. This occurrence is discussed further in Chapter 5.

Stakeholders are thus identified by inquiry using the questions presented above. Once the goals and stakeholders are specified, the goals must be assigned to their responsible agent(s).

Identifying Agents and Agent Responsibilities

Responsibility assignment allocates goals to organizational components (including automated systems). Goals can be used to specify responsibility assignments for certain actions in a system to specific agents*. Typically, only one agent is responsible for ensuring the completion of a goal at any given time; however, different agents may be responsible for the completion of the same goal at different times.

A logical approach for identifying the agents is to consider which agents are ultimately responsible for the achievement or maintenance of each goal by asking the question, *"Who or what agent [is/should be/could be] responsible for this goal?"* For example, in a meeting scheduler system the goal `Meeting scheduled` is the responsibility of the meeting scheduler. Depending on the desired implementation, the agent in this case could actually be either the system (as suggested above) or a human agent; for example, in an email-based implementation of this same system the responsible agent *could be* someone other than the

Agents are responsible for the completion and/or satisfaction of goals within an organization or system.

meeting scheduler system such as a member of the clerical staff. Example 4.5 shows how a textual description may be analyzed to identify a goal's responsible agent(s).

Example 4.5 Consider the description below (NLD #3) that describes the process for an employee who has completed a training course and wishes to advance to the next level of available courses:

> **NLD #3:** *When a professional completes a course or courses that qualifies them to advance to another level in the program, they assemble the course completion certificates, write a letter, and submit the letter and proof of course completion to the headquarters of their major command organization.*

The most obvious goals that may be extrapolated from NLD #3 are: `Course completed` and `Proof of course completion submitted`. By asking *"Who or what agent is responsible for these goal?"* it becomes clear that in the context of this system, the professional is responsible for showing proof of course completion (as illustrated in Table 4.4).

Table 4.4. Goals Identified from NLD #3

Goals	Agent
G_6 : Courses completed	employee
G_{13} : Proof of course completion submitted	employee

Organization and Classification of Goals

Organization of goals entails eliminating redundancies and reconciling synonymous goals while classification of goals involves differentiating goals according to their target conditions (Table 4.2). In GBRAM, all goals are classified as either maintenance goals or achievement goals; maintenance goals can define the scope of achievement goals. Achievement goals must be organized according to their dependency relations so that they may be operationalized as much as possible in the form of goal schemas. Goal dependencies are specified so that a goal hierarchy may then be constructed based on the dependency relations. The following subsections discuss techniques for accomplishing these activities and provide examples from the case studies discussed in Chapter 3 for clarification.

Eliminating Redundancies and Reconciling Synonymous Goals

Goals are initially refined by eliminating redundancies and reconciling synonymous goals. Goals are considered synonymous if their intended states are equivalent or if they mean the same thing to different stakeholders who simply express the goal using different terminology. It is up to the analyst to identify these instances. For example, the goals in a meeting scheduler system `Meeting Arranged` and `Meeting Scheduled` are synonymous and can be reconciled as one goal which encompasses the spirit and scope of both. The analyst can choose either of the two goal names; however, all related essential information must also be maintained. Thus, if the same goal appears more than once, and the same agent is responsible for the goal on all occurrences, all but one of the goals should be eliminated. However, if two different agents are responsible for the same goal at different times, then

all but one of the goals should be eliminated, but the analyst must keep track of all agents who assume responsibility by annotating the goal accordingly.

Redundancies and synonymous goals are most easily identified after the goals have been ordered according to their precedence relations. It is beneficial to identify synonymous goals after ordering them because they are typically listed adjacent to each other (or clustered) in the ordered set, reflecting their shared common precedence relations. Examples 4.6 and 4.7 demonstrate the analysis that results from identifying two pairs of synonymous goals.

Example 4.6 Goals can be consolidated and refined by merging synonymous goals. For example, in Table 4.5, G_7 (`Skills improved`) can 'absorb' G_{17} (`Qualifications improved`) since these goals are synonymous. Thus if two goals are synonymous, reconcile them by eliminating the goal whose content is described by the other.

Table 4.5. Reconciling and Merging Synonymous Goals

Goals	Type	Agent	Stakeholders
G_3 : Training coordinated	Maintenance	AFB	AFB, employee
G_7 : Skills improved	Achievement	employee	AFB, employee
G_{17} : Qualifications improved	Achievement	AFB	employee
G_{20} : Training provided	Maintenance	AFB	AFB, employee

Ideally, stakeholders are encouraged to participate in this refinement process so that analysts can bring such discrepancies to the attention of the stakeholders, allowing the stakeholders to indicate which goal name is most appropriate.

Example 4.7 Similarly, G_{20} (`Training provided`) and G_3 (`Training coordinated`) in Table 4.5 are synonymous and may thus be consolidated into one goal.

Classifying Goals

The GBRAM classification scheme differentiates among types of goals according to the target conditions of the goals; goals are classified as either maintenance or achievement goals. A *maintenance goal* is satisfied as long as its target condition remains true. An *achievement goal* is satisfied when its target condition is attained. GBRAM focuses primarily on achievement goals because they map to actions that occur in the system and aid analysts in specifying the functional requirements* necessary to satisfy the needs of the stakeholders and customers. Since maintenance goals suggest a continuous state within the system, they may generally be mapped to non-functional requirements†. It should be noted that not all maintenance goals map to non-functional requirements. However, during the initial classification of goals, observation has shown the convenience of classifying all goals which suggest a continuous state in the system as Maintenance goals. Given that high-level maintenance goals are often not reflected in operational strategies [7], it is useful to differentiate them from achievement goals. Maintenance goals naturally map to safety requirements; for example, in an electronic commerce Web server the goal `Secure transactions ensured` points to a safety requirement for the system. Thus, the objective of separating the maintenance goals from the achievement goals is to set aside the organization policy level goals for future resolution of conflicting achievement goals. However,

**Functional requirements* describe the behavioral aspects of a system.

†*Nonfunctional requirements* describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate.

organizational goals may not be allocated as a responsibility of, or within, the system. A more in-depth analysis may be required of the maintenance goals, since some of these goals may also be operationalized. Further analysis will ensure that those goals initially classified as maintenance goals which may be refined into operational goals, will be re-classified as achievement goals.

Maintenance goals are classified by considering each goal and asking: *"Does this goal ensure that some condition is held true throughout all other goal operationalizations?"*, *"Does this goal affect decisions at various levels within the organization?"*, and *"Is continuous achievement of this goal required?"* Maintenance goals can also be identified by searching for certain key words (i.e. *provide* and *supply*) that suggest a continual state within the system. Figure 4.3 shows a few key words which have been observed to be helpful in indicating candidate maintenance goals.

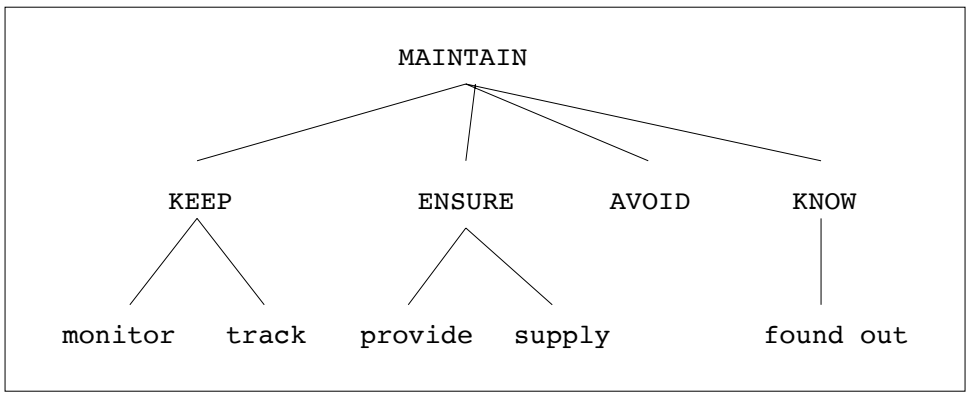

Figure 4.3. Useful Key Words for Classifying Maintenance Goals

NLD #4: Accounts Undesignated. *Professor salaries are not always completely paid by State funds. When we do not know which account or contracts will be used to pay for a Professor's entire salary, we assign the professor to an undesignated account called a dummy number. Since we are not supposed to use dummy numbers, professors assigned to these undesignated accounts must be moved off of them as soon as possible.*

Example 4.8 NLD #4.2, a vignette taken from an interview with a stakeholder in the Financial Services Office (FSO) study (Chapter 3.1, page 37), demonstrates how searching for key words aids in classifying maintenance goals. From this description, it is clear that the FSO staff is 'supposed' to avoid assigning professors to undesignated accounts. By following the GBRAM guidelines for goal identification, this would have initially been specified as `Avoid accounts undesignated`. The word *Avoid* indicates that this is a maintenance goal.

Example 4.9 The goals in Table 4.6 were extracted from the Career Track Training System documentation (See Chapter 3.2, page 45), and classified as maintenance goals by asking *"Does this goal ensure that some condition is held true throughout all other goal operationalizations?"* and *"Is continuous achievement of this goal required?"* For example, G_2 in Table 4.6 (`Tax payers money spent efficiently`) must be achieved on a 'continuous' basis. The AFB business process mandates that career tracks be provided to ensure that tax payers' money is spent efficiently. Note that adverbs may suggest maintenance goals. This goal characterizes a condition which must be held true.

A distinction which can be made between maintenance and achievement goals is that maintenance goals have a pervasive effect on achievement goals. In contrast, achievement goals are relatively self-contained.

Table 4.6. Maintenance Goals, Agents, and Stakeholders

Goals	Agent	Stakeholders
G_1 : ENSURE Career Tracks Provided	AFB	AFB
G_2 : ENSURE Tax payers money spent efficiently	AFB	AFB, DoD, empl
G_3 : MAINTAIN Training coordinated	AFB	AFB, DoD, empl

Achievement goals are classified by asking *"Is completion of this goal self-contained?"*, *"Does this goal denote a state that has been achieved or a desired state?"*, *"Does achievement of this goal depend on the completion of another goal?"*, and *"Does the ability of another goal(s) to complete depend on the completion of this goal?"* Figure 4.4 shows some key words which have been observed to be helpful in indicating candidate achievement goals.

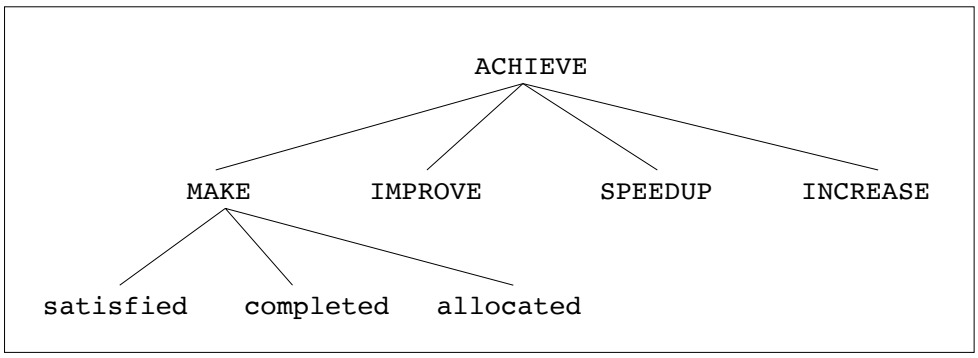

Figure 4.4. Useful Key Words for Classifying Achievement Goals

Example 4.10 Examples of achievement goals in the FSO project include: `Achieve account closed`, `Achieve budget balanced`, and `Achieve budget amendment completed`.

Specifying Goal Dependencies

Dependency relations exist between pairs of goals. A goal dependency implies that a given goal is contingent upon another goal for completion, relying on or requiring the aid of another goal or agent for support. The objective of this activity is to develop an understanding of these relationships among the goals. The notion of precedence relations among goals was introduced by Potts in [60]; a similar approach is discussed by Yu in [88]. In GBRAM, goals are organized according to their precedence relations, simplifying the determination of a goal's preconditions and postconditions. In GBRAM, the only type of dependency necessary for ordering the goals is the precedence dependency. However, two other types of dependencies are recognized: agent dependency and contract dependency; both are forms of precedence dependencies.

A *precedence* dependency between goals G_1 and G_2, where goal G_1 must be completed before goal G_2, is expressed as $G_1 < G_2$. Organizing achievement goals according to their precedence relations enables analysts to envisage operationalizations of these goals for consideration of possible elaborations and refinements. The goals are eventually elaborated so they may be operationalized as much as possible in the form of goal schemas that can be easily translated into a requirements document. This goal schema syntax is presented in Section 4.4 of this chapter.

Precedence relations are identified for each goal by asking *"What goals are prerequisites for this goal?"* and *"What goals must follow this goal?"* The answers to these questions facilitate the organization of goals with the prerequisite goals listed prior to a given goal. Example 4.11 demonstrates who precedence relations are analyzed.

Example 4.11 (Precedence Dependency) Consider goal G_4 (`Course completed`) in Table 4.7. By asking *"What goals must follow this goal?"* and *"Do any goals depend on the availability of this information for achievement?"* it is possible to determine that G_4 must occur before G_5 (`Skills improved`) and G_6 (`Certification granted`) can be achieved. Thus, $G_4 < G_5 < G_6$ (i.e. G_5 cannot be achieved before G_4 is completed).

Table 4.7. Precedence Dependency Example

Goals	Agent
G_4 : Course completed	employee
G_5 : Skills improved	employee
G_6 : Certification granted	AFB

A *contract* dependency between goals G_1 and G_2, where goal G_2 must be achieved if goal G_1 occurs, is expressed $G_1 \rightarrow G_2$; thus a contract dependency differs from a precedence dependency by a trigger. For example, G_1 happens hence G_2 must complete, as opposed to G_2 requiring G_1 to complete to enable G_2 to complete.

Example 4.12 (Contract Dependency) In Table 4.8 a contract relation is observed between G_{11} (`Course completed`) and G_{12} (`Skills improved`). An employee's skills are improved whenever they complete a course. However, if the employee does not complete the course (i.e. `Employee fails course`), then the employee does not satisfactorily improve their skills. Thus, if G_4 fails then G_5 will also fail; this is expressed as $G_4 \not\rightarrow G_5$.

Goals may also share dependencies among agents. For example, consider a payroll system where employees are paid by the hour. In order for one agent to complete the goal,

another agent may have to first complete another goal (indicating a precondition). Before an employee can be paid, the employee's supervisor depends on the employee to provide him the necessary information (e.g. time sheet). In this case, a precedence relation exists between the employee and the supervisor. Precedence relations can thus also be identified by searching for such agent dependencies. Once the goal relationships have been identified and the dependencies specified, the goal hierarchy may be constructed.

Table 4.8. Contract Dependency Example

Goals	Agent
G_{11} : Course completed	employee
G_{12} : Skills improved	employee

Constructing a Goal Hierarchy

Recall that the ultimate goal of GBRAM is the production of a software requirements document. Software requirements documents are typically difficult to index and read. Goals offer a rich outlining structure for organizing requirements information, addressing the need for documents that are easy to index and read. Since goals provide an organizing structure for requirements, we refer to this outlining mechanism as a *goal topography**. A topography is a graphical representation of the physical features of a place. A goal topography is a representation of the features of the SRD expressed in the form of a hierarchy that may

*This term is believed to be new; in this thesis it is proposed for use in the field of requirements analysis.

be mapped to the software requirements document (SRD) thereby providing a clear outline for the SRD. Thus, goals become an organizing mechanism on which to *hang* requirements, scenarios, and other related information such as pre- and post-conditions. The notion of goals as a topography is further developed in Chapter 5.

A goal topography can be represented in a number of ways. This thesis discusses two specific representation styles: outline and hierarchical. When a goal topography is manually constructed, it is typically expressed in the form of an outline. Figure 4.5 shows the top level of the topography, represented in outline form, produced for the vacation/sick leave system (see description in Appendix A).

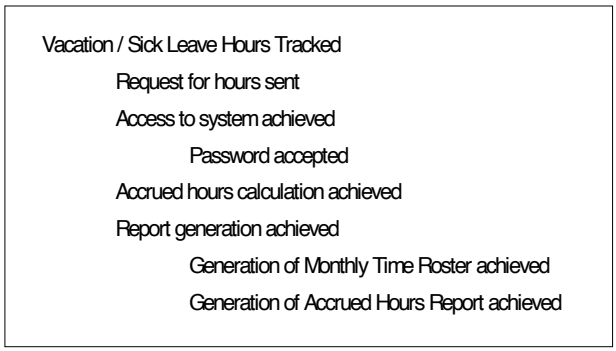

Figure 4.5. Goal Topography in Outline Form for Vacation/Sick Leave System

When a graphing tool is available, it is possible to construct a graphical hierarchy representation for expression of the topography. Figure 4.6 shows the graphical topography in a goal hierarchy format that is equivalent to the topography of Figure 4.5. This representation is similar to the hierarchy representation produced by the Goal-Based Requirements Analysis Tool (GBRAT) discussed in Chapter 6.1. The advantage of topographies expressed

in the form of an outline is that they provide a clear mapping to the outline for the SRD. The advantage of the hierarchy representation is that it may be used to represent goal relationships within the system, which may, as a result of it's visual nature, lead the analyst to reason about goal relationships from a different perspective, prompting the analyst uncover hidden relationships and dependencies.

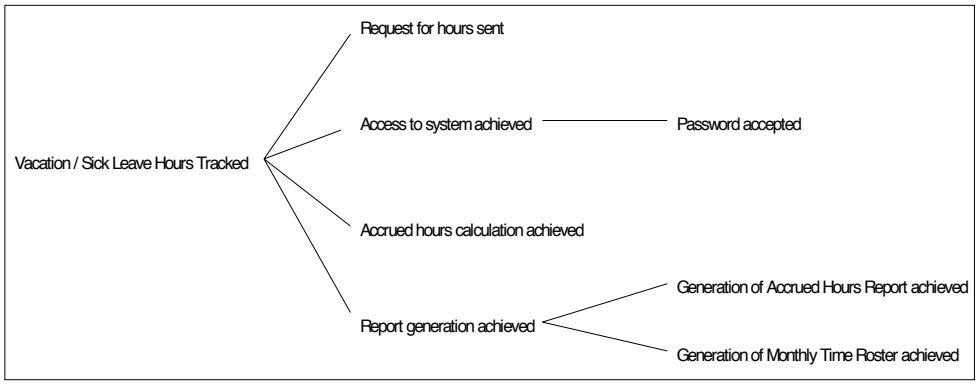

Figure 4.6. Goal Topography in Hierarchy Form for Vacation/Sick Leave System

In constructing the goal hierarchy for a vacation/sick leave system (discussed in Chapter 6.3), four recurring goal classes or categories were identified:

- messaging goals*;
- calculation goals†;
- report generation goals‡; and

Messaging goals pertain to notifications and reminders sent by and/or within the system

†*Calculation goals* involve calculations of accrual rates, balance, etc.

‡*Report generation goals* pertain to the generation of internal departmental reports as well as external institute summary reports

- security and access goals[§].

These four goal classes are clearly visible in both Figure 4.5 and Figure 4.6. `Request for hours sent` is a *messaging goal*; `Access to system achieved` is a *security and access goal*; `Accrued hours calculation achieved` is a *calculation goal*; and `Report generation achieved` is a *report generation goal*. All of the goals identified for this system fit into one of the four goal classes above. The organization provided by goals helps analysts find information and sort goals into naturally different functional requirements. By organizing requirements information into an SRD according to the goal topography, changes to the requirements can be managed. This is important because the goal topography enables analysts to localize the goals and requirements which are affected by a change in related or proximate goals. Since dependency relations are tracked, traceability is made possible and the narrowing of scope facilitates the identification of other goals and requirements that are affected as a result of a change in a specific goal.

There are a number of families of goal classes that provide an organizing principle for proposed software systems. In the Electronic Commerce case study (discussed in Chapter 6.2), several goal classes were identified:

- information display and organization goals;
- process support goals;
- security and access goals; and
- electronic commerce goals.

[§] *Security and access goals* restrict access to certain parts of the system to authorized users

While the goal types are specific to the application domain, some of these goal *classes* are generalizable to different software systems; for example, security and access goals may be observed in most multi-user systems. In addition, the information display and organization goals in the electronic commerce system are analogous to the report generation goals in the vacation/sick leave system. The electronic commerce goals are application specific; since they address the core functionality of the system, they thus correspond to the calculation goals in the vacation sick-leave system.

In the electronic commerce system, the security and access goals restrict Intranet access to members only (paid subscribers) while maintaining some general public access to non-restricted forms of information. In the vacation/sick leave hour tracking system, the security and access goals were ensured that both employees could not access a colleague's account and that the financial services office staff had full access to each employee's records. Although the application domain of both of these systems is distinct, one can certainly envision a generalizable class or subclass of security and access goals which may be viable for more than one system. This prospect is discussed further in Chapter 7.3. Once the goal topography is constructed, analysts must systematically elaborate and refine the goal set.

4.3 Goal Refinement Activities

This section discusses the activities which an analyst must perform in GBRAM during goal refinement. Techniques are provided to execute these activities. As shown in Figure 4.1 on page 69, goal refinement concerns three specific activities which may be summarized as follows:

- *Refinement* of the goal set to prune the size of the goal set;
- *Elaboration* of the goals to uncover hidden goals and requirements; and
- *Operationalization* of the goals into operational requirements.

The remainder of this section provides a discussion of each of these activities within the context of the associated activities shown in Figure 4.1. Examples of the method's applicability, taken from the case studies presented in Chapter 3, are provided throughout the remainder of this chapter. It should be noted that these activities need not be performed sequentially; rather they may be performed concurrently with occasional interleaving and iteration as evidenced by the arrows in Figure 4.1. A generalized summary of the goal refinement activities is presented in Table 4.9.

Table 4.9. Overview of GBRAM Refinement Activities

Activity	Description
Refine	The goal set is refined by eliminating redundant goals and reconciling synonymous goals.
Elaborate	The objective of elaboration is to identify goal obstacles for the consideration of possible ways for goals to fail and to construct scenarios to uncover hidden goals and requirements. Analysts begin the goal elaboration process by considering the possible ways in which the identified goals may be blocked which facilitates the anticipation of exception cases.
Operationalize	The objective of operationalization is to represent the goals a bit more formally (e.g. more formal that English) so that they may be mapped onto actions in a set of goal schemas.

Elaboration of Goals

Goals are elaborated by considering scenarios* and goal obstacles†. Goal obstacles are identified in order to consider the possible means of goals failure. They are elaborated further by identifying scenarios to develop an understanding of how the goals can be operationalized. Finally, goal constraints‡ are identified to expand the analysts' understanding of what obligations must be met for goal completion.

Specifying Goal Obstacles

The objective of specifying goal obstacles for each goal is to capture any information pertaining to the goals and system objectives that might otherwise be overlooked. Considering the possible ways in which goals may be blocked or may fail forces the consideration of specific cases that must be handled by the desired/proposed system due to unforeseen activities which may prevent goal completion. This step requires analysts to be inventive because they must identify and construct goal obstacles by inquiry from the available information sources.

Jackson observes that "whatever relationships you describe among phenomena of an informal domain, there can always be exceptions" [47]. Since every goal can be refuted by at least one counter example (through its negation), it follows that a goal is a *refutable description*§. When describing informal domains it is unlikely that one can be sure of always

---

* *Scenarios* are behavioral descriptions of a system and its environment arising from restricted situations.
† *Goal obstacles* prevent or block the achievement of a given goal.
‡ A goal *constraint* places a condition on the completion of a goal.
§ "A *refutable description* says something precise about the world that could be refuted by a counter example" [47].

covering every case or considering all the factors. There are many ways in which goals can fail; this step requires analysts to consider such possibilities. Goal obstacle analysis thus facilitates the identification of exception cases for purposes of goal operationalization and guiding the identification of new, additional goals.

Goal obstacles are identified by analyzing statements that illustrate an example of a goal being blocked by another goal or conditions which prevent its completion. Figure 4.7 shows a few key words which have been observed to be helpful in pointing to candidate obstacles.

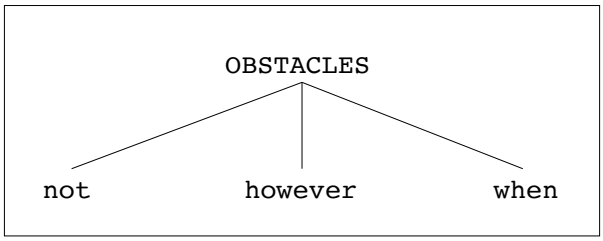

Figure 4.7. Some Useful Key Words for Identifying Obstacles

Obstacles can also be identified by asking *"What other goal or condition does this goal depend on?"* The answer to this question is then worded to emphasize the state as true, thereby denoting a goal obstacle. Example 4.13 demonstrates how obstacles are identified by considering the conditions placed upon a goal.

Example 4.13 Goal obstacles may be identified in parallel to goal identification. For example, consider the Process Scenario below (NLD #5) in which a statement that imposed a condition on an identified goal provides insight into a potential obstacle.

NLD #5: ... *The individual is then certified. However, the certification is only useful when it appears in the Training System Directorate (TSD) database which is used to ensure that the qualifications of professionals meet the requirements of their positions, and further qualifies and prioritizes people for additional training.*

The statement above provides an example of an event that may prevent the employee from being able to qualify for additional training. In Table 4.10, G_{18} (`Certification status improved`) may be blocked by the obstacle identified from NLD #5: `TSD database not updated with certification status.` (Career Track Training System; Chapter 3.2).

Table 4.10. Goal Obstacles Extracted From NLD #5

Goals	Agent	Goal Obstacles
G_{18} : Certification status improved	employee	1. TSD database not updated with certification status

Other possible goal obstacles may be considered and identified by asking: *"Can the agent responsible for a goal fail to achieve the goal?"*, *"Can the failure of another goal to be completed cause this goal to be blocked?" "If this goal is blocked, what are the consequences?"*, and *"What other goal(s) or condition(s) does this goal depend on?"*

Example 4.14 Consider G_4 (`Career portfolio routed to DTM`) in Table 4.11. The completion of this goal depends on whether or not the supervisor's concurrence was obtained. Thus, `Supervisor's concurrence not obtained` is an obstacle to G_4 (Career Track Training System; Chapter 3.2).

Table 4.11. General Goal Obstacle For G_4 Identified From NLD #5

Goals	Agent	Goal Obstacles
G_4 : Career portfolio routed to DTM	employee	1. Supervisor's concurrence not obtained

For each goal, the normal first case goal obstacle is specified by simply negating the verb in the goal name [60]. These are considered general *goal failure* (G) obstacles because they denote basic goal failure (expressed $\neg G_i$). Each general failure obstacle must then be analyzed to consider other possible obstacles (See Example 4.18). When a goal having a precedence relation is obstructed because the precedence goal fails, it is considered a *prerequisite failure* (P) obstacle (expressed $G_1 \not\leq_P G_2$) (see Example 4.15). When a goal fails because the responsible agent is irresponsible, it is considered an *agent failure* (A) obstacle (expressed $G_1 \not\leq_A G_2$) (see Example 4.16). In this case, the irresponsible party must be tracked down and held accountable. When a goal fails because the goal that it holds a contract relation with fails, it is considered a *contract failure* (C) obstacle. Recall that a *contract* relation exists between goals G_1 and G_2, if goal G_2 must be achieved when goal G_1 occurs (expressed $G_1 \rightarrow G_2$). A contract obstacle is thus expressed $G_1 \not\rightarrow G_2$ (see Example 4.17).

Example 4.15 (Prerequisite Goal Failure.) Consider goal G_8 (`Qualified personnel identified`) in Table 4.12, in order to identify those employees that qualify for training courses, the employee course preferences must be made available. Thus, goal obstacle #2 (`Preferences not made available`) is an example of a prerequisite failure for goal G_8. This prerequisite failure is expressed as $G_6 \not\leq_P G_8$ because if G_6 fails then G_8 cannot

occur.

Table 4.12. Prerequisite Failure Obstacle Example

Goals	Goal Obstacles
G_6 : Preferences made available	1. Preferences not made available (G)
G_8 : Qualified personnel identified	1. Qualified personnel not identified (G) 2. IPs not made available (P)

Example 4.16 (Agent Failure Obstacle) Consider goal G_{16} (`Approval granted`) in Table 4.13. In the event that a goal obstacle denotes more than one type of obstacle, it may be refined by decomposing it into separate obstacles. For example, `Approval not granted` may be either a general obstacle or an agent obstacle. The goal obstacle for G_{17} (`Certification not granted`) can be either an agent failure or a prerequisite failure. It can be further decomposed to differentiate the agent failure as a separate obstacle by asking *"Can the failure of another goal to complete cause this goal to be blocked?"* If the employee fails the course, then certification cannot be granted. Thus, an additional obstacle must be specified for G_{17}: `Employee fails course`.

Table 4.13. Agent Failure Obstacles Example

Goals	Agents	Goal Obstacles
G_{16} : Approval granted	AFB	1. Approval not granted (A,G)
G_{17} : Certification granted	AFB	1. Certification not granted (A,G,P) 2. Employee fails course

Example 4.17 (Contract Failure Obstacle) Obstacles may be caused by a contract failure. In Table 4.14 there exists a contractual relation between G_{11} (`Course completed`) and G_{12} (`Skills improved`). An employee's skills are improved whenever he or she completes a course. However, if the employee does not complete the course (e.g. G_{11} `: Employee fails course`), then the employee does not improve their skills to the degree necessary for satisfaction of the goal. Thus, if G_{11} fails, then G_{12} also fails. This is expressed as $G_{11} \not\Rightarrow G_{12}$.

Table 4.14. Contract Failure Goal Obstacles Example

Goals	Goal Obstacles	Scenarios
G_{11} : Course completed	1. Course not completed (G)	1.a Empl. drops out of course 1.b Empl. never enrolls in course 1.c Empl. fails course
G_{12} : Skills improved	1. Skills not improved (G)	1.a Course not taken 1.b Course not completed 1.c Empl. fails course

Example 4.18 (Normal First Case Obstacle) It may be the case that a normal first case goal obstacle is implicitly an agent failure. For example, G_{13} (`Proof of course completion submitted`) in Table 4.15 will more than likely be blocked due to a failure to achieve the goal by the responsible agent. In cases such as these, analysts must then consider reasons that could prevent the agent from achieving the goal. A prerequisite failure for G_{13} such as `Course not completed` could then be identified and denoted as $G_{11} \not\leqslant G_{13}$. Trivial obstacles force analysts and stakeholders to consider specific cases that must be handled due to activities which prevent goal completion.

Table 4.15. Normal First Case Failure Obstacles

Goals	Goal Obstacles	Scenarios
G_{11} : Course completed	1. Course not completed (G)	1.a Empl. drops out of course 1.b Empl. never enrolls in course 1.c Empl. fails course
G_{13} : Proof of course completion submitted	1. Course completion proof not submitted (A) 2. Course not completed (P)	1.a Course completion proof not available 1.b Employee didn't complete course

Once the goal obstacles are specified, analysts must consider the possible scenarios that are likely for each obstacle. Scenario construction is discussed in the following section.

Constructing Scenarios

The ways in which goals can fail are identified during goal obstacle analysis. The objective of this activity is to elaborate this information further via scenario analysis. Scenarios* offer a natural way to describe special, exceptional circumstances. Scenario analysis permits the consideration of alternative possible operationalizations of goals for the identification of the most plausible solutions.

Scenarios are the most creative artifact of the analysis process and play a major role in discovering goals and thus system requirements [7]. Scenarios denote concrete circumstances under which a goal may fail, helping analysts uncover hidden goals or issues needing further resolution that might otherwise go unnoticed or be overlooked. Decomposition, as described in [7] enables more effective generation of scenarios to assist analysts in acquiring and validating requirements, thereby supporting the process of refining goals. When goal

Scenarios are behavioral descriptions of a system and its environment arising from restricted situations.

priorities change, scenarios facilitate the evaluation of these new priorities. Obstacles are thus elaborated via scenarios. Scenario analysis is also useful for determining the post-conditions of different behaviors and goals (i.e., what happens if a goal isn't achieved?). By identifying obstacles that block or prevent goals, a greater degree of coverage can be achieved in a requirements specification.

Scenarios are identified by considering the goals and goal obstacles previously identified to determine the reasons why and the circumstances under which a goal may be completed or can fail. By asking *"Why?"* and *"What happens if this goal isn't achieved?"* scenarios can be identified that address why a goal failed or what the consequences are if the goal should fail. Initially, the normal first case obstacles are considered and possible scenarios are defined for each one. This is done for each obstacle by asking *"What are the circumstances under which this obstacle can occur?"* *"Why did this obstacle occur?"* and *"Why was this goal not achieved?"* Answers to these questions facilitate the identification of scenarios that address why a goal failed or what the consequences are if the goal fails. Scenario identification and construction provides a systematic way to find abnormal cases so that exceptions may be specified later.

Example 4.19 *Scenario analysis allows analysts to uncover hidden requirements.* Consider a vacation/sick leave hours tracking system in which employees must submit their hours to the Financial Services Office (FSO) personnel on a monthly basis. Figure 4.8 shows two obstacles for the goal `Vacation hours record updated` and the corresponding scenarios identified for each obstacle.

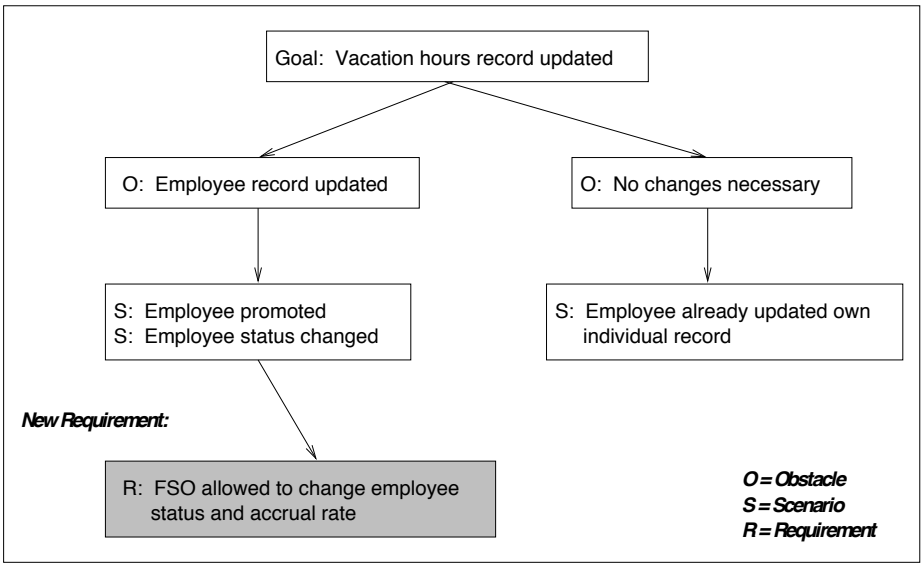

Figure 4.8. Uncovering New Requirements for Vacation/Sick Leave System

The obstacle on the right side of Figure 4.8 represents a normal goal obstacle (e.g. the employee has already updated his/her hours and thus has no need to update their vacation hours). However, the two scenarios (`Employee promoted` and `Employee status changed`) that correspond to the obstacle (`Employee record updated`) on the left side of the figure require an action to be taken by authorized personnel in the FSO. When an employee is promoted, their accrual rate changes. If an employee reduces their workload from full-time to half-time status, this, too, requires a change in the employee's accrual rate. This analysis of the obstacles via consideration of possible scenarios yields insight into who is authorized to modify information pertaining to employee accrual rates and employment status. Clearly employees cannot be granted the authorization to change their own accrual rates. This responsibility must be assigned to an agent (in this case the FSO). This type of backtracking analysis to find the causality in scenarios highlights the issues related to

responsibility and demonstrates how analysts are able to uncover new requirements such as the new requirement (as shown in Figure 4.8) by constructing scenarios for the previously specified obstacles. The building of causal event models in scenarios for safety critical systems may be rather complex whereas in information systems they are generally less necessary; the process of building causal event models is up to the analyst.

Example 4.20 Consider G_7 (`Available course slots announced`) in Table 4.16 and its corollary normal first case obstacle G_{17} `Available course slots not announced`. This obstacle may be analyzed by asking *"Why did this obstacle occur?"*, giving rise to the possibility that no slots are available (G_7 obstacle #2). Given the obstacle `No slots available`, it is possible to investigate the circumstances leading to this occurrence by asking *"What are the circumstances under which this obstacle can occur?"* Such an analysis yields the identification of two scenarios for obstacle #2: `2.a) All courses closed (max capacity reached)` and `2.b) Course cancelled (no slots available)`.

Table 4.16. Obstacle and Scenario Analysis Example

Goals	Goal Obstacles	Scenarios
G_7 : Available course slots announced	1. Available course slots not announced 2.b Course cancelled 3. Empl. course prefs not available	2.a All courses closed (max capacity reached) 2. No slots available (no slots available)

It is not sufficient for analysts to simply specify possible scenarios. Scenarios are only useful if they are subsequently used to consider the possible ways in which the system could (and should) respond given an exception such as `max capacity reached`. For example, will

overloads be allowed? If there is sufficient interest in a particular course which is closed, will another section be opened? Scenarios lead analysts to consider such options and provides a mechanism for reasoning about possible alternatives.

Table 4.17. Scenarios to Uncover Potentially Overlooked Issues

Goals	Goal Obstacles	Scenarios
G_{14}: Submitted paperwork reviewed	1. Submitted paperwork not reviewed	1.a Paperwork submitted late 1.b Paperwork not complete 1.c Paperwork not received

Example 4.21 Scenarios may help uncover hidden goals, issues or goal obstacles. For example, consider the goal G_{14} (`Submitted paperwork reviewed`) in Table 4.17. The first normal case obstacle for this goal is `Submitted paperwork not reviewed`. There are several possible circumstances leading to the occurrence of such a goal obstacle: `Paperwork submitted late`, `Paperwork not complete`, and `Paperwork not received`. By identifying these scenarios, it is possible to uncover potential issues which may have otherwise been overlooked. For the system to be effective, it must *know* how to respond to each of the scenarios. For example, if the paperwork is submitted late, what are the consequences? The system must 'know' whether or not to accept the paperwork late without penalty or whether, for example, a late fine must be administered before the paperwork can be processed.

Example 4.22 Scenario analysis is also useful for determining the postconditions of different behaviors and goals. For example, by asking *"What happens if this goal isn't achieved?"* it is possible to identify the potential postconditions for each goal. Consider goal G_9 (`Course & personnel matched`) in Table 4.18. If this goal is not achieved, employees will be unable to take a course they specified in their preference lists, and consequently will not improve their skills.

Table 4.18. Using Scenarios to Determine Postconditions

Goals	Goal Obstacles	Scenarios
G_9 : Course & personnel matched	1. Course & personnel not matched	1.a No course available 1.b Empl. awaiting certification to qualify 1.c No qualified personnel available

Example 4.23 Exception cases may be identified by considering the possible scenarios. Figure 4.9 shows several scenarios for the case in which an employee submits their vacation/sick leave hours at the end of the month. The scenarios on the left side of the figure represent the ideal case when an employee successfully submits their hours without incidence. The scenarios in the shaded boxes on the right side of the figures correspond to the exception case during which an employee fails to submit their hours. It may be the case that the employee forgot to submit his or her hours because (`Employee not reminded`). By considering the circumstances that may have prevented an employee from being reminded, several possibilities become evident. In the event the employee is out of town, the analyst must consider how the system should react if the employee returns to the office after the submit-

tal deadline. If an employee is away for an extended period of time (e.g. a professor who is away on sabbatical for a year) the system could respond in various ways; for example, the system could refrain from sending reminders to employees on sabbatical or employees could be allowed to request a stop on reminders being sent until a certain specified date (e.g. their return date).

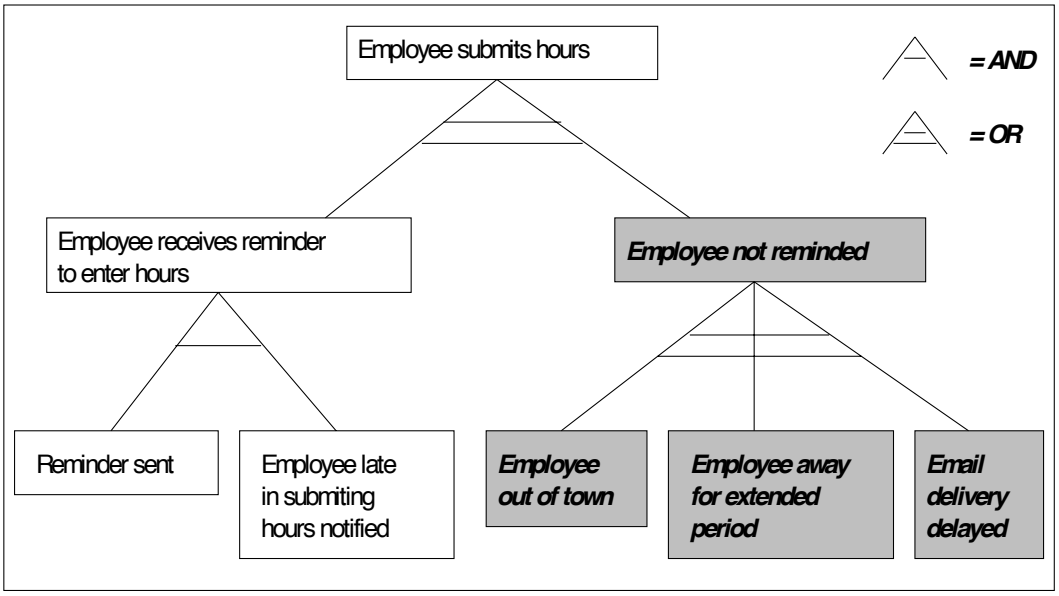

Figure 4.9. Considering Exceptional Cases

During scenario analysis, stakeholder participation is especially appropriate and encouraged. Stakeholders not only bring to the table a wealth of knowledge and expertise about their particular domain, they also offer a rich perspective due to their personal experiences and recollection of special cases. It is thus beneficial to have domain experts construct scenarios, given that they are likely to think of more possible ways in which goals

can fail than is an analyst who may be unfamiliar with the domain and/or the organization. Although stakeholder participation is critical at this stage, analysts play an important role in guiding the stakeholders by asking questions which impose a structured inquiry-driven analysis.

During the analysis process, it is helpful for analysts to employ an inquiry-driven approach to communicate with stakeholders. Guidelines for when and how to ask these types of questions are provided in Chapter 5. Additionally, Chapter 5 provides guidelines and heuristics for how and when to construct scenarios. Scenarios need not only be constructed for goal obstacles. Scenarios also facilitate the consideration of assumptions and issues pertaining to the goals themselves. During obstacle and scenario analysis, it is possible to also identify goal constraints. The following section discusses techniques for identifying goal constraints.

Identifying Constraints

The objective of this step is to identify any constraints that exist for goal completion. Constraints provide information regarding circumstances that must exist or conditions that must be met for a given goal to be completed. Constraints are identified by analyzing each textual description fragment to determine if it states a requirement that must be met in order for a given goal to be completed or if it imposes some constraint on the goal(s). Statements that seem to be independent of other goals or requirements should also be stated as constraints.

As a general rule, constraints may be identified by looking for dependency relations and by searching for temporal connectives, such as *during*, *before* and *after*, or variants thereof. Figure 4.10 shows a few key words which have been observed to be helpful in pointing to candidate constraints.

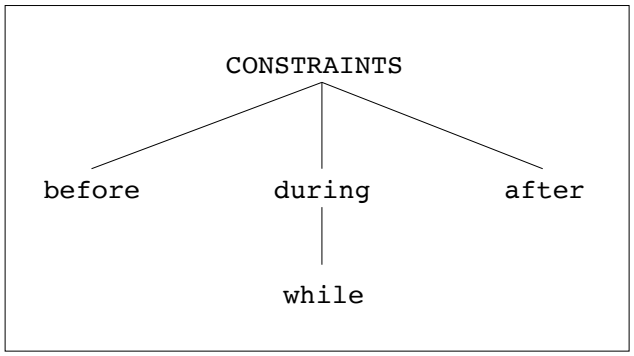

Figure 4.10. Some Useful Key Words for Identifying Constraints

Example 4.24 Temporal connectives enable analysts to argue about *when* statements are true or when goals can be completed. Consider the goal `Meeting arranged` in the meeting scheduler system with the constraint: Meeting room must be available *during* the meeting date/time.

Example 4.25 Consider NLD #3 on page 81, the word *qualifies* is a key indicator that a condition must be met. Before an employee may advance to a new certification level, the last course taken must officially qualify him or her for advancement. Table 4.19 shows two constraints that may be extracted from NLD #3.

Table 4.19. Constraints Extracted from NLD #3

No.	Constraints
1	Course must qualify employee to advance to another level
2	Certification enables employee to move up to a higher certification level

Once the goals have been specified and elaborated to the greatest extent possible, this information must be operationalized and translated into the actual natural languages expression of the requirements in the SRD. The following section explains the operationalization process.

Operationalization of Goals Into Requirements

Goals are a logical organizing mechanism for the incorporation of software requirements information. During operationalization, the actions described by stakeholders and extracted from available documentation are related back to the goals. A fairly formal way to represent both the goals and the actions to which they are mapped is needed. In GBRAM the operationalized goals, responsible agents, stakeholders, constraints, obstacles, and scenarios are mapped onto actions which are ultimately consolidated into a set of goal schemas*. The resulting artifact, while not formal in the strict sense, provides a textual representation of system requirements organized according to system goals as prescribed by the topography. This section addresses the translation of the goal information into a software requirements document by consolidating the results of the analysis effort into a set of goal schemas.

* *Goal schemas* are models which specify the relationships between goals and agents in terms of events that cause a change of state. In a goal schema, goals are specified as events in terms of pre- and post-conditions.

Although a wide variety of representations are available [48,73,75,11], GBRAM employs an informal model-based style similar to the FUSION method [19]. The goal schema syntax, which borrows from the FUSION operation model, is presented in the following subsection.

4.4 Goal Schemas

The goal schema specifies the relationships between goals and agents in terms of events that cause a change of state. Goals and their operationalizations (actions) are specified as events in terms of pre-conditions* and post-conditions†. This section defines the syntax for specifying goal schemas and their associated actions. An explanation is provided of how goals may be expressed in a series of goal schemas that can be mapped to a software requirements document and organized according to the goal topography.

Schema Syntax

The goal model incorporates all of the information acquired during goal analysis and elaboration. It is expressed as a series of schemas. There must be one schema for each goal. Figure 4.11 presents the schema syntax for goal models used to specify goals. Descriptions are provided, following Figure 4.11, to define the semantics of each clause in the schema. It should be noted that it is possible for some clauses in the schema to remain unspecified (e.g. Constraints and Subgoals).

*The *precondition* characterizes the conditions under which the goal may be achieved.
†The *postcondition* characterizes the state of the system once the goal is completed.

Goal:	*Name*
Type:	*Name*
Description:	*Text*
Action:	*Name*
Agent:	*Name(s)*
Stakeholders:	*Name(s)*
Constraints:	*Items*
Obstacles:	*Items*
Preconditions:	*Condition*
Postconditions:	*Condition*
Subgoals:	*Name(s)*

Figure 4.11. Schema Syntax for Goal Models

- **Goal:** *Name*

  A goal is an objective that the system must meet. Goals may block other goals and may be decomposed into subgoals and ultimately operationalized. *Name* is the unique identifier for the goal. Intuitive and informal identifiers are employed. Goals are stated to indicate a desired state. For example, goals should not be worded as an action (`Announce available course slots`), rather the goal should be worded as `Available course slots announced`.

- **Type:** *Name*

  Goals are classified according to the behavior they require: to achieve some state or maintain some condition/state. *Name* can take on one of two identifiers: Achievement or Maintenance. An achievement goal is satisfied when the target condition is attained. A maintenance goal is satisfied as long its target condition remains true.

- **Description:** *Text*

  The description is an informal textual description of the goal.

- **Action:** *Name*

  Action is the name of the goal operationalization. It is the behavior that must take place for the goal to be achieved.

- **Agent:** *Name*

  The agent is responsible for completing or achieving a goal. *Name* is the unique identifier that corresponds to the responsible agent.

- **Stakeholder(s):** *Name(s)*

  The stakeholders are those participants that claim an interest in the completion of the goal. *Name* is a list of unique identifiers that correspond to the stakeholder(s).

- **Constraints:** *Items*

  Items correspond to the constraints under which an objective or goal must be achieved. A constraint specifies some requirement or condition that must be met in order for a given goal to be completed. Items are informal textual descriptions of certain requirements and conditions that must be met.

- **Obstacles:** *Items*

  Items is a list of all obstacles that may block the completion of a goal. A goal obstacle may be either a general failure, an agent failure, a contract failure, or a prerequisite failure.

- **Preconditions:** *Condition*

  A precondition must exist for the achievement of a goal to be possible. For example, in the career training example, before the goal `Certification granted` can be achieved, an employee must first complete a qualifying course (`Course completed`). *Condition* is a predicate (list of clauses) defining the precondition. For the predicate to be satisfied, each clause must be true. The predicate is false if any of the clauses is false. If no condition is declared, then the goal is not dependent on any other agent, entity or goal.

- **Postconditions:** *Condition*

  Postconditions are true after a goal has been achieved or completed. They relate to preconditions (the state of the system before the goal is completed).

Each function is expressed in the form of an operational definition. There is at least one operational definition for each goal; in some cases, two operational definitions should be provided for each goal. The first operational definition specifies the user (stakeholder) point of view, whereas the second operational definition specifies the system point of view. Additionally, it is possible that there may be more than one action per goal. Figure 4.12 defines the semantics of each clause in an operational definition.

```
action  actionName
    agent:      agentName
    reads:      item(s)
    changes:    item(s)
    assumes:    preCondition
    result:     postCondition
end actionName
```

Figure 4.12. Operational Definition Syntax

- **action:** *actionName*

  The *actionName* is the name of the function that the system allows the user to invoke and/or perform.

- **agent:** *agentName*

  agentName is the party responsible for invoking the function.

- **reads:** *item(s)*

  items(s) are the sources and input information required by the agent to perform the function.

- **changes:** *item(s)*

  item(s) are the sources or input information associated with the function in terms of modifications that result to the system or data within the system due to user inputs.

- **assumes:** *preCondition*

  preCondition denotes some condition that must exist for the function to be completed.

- **result:** *postCondition*

  postCondition is the condition that is true after the function is completed.

The notation of Figure 4.12 is not a formal notation; rather, it captures the intuition of the data modeling. The *assumes* and *results* clause may be expressed in some formal syntax such as FUSION. GBRAM assumes that another analysis method, such as OOD [18] or FUSION [19], is performed so that these specifications may be imported into GBRAM.

The following examples show how goals are specified using the goal schema model. The first example demonstrates how a goal can be further refined during goal schema construction by referring to the maintenance goals. Figure 4.13 shows the goal schema model for G_7 (`Available course slots announced`).

Goal:	Available course slots announced
Type:	Achievement
Description:	HRD must announce the courses and the number of slots available so that qualified personnel can be identified, matched, and notified.
Action:	Announce course slots
Agent:	HRD
Stakeholders:	HRD, employee, TSD
Constraints:	
Obstacles:	1. No slots available 2. Employee prefs not available 3. Al courses closed (max capacity reached) 4. Course cancelled (no slots available)
Preconditions:	1. Employees course prefs ready 2. Preferences made available
Postconditions:	Employee an course slot matched
Subgoals:	Qualifying training course slots announced Position training course slots announced

Figure 4.13. Schema for G_7 : Available course slots announced

The two subgoals for G_7 (shown in Figure 4.13) may be identified by considering goals two maintenance goals. G_6 and G_7 specify that there are actually two different types of training courses offered to AFB employees: qualifying training and position training. The action associated with goal G_7 is defined in Figure 4.14.

```
action  AnnounceCourseSlots
    agent:      HRD
    reads:      course.slots
    changes:
    assumes:    course.slots available
                employee.prefs ready
                employee.prefs made available
    result:     course.slots announced
end AnnounceCourseSlots
```

Figure 4.14. Operational Definition for G_7 : Available course slots announced

In the specification of Figure 4.14, `action` is the action which should be taken to achieve the goal (G_7). In order to announce the available course slots, the environment must supply `course.slots`. Since this particular action merely involves the broadcasting of information (e.g. the available course slots) `course.slots` is not modified by the action. The `assumes` clause shows what the environment is responsible for ensuring that course slots are available before the action (`AnnounceCourseSlots`) occurs.

The next example demonstrates how goals may be decomposed (and reorganized) into subgoals by analyzing dependency relations. Each subgoal should map to one action. Thus, if a subgoal appears to map to several actions, it should be further decomposed and refined. Heuristics and guidelines for this decomposition process are provided in Chapter 5. Figure 4.15 shows the goal schema model for G_9 (`Course & personnel matched`).

The action associated with goal G_9 is defined as shown in Figure 4.16. In this specification, `action` is the operation required in order to achieve goal G_9 (`Course and personnel matched`). For qualified employees to be matched with the available course slots in the appropriate courses, the environment must supply information including: the available course

Goal:	Course & personnel matched
Type:	Achievement
Description:	Employees are matched to a course based on: the course preferences they specify, the courses the employee has previously taken, and the employee's certification status.
Action:	Match course & personnel
Agent:	HRD
Stakeholders:	HRD, employee, TSD
Constraints:	1. Course Ɛ employee course prefs 2. Employee must be certified at prerequisite level 3. Course must qualify employee for higher certification
Obstacles:	1. No course available 2. Employee awaiting certification to qualify 3. No qualified personnel identified
Preconditions:	1. Qualified personnel identified 2. Available course slots announced
Postconditions:	1. Trainee notified 2. Employee enrolled in class
Subgoals:	G_7: Available course slots announced G_8: Qualified personnel identified

Figure 4.15. Schema for G_9: Course and personnel matched

slots (`course.slots`) and the course preferences for each employee (`employee.prefs`). The `assumes` clause shows what the environment is responsible for ensuring before the action (`MatchCourseAndPersonnel`) can occur.

```
action  MatchCourseAndPersonnel
    agent:      HRD
    reads:      course.slots
                employee.prefs
    changes:    course.slots
    assumes:    course.slots announced
                employee.prefs ready
                employee.prefs made available
    result:     course.slots  matches  employee.prefs
                ∀ c: course ∋ c.slots  matches employee.prefs
                 ➛ c.slots = c.slots - slot
end AnnounceCourseSlots
```

Figure 4.16. Operational Definition for G_9 : Available course slots announced

The predicates `announced`, `ready` and `available` must be true. As a result of the action, goal G_9 is achieved. The qualified personnel are matched with available course slots according to `employee.prefs`. Another result is that each time an employee is matched with a course, there is one less slot in that course. From the postconditions in the goal schema for G_9 in Figure 4.15, it is clear that once the goal is completed, the employee can be notified and ultimately enroll in the course. By considering what the pre and post conditions are for each goal, it is possible to identify goals that may have been previously overlooked. For example, the postconditions for G_9 includes a goal which had previously not been stated: `Employee enrolled in class`. Clearly an employee must be enrolled in

a course to attend and complete it. Finally, goal G_9 shares a contract relation with goals G_7 and G_8 (`Qualified personnel identified`). Since $(G_7 < G_8) \rightarrow G_9$, we classify goals G_7 and G_8 as subgoals of G_9.

While declaration of objects and their constraints is not supported by GBRAM, analysts must be able to write expressions over terms (e.g. `ready` in Figure 4.14) to be able to specify pre- and post-conditions. In Figure 4.16, `slots` and `course` are declared as entities in the system. These entities were constructed outside of GBRAM using an information modeling method readily available to analysts such as FUSION. Thus, the examples provided herein assume that the data model is produced using the data modeling approach of the analysts' choice.

The next example extends the goal schema further than the previous two examples by actually specifying one of the goal obstacles. Figure 4.17 shows the goal schema model for G_{11} (`Course completed`).

The action associated with goal G_{11} is defined as shown in Figure 4.18. In this action specification, `action` is the operation required to achieve goal G_{11}. The responsible agent for this action is `employee`. The assumes clause specifies that it is assumed that the employee has the appropriate prerequisite skills to take the course and that the employee was matched to a slot in the course. The result of the action is that the employee completes the course and the employee's skills are improved.

Investigation indicates that it is reasonable to consider the appropriateness of using goal schema models for further specification of goal obstacles; for example, Obstacle #1 (`Employee drops out of course`) for G_{11} (see Figure 4.17 on page 121). The action is defined as shown in Figure 4.19.

Goal:	Course completed
Type:	Achievement
Description:	In order for an employee to improve their skills they take training courses. To improve their certification status, they must take courses which specifically qualify them for certification.
Action:	CompleteCourse
Agent:	employee
Stakeholders:	employee
Constraints:	Course must improve skills & certification status
Obstacles:	1. Employee drops out of course 2. Employee never enrolls in course 3. Employee fails course
Preconditions:	1. Course & personnel matched 2. Trainee notified
Postconditions:	1. Course completed 2. Skills improved
Subgoals:	

Figure 4.17. Schema for G_{11}: Course Completed

```
action   CompleteCourse
    agent:      employee
    reads:      course.slots
    changes:    employee.skills
    assumes:    employee.skills
                course.slots  match  employee.prefs
                employee notified
    result:     course completed
                employee.skills improved
end CompleteCourse
```

Figure 4.18. Operational Definition for G_{11} : Course completed

```
action  EmployeeDropsCourse
    agent:      employee
    reads:      course.slots
    changes:    course.slots
                employee.CP
                employee.prefs
                employee.skills
    assumes:    employee unstable
    result:     course not completed
                employee.skills unchanged
                course'.slots = course.slot + {slot}
end EmployeeDropsCourse
```

Figure 4.19. Operational Definition for Obstacle for G_{11} : Course completed

The process of specifying the action for this obstacle gives rise to several key issues. If an employee drops a course, then a slot in the course becomes available for another employee. However, this highlights the need for a drop/add deadline. For example, if the drop/add deadline is set for one week before the course begins, then the slot becomes available with enough time for another qualified employee to be identified and matched to the course slot. However, if the employee drops the course mid-term, even though the slot becomes physically available, it is too late for another employee to fill that slot and take the course. This example demonstrates that by specifying the actions associated with obstacles, it is possible to further elaborate the requirements for the system. In this case, a drop/add deadline is added in order to minimize the occurrence of a course being dropped mid-term, thereby preventing other employees from enrolling in the course.

4.5 Tool Support

Part of this effort involved the development of a tool to support the Goal-Based Requirements Analysis Method. The tool is presented here as an enabling technology rather than a major contribution of this thesis.

The Goal-Based Requirements Analysis Tool (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 geographic and time differences.

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

Users

It is assumed that the typical GBRAT user will be an experienced requirements engineer with a considerable working understanding of the Goal-Based Requirements Analysis Method, the World Wide Web (WWW), and Web-based applications. It is assumed 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 in order

to identify and specify the goals of the desired system. After the analyst has gathered all available information about the desired system, goals may be extracted from these information sources and specified using GBRAT.

System Features

GBRAT features enable users to create project repositories, specify goals, view goals from several perspectives, and order goals. The examples in this section which illustrate these features are taken from the electronic commerce Web server analysis. Several examples from this study are detailed to demonstrate how GBRAT enables analysts to easily identify synonymous goals and manage traceability via the Web.

Project Repositories

Goals concerning a given system are stored in a project repository. Each project repository has a specified project name and description, and the name of the analysts working on a given project. From within a specific project repository, analysts can create new goals or view the previously specified goals using three filters: the maintenance and achievement goal filter, the agent filter, and the goal hierarchy 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.

Goal Traceability

Gotel et. al. address the conundrum of requirements traceability among agents and artifacts [36] by modeling the dynamic contribution structures underlying requirements ar-

tifacts. 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 (Figure 4.20) to ensure that each goal can be traced back to its place (i.e., document) of origin. For example, this also enables analysts to easily identify goals which may have been extracted from more than one information source so that any similarities and differences may 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 Achievement & Maintenance Goals

Goals may be viewed by various filters in GBRAT. When achievement and maintenance goals are viewed, the goals are displayed alphabetically and a browser is provided, as shown in Figure 4.21. The goals in Figure 4.21 are achievement goals. By selecting a goal from the list in the left frame, users can view the selected goal's properties.

Viewing Goals by Agent

Analysts may wish to view the goals for which a particular agent is responsible. In GBRAT, the relevant goals that each agent is responsible for are displayed in the same format as described above. However, a different table is created for each agent. For example, Figure 4.22 shows a few of the achievement goals which a consortium member is responsible for in the electronic commerce web-server system. Recall that more than one agent can be associated with a goal. Experience with GBRAT has shown that when the same goal is identified from two different sources, the primary difference between the two goals

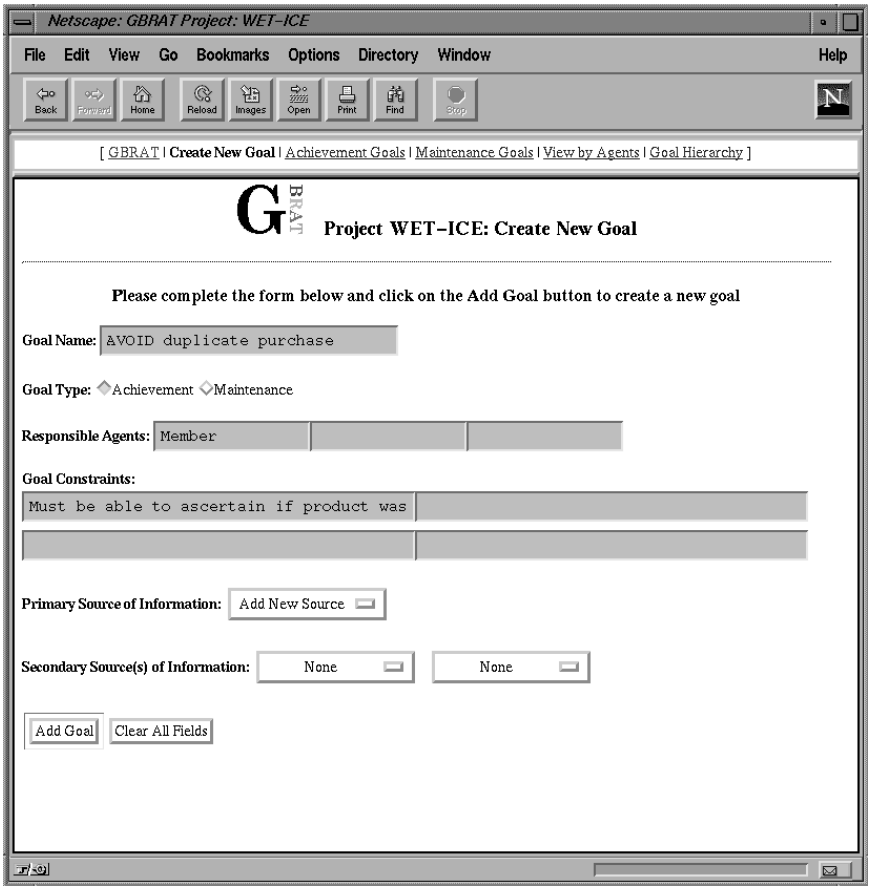

Figure 4.20. GBRAT Form for Creating Goals

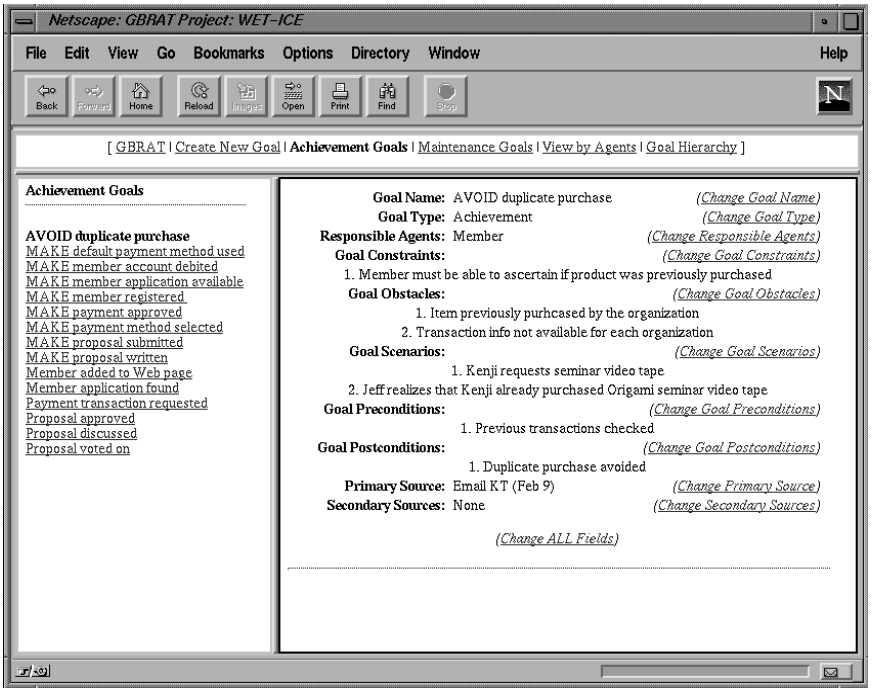

Figure 4.21. Viewing Achievement Goals

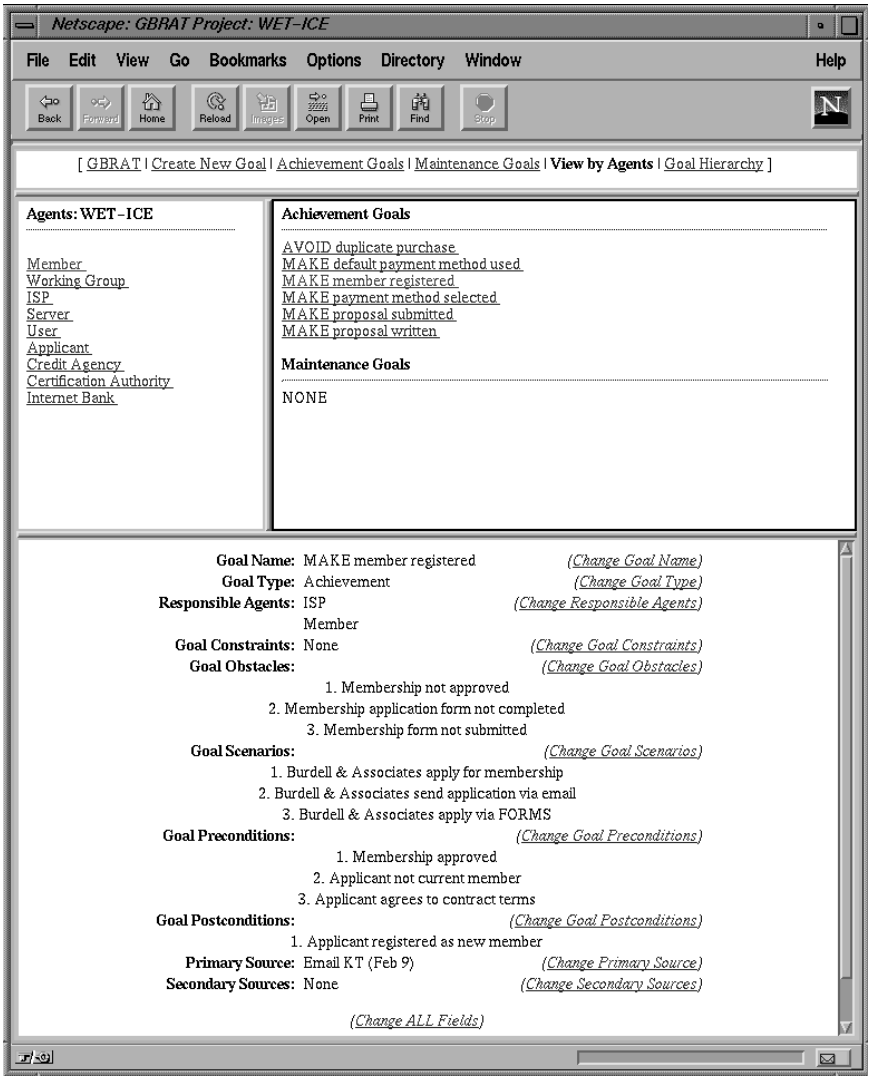

Figure 4.22. Viewing Goals by Agent

is often the responsible agents. Given this condition, GBRAT notifies the user when this duplication occurs and allows the user to merge the two goals into one goal with multiple responsible agents.

Viewing Goals by Precedence Relation

All achievement goals are related to the other goals in the system. As explained in Chapters 4 and 5, a *precedence* relation exists between goals G_1 and G_2, when goal G_1 must be completed before goal G_2. The main reason for organizing 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 to produce a total ordering of 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 4.23 shows the goal ordering produced by GBRAT based on the users' specifications. This view of the goals is helpful in that 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 Goal Hierarchy Relationships

Goal hierarchies are useful for representing relationships between goals and subgoals and for reasoning about goal relationships [7], [24]. GBRAT allows users to build a graph hierarchy to visualize the relationships among goals by employing a drag and drop interface. To show these goal relationships, an n-ary tree structure can be constructed. The name of the root node is the same as the project (repository) name. The tool supports any number of

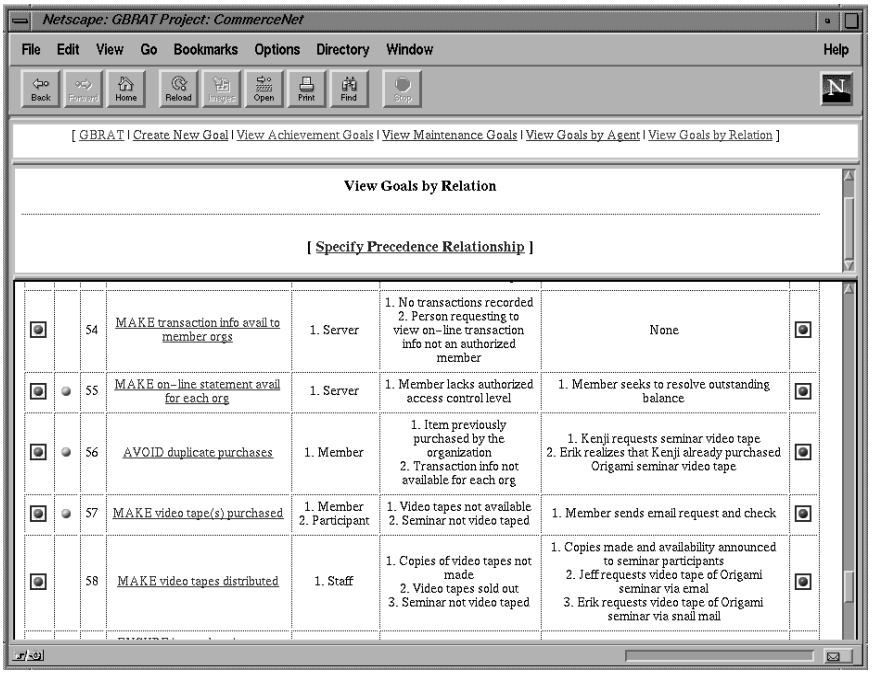

Figure 4.23. Viewing Goals by Precedence Relation

children at each level in the tree. This structure clarifies the relationship between subgoals and goals; when a particular goal is a subgoal of another goal, it must be completed as a prerequisite of its parent. Figure 4.24 shows the goal hierarchy tool. The goal list, at right, includes an item labeled 'OR' so that OR relationships among goals may be specified. For example, in Figure 4.24 the goal `MAKE member account debited` depends on one OR the other of its subgoals being completed. By default, AND relationships are assumed for subgoals, so hierarchies that reflect more complex requirements may be built using these AND and OR relationships.

To make the visualization as malleable as possible, the tool supports 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 goal priorities, dependencies and relationships. Each subtree in the hierarchy may be collapsed, allowing users to view the tree at various levels of abstraction. As shown in Figure 4.24, an arrow appears to the left of each subtree to collapse individual subtrees.

Implementation

The WWW provides a consistent user interface and the ability to incorporate a wide range of technologies and document types, allowing multiple users in different physical locations to access information via the Web by using Web browsers. These characteristics played a role in the decision to develop GBRAT as a Web-based application. GBRAT allows analysts working in different locations ready access to the same documents. The Netscape browser offers a consistent interface across different platforms, nonstandard HTML tags, and built-in security capabilities that enabled us to limit access to registered GBRAT users.

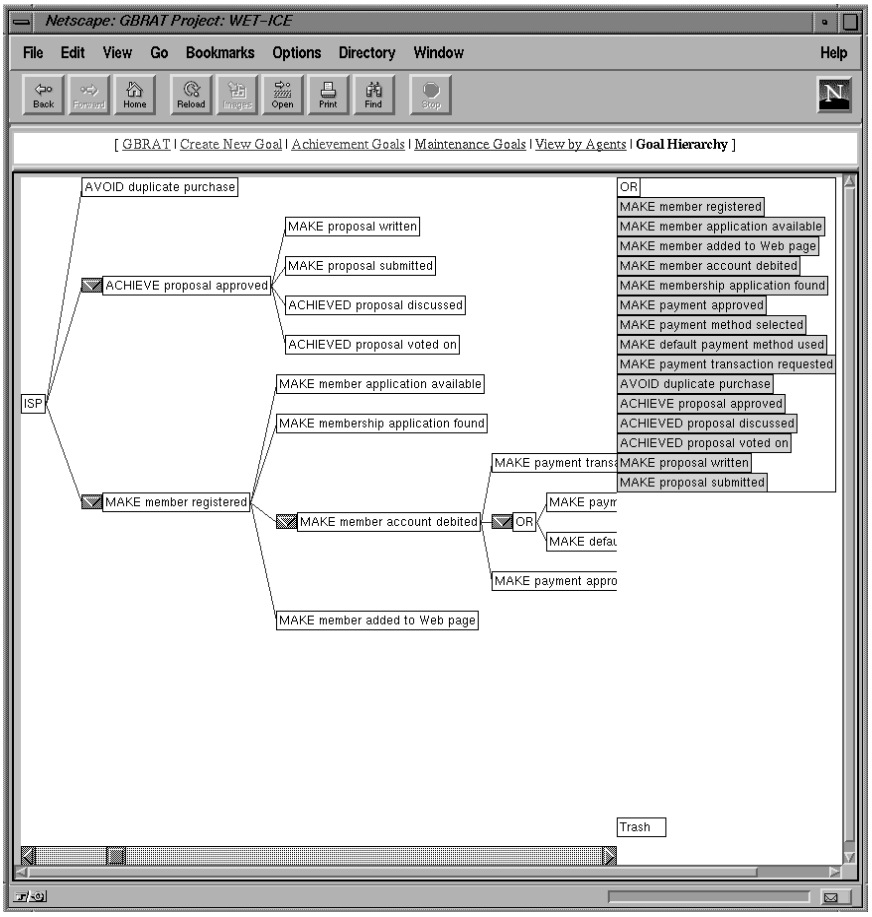

Figure 4.24. Goal Hierarchy

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 caching capability of WWW browsers often requires repeated reloading of modified pages. However, by using Perl scripts to retrieve information from the goal database, pages are dynamically generated, providing the user with the most current information. 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 (Figures 4.21, 4.22, and 4.23).

Along with Perl CGI scripts, a Java applet is used for specifying hierarchical relationships among goals. The applet was created using SubArctic, a Java-based user-interface toolkit currently being developed at the Georgia Tech Graphics, Visualization and Usability center. This toolkit allowed for drag-and-drop interaction and the use of constraints for user-interface elements and layout.

4.6 Summary

The requirements developed by analysts require models which can be easily understood by the stakeholders. Typically, as the formalization of requirements progresses, the stakeholders' understanding of the models decreases due to the complexity of the resulting representations. The Goal-Based Requirements Analysis Method presented in this chapter provides appropriate representation mechanisms to enhance stakeholder comprehension and facilitate communication between analysts and stakeholders while concomitantly offering a reasonable representation which can be easily translated from the language and conventions

of the stakeholder's 'workplace' to the language and conventions of analysts and developers. Chapter 5 discusses the heuristics which guide analysts in the timely and knowledgeable application and use of the techniques presented in this chapter.