Goal Identification and Refinement in the Specification of Software-Based Information Systems

A THESIS

Presented to

The Academic Faculty

By

Ana I. Antón

In Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy in Computer Science

Georgia Institute of Technology

June 1997

Goal Identification and Refinement

in the Specification of

Software-Based Information Systems

Approved:

Peter A. Freeman, Co-Chairman

Colin Potts, Co-Chairman

Gregory Abowd

W. Michael McCracken

Alistair Sutcliffe

Date approved by Chairman _____

Dedication

With love and gratitude

to my parents Blanquita and Manolo:

for their continued support and inspiration;

for teaching me that no goal is too lofty given perseverance and dedication;

for their example of courage, integrity, and strength of character;

for showing me by example the importance of

standing up for what you believe is right no matter the risk;

for being my best friends; and

for instilling in me the values that have made me the person I am today.

Acknowledgements

It seems to me shallow and arrogant for any man in these times to claim he is completely self-made, that he owes all his success to his own unaided efforts. Many hands and hearts and minds generally contribute to anyone's notable achievements.

Walt Disney

One has few opportunities to publicly thank those who have influenced them. This thesis would not have been possible without the help and support over the years of the following faculty members, colleagues, friends, and family.

I must begin by thanking my Ph.D. committee members; in particular, my co-advisors Dr. Peter A. Freeman and Dr. Colin Potts. Dr. Peter Freeman encouraged me to enter the doctoral program. During my time as his student he frequently had more faith in me than I had in myself; he has always built my confidence and sustained faith in my abilities. Dr. Colin Potts' instinct for what constitutes good software engineering research has helped shape my approach to research. Both men have contributed immensely to shaping me as a person, a professional, and a researcher. Mike McCracken has served as a very active committee member, and has always been an excellent sounding board and friend. Thanks are also extended to Dr. Gregory Abowd and Dr. Alistair Sutcliffe for their contributions as committee members.

Dr. Kurt Eiselt and Dr. John Stasko deserve considerable thanks for providing advice, mentoring, and friendship. In the same vein, I wish to thank Dr. Jim Foley of MERL, Dr. Michael Thomas of Georgia Tech, and Dr. Reid Smith of SCR for their guidance and friendship.

A special thanks is also extended to my sponsor and excellent colleague, Dr. Kenji Takahashi, as well as Jeff Smith and Eric Battaler of NTT. Thanks to Darcy Painter and Eric Trevena for allowing me to conduct critical case studies which required a significant amount of their time. Also instrumental in the preparation of this thesis was my editor and good friend Kristen Thorvig.

Without the assistance of Dr. Adolfo Ponce de Leon for the past 19 years, I never would have made it through high school, college, and graduate school; he has always helped me behind the scenes without ever receiving due credit for the impact of his assistance.

By far the most inspirational individual I met during graduate school was Vice Admiral Richard H. Truly. This man is truly a giant. Richard has profoundly influenced me by way of his example, character, humor, and friendship. He taught me about leadership under stress, integrity, and courage. His advice to me as I began my transition from graduate student to professional was invaluable.

I wish to thank Dr. Pat Crecine for his friendship, mentoring, and for broadening my perspective of higher education. I also owe thanks to gus Baird and Dr. Bud Suddath for their influence during their time at Tech.

A special thanks to my office mates: my partner in crime, Eileen Kraemer, for making me laugh on a daily basis and for advising me at each stage of the Ph.D. program; and Kurt Stirewalt, the best whistler this side of the Mason-Dixon line. A moment of silence.

It was very rewarding to work with several students, particularly, the three students who built GBRAT: Eugene Liang, Roy Rodenstein, and Santiago Abraham. To my good friends Gwen Baker, Doug Bodner, David Cardoza, Lara Catledge, Peggy Eisenhauer, Allison Elliott, Susan Haglund, Yves Jean, Wallace Jones, Colleen Kehoe, Drew Kessler, Msgr. Don Kiernan, Marie and Ken Little, Fr. Richard Lopez, Jim Mason, Scott McCrickard, Melody Moore, Nancy and Scott Register, Paulo Santos, Alex Snoeren, Suzanne Suddath, David Skinner, Julie Swann, and Mrs. Pat Thomas: thank you.

Sincere thanks are given to CoC staff members: Cathy Beam, Cathy Dunahoo, and Catherine Gholson, as well as David Leonard, Pam Buffington, Uwanna Smith, Karen Carter, Bryan Rank, and Peter Wan for their technical support.

There's a very unique, special, and somewhat large group of friends that I wish to thank. Thanks for keeping an eye on me these last few years.

I thank God for enabling me to achieve all that I have achieved, and I thank the people who deserve the most acknowledgement and to whom I dedicate this dissertation: my parents.

A.M.D.G.

Contents

List of Tables

List of Figures

Glossary

- **Abstraction** is the process of extracting goals from different sources. It facilitates the organization of detail-intensive requirements information so that 'concrete phenomena' may be described in a more abstract manner. Stage [80] states that in some cases a high level of abstraction facilitates determining points of resemblance among similar situations.

- **Agents** are responsible for the completion and/or satisfaction of goals within an organization or system. For example, given an Electronic Meeting System (EMS), a meeting initiator is the agent responsible for calling, or initiating, a meeting. Given a course registration system, a student enrolling for a course is the agent responsible for registering for that course.

- **Constraints** place a condition on the achievement of a goal.

- An **Enterprise** is the business or organization for which the proposed system is intended.

- **Exceptions** are special or extraordinary circumstances that occur in the system.

- **Goals** are targets for achievement which provide a framework for the desired system. Goals are high level objectives of the business, organization, or system. They express the rationale for proposed systems and guide decisions at various levels within the enterprise. `Corporate profits maximized` is an example of a high-level enterprise goal. The two primary types of goals discussed in this thesis are achievement and maintenance goals.

  - *Achievement goals* are objectives of an enterprise or system. For example, a university course registration system may need to satisfy the goal of enrolling students in courses before the first day of class each semester. The object of the goal, identified by the stakeholders as the primary purpose of the system, is course registration. In general, achievement goals may be mapped to functional requirements.

  - *Maintenance goals* are those goals which are satisfied while their target condition remains constant or true. They tend to be operationalized as actions or constraints that prevent certain states from being reached. In general, maintenance goals map to nonfunctional requirements.

- **Goal dependency** relations exist between pairs of goals [60].

  - A *precedence* relation between goals G_1 and G_2, where goal G_1 must be completed before goal G_2 is expressed $G_1 < G_2$.

  - A *contract* relation between goals G_1 and G_2, where goal G_2 must be completed if goal G_1 occurs is expressed $G_1 \rightarrow G_2$.

- **Goal obstacles** prevent or block the achievement of a given goal [60]. Abstracting and identifying goal obstacles allows the consideration of possible methods of failure and the anticipation of exceptional cases.

- **Goal refinement** is the process of subdividing a set of goals into a logical subgrouping so that system requirements may be more easily understood, defined, and specified. This logical subgrouping consists of subgoals which must be met in order for the parent goal to be met.

- **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.

- **Operationalization** is the process of defining a goal with enough detail so that its subgoals have an operational definition.

- A **Precondition** is a milestone or intermediate goal which must be achieved before another (dependent) goal may be achieved. Precedence relations suggest candidate preconditions.

- A **Requirement** specifies how a goal should be accomplished by a proposed system. Requirements define the capabilities that a system must provide in order to satisfy the goals of stakeholders. An example of an operation requirement for the goal `Corporate profits maximized` is: `Customer calls should be handled in less than 5 minutes` with the theory that you can improve the cost/benefit to the corporation.

  - *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.

- **Scenarios** are behavioral descriptions of a system and its environment arising from restricted situations. They capture combinations of actions, exemplify behaviors and enable hidden needs to be uncovered. Scenarios are also useful for evaluating design alternatives and validating designs.

- A **Stakeholder** is anyone who claims an interest in a given enterprise or system. Stakeholders are those individuals who can share information regarding the proposed system, its implementation, or the problem domain.

Summary

This dissertation introduces the Goal-Based Requirements Analysis Method (GBRAM) for the identification and refinement of goals into operational requirements for software-based information systems. The method evolved as a result of its application to real world systems and processes, and was validated through its application to the redesign of a large commercial system. Further validation of the method was performed via an empirical evaluation. From these evaluations it was concluded that the method's straightforward, methodical approach to identifying system and enterprise goals and requirements suggests goal identification and refinement strategies and techniques through the inclusion of a set of heuristics, guidelines, and recurring question types. Four sets of heuristics are presented: identification heuristics, classification heuristics, refinement heuristics, and elaboration heuristics. The heuristics and guidelines are shown to be useful for identifying and analyzing specified goals and beneficial for the refinement of these goals. The heuristics and supporting inquiry include references to appropriate construction of scenarios and the process by which they should be discussed and analyzed. A set of recurring question types is presented to assist analysts in applying an inquiry-driven approach to goal-based analysis. Formative and summative case studies serve as both the origin and validation of the method presented.

CHAPTER 1

Introduction

If you don't understand the user's requirements, it doesn't matter how you code it.

Ed Yourdon

The core of this research is aimed at improving the early stages of the software development process in complex environments. The early planning and analysis stages have been recognized by many observers as critical, yet there have been few substantial advances in this arena. The primary focus of this work is the transformation of enterprise and systems goals into requirements, with the objective of formulating a method that is useful for identifying and refining the goals that systems must achieve and subsequently converting them into operational requirements. Specifically, the method detailed in this thesis provides procedural support for the early identification and elaboration of goals in the specification of software-based information systems. The objective of this chapter is to establish the context for this research.

1.1 Requirements Engineering

Requirements describe the desired functionality of a system. In general, there are two types of requirements: functional and nonfunctional. Functional requirements describe the

behavioral aspects of a system; non-functional requirements describe the non-behavioral aspects of a system.

Requirements analysis is the process by which the purpose and functionality of a system is elicited and modeled [53]. Requirements analysis and specification is one of the most challenging and error-prone activities in the software process. One reason for this is the high level of communication-intensive activity required. Stakeholders frequently lack a clear understanding of what the desired system should do and often tend to change their minds regarding the functionality that the system must exhibit. It is important to identify the issues that lead to instability early in the formulation of software requirements. To clearly identify these issues, stakeholders and analysts must communicate and negotiate the terms for any proposed system. Requirements are useful for such communication, serving as a contractual language understood by both parties.

Requirements engineering is plagued by practical problems such as complexity, volatility, ambiguity, the need to rely on intuition, and disagreements among stakeholders. The discussion of these practical issues sets the stage for the principles and methods presented in this thesis.

Complexity

Obtaining requirements is conceptually complex. It demands grasping a set of nebulous ideas that may be incomplete or in conflict, and transforming that set of ideas into a complete and consistent elaboration of the technical requirements for a software system. These technical requirements must be easily comprehensible to the intended customer or stake-

holders. Projects often fail to fulfill stakeholders' goals, however, because the requirements are inadequately explored and described. Analysts usually work from a vague statement of needs or goals. Additionally, each stakeholder has different requirements and priorities which sometimes conflict with the goals of the group as a whole. As a result, requirements are often not representative of the enterprise goals and/or conflict with them. Often the strategies for conflict identification and resolution are inadequate.

Volatility

Requirements creep is defined as the evolution in requirements between the time at which actual coding begins and the time of product delivery [54], with direct reference to changing and adding requirements. *Requirements churn* refers to the frequent modification of the same requirements or their priorities. The volatility of requirements, especially after the requirements "phase," is the source of one of the greatest problems in software development.

Capers Jones observes that "Creeping user requirements will grow at an average rate of 1% per month over the entire development schedule" [49]. For a two year project, this translates into an increase in functionality of 24% or 36% for a three year project. Requirements creep is a significant source of cost and time overruns. While systems are subject to a certain amount of requirements creep, such volatility can be minimized by the extensive exploration and identification of goals and requirements early in the requirements process.

Because requirements are volatile and constantly changing, there is much iteration in the refinement process due to the misunderstanding and/or misinterpretation of the requirements. Discussions transpire in which issues are highlighted, conflicts arise, and resolution is sought; when no record is kept of the analysis activities, omissions and the loss of information are inevitable. Better tools and process models are needed to support the negotiation, conflict resolution, and requirements refinement process [21].

Reliance on Intuition

Requirements elicitation techniques have received attention in recent years [23,28,35]. However, a definitive process for analyzing the gathered materials remains largely unaddressed and is left primarily to intuition. Requirements elicitation techniques have included sequential interviews of individual stakeholders or end-users, questionnaires of the general user group/community, and observations of end-users interacting with a particular system [35]. In large software development projects these techniques often prove inefficient, leading to poor communication among project members, thus necessitating a lengthy resolution process [16].

Ambiguities

When stakeholders write a requirements document, it is usually plagued with informational ambiguities, uncertainties, and gaps. In some cases, a requirements document is not always available, requiring analysts to extrapolate the needs and goals of the stakeholder

from only a vague statement of objectives. Often such documents lay the foundation for the requirements analysis process. Analysts and developers must carefully refine and formalize the information provided by the customer while iterating with the objective of producing a functional specification.

Requirements must be stated accurately before they can be implemented correctly. Consider the requirement: `The system should be quick`. Such a requirement provides no intuition into how "quick" the system should be or under what circumstances it should operate. This requirement is neither precise nor accurate. In contrast, the requirement: `The system should respond to a request for help in less than 10 seconds` exhibits much more precision. The expression of precise, clear requirements is essential in the prevention of requirements errors that arise due to ambiguities. Techniques are needed to guide analysts through the process of identifying specific goals and needs from ambiguous starting points.

Disagreements with Stakeholders

Individual analysts have traditionally been responsible for eliciting requirements from users, synthesizing the information acquired, and then modeling or developing a representation of the system requirements. The model developed is typically based on the analyst's personal understanding of the requirements after he has synthesized the information. The stakeholders' only involvement throughout the requirements determination process is limited to serving as an information source. However, when an analyst creates a model within the context of these parameters, the structure of the model is usually understandable only to

the analyst [70]. A model jointly developed by both the analyst and a group of stakeholders affords a model that is easily understood and readily accepted by both parties.

In group requirements determination, analysts, developers and stakeholders are represented and included in the process. Researchers have advocated increased user involvement in the systems development process, indicating that this heightened level of participation contributes to the development of better systems. According to Hayes, user participation enables the creation of a 'better' model than that created solely by the analyst and increases the probability of implementation success [43]. To this end, continued interaction with and the involvement of stakeholders throughout the requirements process is strongly advocated.

Given the above problems of complexity, volatility, and ambiguity, we now examine the consequences of poor requirements processes in the context of system development efforts.

1.2 Consequences of Poor Requirements

The consequences of misinterpreting requirements may be very severe. This section clarifies and substantiates the importance of clearly understanding the requirements early in the planning stages of the development process.

The ultimate consequences of requirements errors are expensive. In some large systems up to 95% of the code had to be rewritten to satisfy changed user requirements [9]. Boehm also reports that 12% of the errors discovered in a software system over a three year period were due to errors in the original system requirements. The correction cost can be up to $\frac{1}{3}$ of the total production cost [9]. If a software requirements error is detected and corrected during the analysis stage of the development process, the error is relatively simple

to correct since it involves only a correction to the requirements specification. However, if a requirements error is not corrected until the maintenance stage, a much larger inventory of artifacts is affected (e.g. specifications, code, as well as user and maintenance manuals). Additionally, late corrections involve a formal change control and approval process and require extensive re-validation. Fjeldstad and Hamlen estimate that 47% of the time devoted to maintenance activities is devoted to the actual enhancement and correction tasks while 62% is dedicated to comprehension activities [34]. These figures suggest that maintenance costs may be significantly reduced if efforts are devoted to increasing the level of comprehension within the system specifications. Thus, it is critical to ensure that requirements are identified as early as possible and that measures are taken to prevent any requirements from being overlooked. Because requirements engineering is expensive, the objective is to improve the process as quickly and efficiently as possible.

As the software lifecycle progresses, the cost of repairs due to errors made in specifying the requirements increases significantly [9]. Failure to identify errors in requirements can be dangerous; for example, 62% of safety-related functional faults in the Voyager spacecraft and 79% in Galileo were caused due to errors in recognizing the requirements [57]. It must also be noted that requirements errors are persistent. According to Boehm, $\frac{2}{3}$ of requirements errors are detected after delivery [9]. That is, the system is already delivered to the customer before the requirements error is detected by an analyst or the customer.

The consequences associated with poor identification of software requirements, discussed above, are largely due to lack of validation. The next section examines the origin of these requirements errors.

1.3 Requirements Validation

Requirements errors are often due to improper requirements validation. Requirements validation involves assuring that a system meets the actual needs of the stakeholders and customers. Validation is usually defined by asking variants of the following question: "Are we building the right system?" Validation is a subjective process in which assessments are made as to how well the proposed system addresses the stated needs of the stakeholders. It involves iterative reviews involving the participation of stakeholders and the developers. The process is necessary to ensure that requirements are consistent, complete, and realistic. Traditionally, validation includes activities such as walkthroughs and prototyping. Given these constructs, goal analysis is utilized as a key tool for validating software requirements and is presented in this thesis as an approach to validation.

Stakeholders frequently fail to use, or even discard, a system based on their acquired knowledge of the system as an ineffective means of assistance in the achievement of their goals. The cause of this misuse of effort and lack of efficiency is commonly attributed to the need to develop an understanding of the requirements from stakeholders who do not themselves understand the requirements. The problem with requirements engineering stems from stakeholders lack of understanding of their own requirements; this ambiguity is thus passed on to the analyst, resulting in imprecise objectivization of software requirements. Goal analysis improves the understandability of the requirements by enabling the production of a set of requirements that may be validated by those to whom the requirements bear the greatest significance. Since multiple stakeholders have many goals, it can be difficult to develop a clear understanding of the desired diversity. Goal analysis clarifies the stake-

holders' goals by tracking the rationale associated with specific goals. Thus, goal analysis becomes a means to validate system requirements.

1.4 Goals in Requirements Analysis

Traditional approaches to requirements analysis focus on the elicitation of specific requirements. As noted in Chapter 1.3 stakeholders usually have a better understanding of the general goals they want to achieve than they do the functionality that should be exhibited by the desired system. The requirements specification, based on formal models and formal specifications, often serves as a contract with the stakeholders. When stakeholders are unfamiliar with these notations, or have not been trained in formal specifications, these documents can be cumbersome and intimidating. Since requirements specification documents serve as a contractual language, it is important to provide stakeholders with information in an understandable language in which they may actively participate. By focusing on goals instead of specific requirements, analysts enable stakeholders to communicate using a language based on concepts (e.g. goals) with which they are both comfortable and familiar. It must also be noted that enterprise goals and system goals are more stable throughout the lifetime of an enterprise than are the requirements defined at any one time. It is therefore imperative to utilize an understanding and structure of goals to derive requirements from a stable starting point (e.g. goals).

The changes that occur throughout the progression of requirements model from an informal representation to a more formal representation may contribute to the difficulties in communication between the analyst and stakeholders. The transition from requirements

to design may be difficult for end users to follow due to the changes in representation. Functional modeling techniques do not provide the necessary 'evolutionary formalization' [5] needed to bridge the gap between the analysts' and stakeholders' understanding of the system requirements. Goals are evolutionary and may thus provide a common language for analysts and stakeholders.

This thesis builds upon existing work which addresses the utilization of goals in systems development for the direct transformation of goals into formal specifications. The objective is the direct transformation of goals in natural language requirements. The Goal-Based Requirements Analysis Method (GBRAM) presented in this thesis introduces techniques that force in-depth, methodical, and systematic analysis to clearly elucidate the desired goals.

1.5 Overview of Remaining Chapters

The research discussed in this thesis stems from the need for better heuristics and procedural guidance for initially identifying and constructing goals, with particular emphasis focused on those which involve the relationship between goals and scenarios. The need exists for prescriptive advice in the form of goal-identification heuristics and a set of consistently recurring questions to guide the process. Research conducted in the preparation of this thesis has identified a set of issues which practitioners should consider.

This thesis offers prescriptive guidance for goal identification, providing requirements elaboration techniques centered upon goals and scenarios.

Chapter 2 provides a survey of the related work in the field and discusses the use of goals in requirements engineering as well as the influence of goals in other disciplines.

Chapter 3 presents the case studies which served as the conceptual origin for the Goal-Based Requirements Analysis Method. The discussion in this chapter justifies the heuristics presented in Chapter 5.

Chapter 4 details the Goal-Based Requirements Analysis Method (GBRAM), focusing on the activities with which analysts are involved when employing the method. This chapter discusses the application of the method, illustrating its salient features through examples from the case studies discussed in Chapter 3.

Chapter 5 presents the guidelines and heuristics which guide analysts through goal-driven requirements analysis. Examples from the case studies are provided to elucidate the heuristics; the heuristics are accompanied by basic questions or inquiries for the analyst to apply when using the method. A discussion of the possible changes that result from asking these questions and guidelines for the kinds of refinements the analyst can make are also concomitant themes in this chapter.

Chapter 6 discusses the two main validation efforts for the method presented in this thesis: a large scale industrial case study involving the reengineering of an electronic commerce Web server, and an empirical investigation in which the method was applied to a small system by individuals who were previously not familiar with goal-based approaches.

Chapter 7 summarizes the contributions of the thesis and future work needed to further refine the method.

CHAPTER 2

Survey of Related Work

The world cannot be understood from a single point of view.

Eleanor Roosevelt

Although there has been relatively little attention paid to the process of acquiring goals for system development, the process of analyzing the objectives of a system has been addressed in depth. To position the work in this thesis, some of the more relevant previous work in this area is briefly surveyed.

In general, the existing techniques for acquiring and specifying system objectives are either formal or informal. Formal techniques address the specification of declarations and assertions in a logic-based formal language, which readily lend themselves to formal analysis. Specification languages describe in detail and with minimal ambiguity the behavior of mechanistic software systems. Since formal analysis affords the ability to engage in consistency checking [44], reachability analysis [17], and formal error checking, these techniques are receiving increased attention from industry in safety-critical domains; however, formal languages (e.g. Z [79], VDM [50], Larch [39]) are not well suited for capturing requirements models due to their limited scope. Practitioners who employ formal methods are concerned with mathematical correctness and precision [47]; formal languages lack constructs to clearly delineate the separation between domain descriptions and actual requirements [47], making

it difficult to consider adequately the relationship between the formal abstraction and the reality of the domain and environment of the proposed system. It has been observed that formal specification languages may not be useful for describing some systems (e.g. business processes which tend to not be mechanistic) [7]. The successful use of specification languages thus depends on the application of an analysis method. Informal techniques sometimes result in descriptions which are much more vague than those produced with formal methods.

Operational concept definitions (OCD) have been introduced as an alternative to specifications [7]. An OCD describes systems or processes in terms of scenarios, critical incidents, or examples of the problems an organization must solve. Whereas, specifications abstract away from the concrete, OCDs describe organizations through a series of concrete examples. These examples serve as a rich source of information system requirements and make the OCDs easier to understand for both stakeholders and analysts/developers; however, care must be taken to ensure that the scenarios actively represent the given organization's needs. Thus, it is imperative to relate OCDs to the goals of the organization. The study described in Chapter 3.1 discusses this relationship between goals and scenarios.

The work in this thesis attempts to provide more solid guidance for the initial identification and construction of requirements. Future extensions to this work can begin to offer the formal underpinnings needed to bridge the gap between formal and informal methods.

The general areas of work in the requirements engineering literature which are important to consider when discussing goal-based requirements analysis are: goal-driven approaches, inquiry-driven analysis, scenarios, viewpoints and negotiation. It should be noted that the analysis of goals is not unique to software; goals are also addressed in non-computing

intensive arenas such as goal-based learning and strategic planning. This chapter provides a summary of the related work in requirements engineering and discusses the influence of goals in other disciplines.

2.1 Analysis of Goals

A critical factor in successful projects is often that developers not only understand *what* they are developing, but *why* they are developing a given system [20, 63]. Goal-driven approaches focus on why systems are constructed, providing the motivation and rationale to justify software requirements. Requirements are often difficult for stakeholders to understand, but may be justified and explained to stakeholders through a discussion of the goals. These methods for requirements development have mainly concentrated on the notations for describing requirements rather than on where and how to obtain the information that is being described. The work in this thesis focuses on aspects other than specific representations, emphasizing analysts' tendencies to work with different sources of knowledge possessing various semantic properties.

The existing goal-based methods assume that the task of identifying goals, agents, and constraints is a straightforward process. In contrast, the Goal-Based Requirements Analysis Method (presented in Chapter 4) assumes that goals are not always explicitly stated *a priori* and that the process of identifying and abstracting goals requires guidance before it can be deemed 'straightforward.' The goal identification strategy is initially applied to informal descriptions of the desired system, which for example, may be taken from transcripts of the interactions between the elicitor and the stakeholder(s) or procured by other methods.

The goal refinement paradigm has been been discussed in the literature by Antón et. al. [4, 7], Dardenne et. al. [23, 24], Potts [60], Sutcliffe et. al. [81], van Lamsweerde et. al. [87], and Yu et. al. [17, 90]. Since a summary and comparison of several of these goal-driven approaches is available in [38], this chapter discusses two specific aspects of the goal-driven paradigm: the classification of goals and the refinement of goals.

2.1.1 Classification of Goals

Goals may be classified in several ways. Dardenne et. al. [24] offer a goal classification scheme which distinguishes three types of goals according to the conditions that specify the targets of the goals: achievement, maintenance, and avoidance. An achievement goal is satisfied when a target condition is attained. A maintenance goal is satisfied while its target condition remains true. An avoidance goal is satisfied while its target condition remains false. This classification scheme is well suited to operationalizing goals as actions that must be performed by the proposed system.

The model presented by Sutcliffe and Maiden [81] classifies goals according to six classes of desired system states: positive state, negative state, alternative state, exception-repair, feedback and mixed state. The six classes are expressed in terms of a policy-goal model comprised of three levels: policy level, functional goal level and domain goal level. Policy goals describe what should be done. Functional level goals provide information about what may be done to achieve policy level goals. Domain level goals are refined giving consideration to management implications, operational implications, and opportunities for automated support. This goal decomposition model [81], facilitates the structure of the problem

space by refining policies to identify the functions necessary for achievement. Extensions to this work [58,82] address the issues related to requirements completeness by identifying mismatches between goals and problem decompositions that can suggest incompleteness in a specification.

2.1.2 Dependencies Among Goals

Yu and Mylopolous' modeling framework is comprised of goals, rules, and methods to support the systematic analysis and design of business processes [90]. The framework consists of two main components: the Actor Dependency model and the Issue Argumentation model. The two models distinguish between process goals and design goals. The Actor Dependency model represents the organization as a network of interdependencies among actors; the Issue Argumentation model supports the reasoning that occurs throughout the design process, capturing the design argumentation and the merits of the various alternatives pertaining to issues of concern [90].

The method presented in this thesis relies on two types of goals: achievement and maintenance goals. In general, achievement goals are descriptive and map to functional requirements while maintenance goals are prescriptive and map to nonfunctional requirements. Since the objective of the Goal-Based Requirements Analysis Method (GBRAM) is to specify the requirements for a proposed systems, this classification using two goal types simplifies the process of operationalizing requirements.

2.1.3 Refinement of Goals

Several approaches to goal refinement are addressed in the literature. Dardenne and van Lamsweerde's goal refinement method, detailed in [24], offers a way of modeling concepts acquired during requirements elicitation. The requirements model, constructed during requirements acquisition, represents domain-specific instances of the conceptual meta-model components. Three levels comprise the meta-model: domain-level, meta-level, and instance-level. Domain-independent abstractions are defined in the meta level. Concepts specific to the application domain are defined in the domain level as instances of meta-level abstractions. Instances of domain-specific concepts are defined in the instance level. The three main component types in the meta-model are meta-concepts, meta-relations, and meta-attributes. Goals, constraints, agents, actions, etc., are considered meta-concepts. Meta-relations are defined as, for example, agent-performs-action, action-ensures-constraint, and constraint-operationalizes-goal. Pre-condition is an example of a meta-attribute. The interested reader should refer to [23] for a more detailed discussion. In summary, the goal acquisition strategy involves defining agents, goals, and constraints.

KAOS is a goal-driven elaboration method [23,24,87] which provides formal rules for deriving requirements based on theories of formal specification languages. It is explicitly directed towards creating specifications for composite systems, treating functional and non-functional requirements with the objective of addressing high-level domain goals as well as system specific goals. The goal refinement method of [24] was used in an informal study to specify the requirements for a software requirements validation environment [4]. The objective was to sketch the structure of a specification in terms of the goals to be auto-

mated and their operationalization into high-level actions to be performed by software, hardware and users. Using the goal refinement method of [24] to establish the requirements for a requirements validation system that uses the Inquiry Cycle [61,64] as the fundamental methodology, the main goals of the system were determined only with difficulty. The effort was not as straightforward or methodical as the literature suggests and the refining of high level goals, without a description of the system to be built, was not obviously systematic, requiring guess work and inventiveness. Informal studies suggest that KAOS is difficult to use and understand for persons not trained in formal methods and notations. KAOS, like other approaches, provides little support for formal reasoning regarding alternative assignments, an issue of utmost importance in the requirements engineering process. This becomes a legitimate concern when a decision is required to determine what should be automated and what should not. Chapter 3.1 discusses an approach to this problem which is adopted in GBRAM.

An evaluation of the meeting scheduler system observed that discovering implicit goals is a non-trivial task [87]. At times, it was necessary to depart from the strict application of the strategy, using scenarios to validate the goal structure and identify new goals and constraints. It is clear that the need exists for hybrid-strategies which combine goal-driven, scenario-driven and viewpoint-driven approaches. The work in this dissertation codifies a set of heuristics to guide this process of knowing when to break away from rigorous strategies and employ semi-formal techniques.

i^* is a methodical approach to designing business processes in the context of information systems development [89]. This approach offers a formal representation of goals and their behaviors with a formal decomposition structure, treating nonfunctional requirements.

The Goal-Based Requirements Analysis Method presented in this dissertation, is effective for treating functional requirements; Chung et. al. provide a thorough presentation of nonfunctional requirements in [17,88,90]. As previously mentioned, KAOS provides little support for reasoning about alternatives available to analysts. In contrast, Yu's strategic dependency model [90] supports the process of suggesting, exploring, and evaluating alternative solutions, providing the rationale for networks of actors in which agents depend on each other to achieve goals, perform tasks, and furnish resources. The model facilitates the identification of what is at stake for whom, and what impacts are likely if a dependency fails. This dissertation discusses four types of dependencies: goal, precedence, agent, and contract dependencies. The approach adopted in GBRAM differs from Yu's model [90] in that dependency relations are used primarily to order goals so that they may be subsequently refined (Refer to Chapter 4.2).

Current goal-based methods have not provided adequate strategies for the initial identification of goals. The objective of the research presented in this dissertation is to provide straightforward and methodical support for the analysis and refinement of goals for the specification of requirements in software-based information systems. The approach presented in Chapter 4 addresses the initial identification and structuring of requirements information.

2.2 Inquiry-Driven Analysis

Field studies [21,56] have emphasized the importance of improving communication with stakeholders in requirements analysis, achieving consensus among the stakeholders, and maintaining traceability [36,37,83] as requirements evolve. The Inquiry Cycle (IC) Model

addresses this need to support communication during the requirements process [61,64]. The model consists of a series of questions and answers designed to pinpoint where and when information needs arise. It provides a formal, yet dynamic, structure for describing discussions about requirements and for capturing the ongoing requirements elaboration process. As shown in Figure 2.1, the requirements are incrementally elaborated upon through expression, discussion, and commitment, contributing to the refinement of the requirements. Inquiry is supported so that participants (i.e., stakeholders and analysts) know what information is missing and which assumptions are pending. Since the IC is artifact-based, it enables stakeholders participating in the analysis process to share 'awareness' as they discuss artifacts that are visible and explicit.

The three artifacts of the Inquiry Cycle, Documentation, Discussion, and Evolution, are shown in Figure 2.1. Requirements documentation refers to the process of stakeholders writing down the proposed requirements. In requirements discussion, stakeholders challenge the proposed requirements with annotations. During requirements evolution, stakeholders attach change requests to the requirements on the basis of discussion, then refine requirements when the change requests are approved. The Goal-Based Requirements Analysis Method (GBRAM) incorporates all three IC artifacts in order to improve communication with stakeholders so that consensus may be reached at an early stage. The IC is flexible in that it allows different strategies to be employed to facilitate different styles of requirements analysis. The GBRAM instantiates the IC with goals expressed in natural language using goal hierarchies.

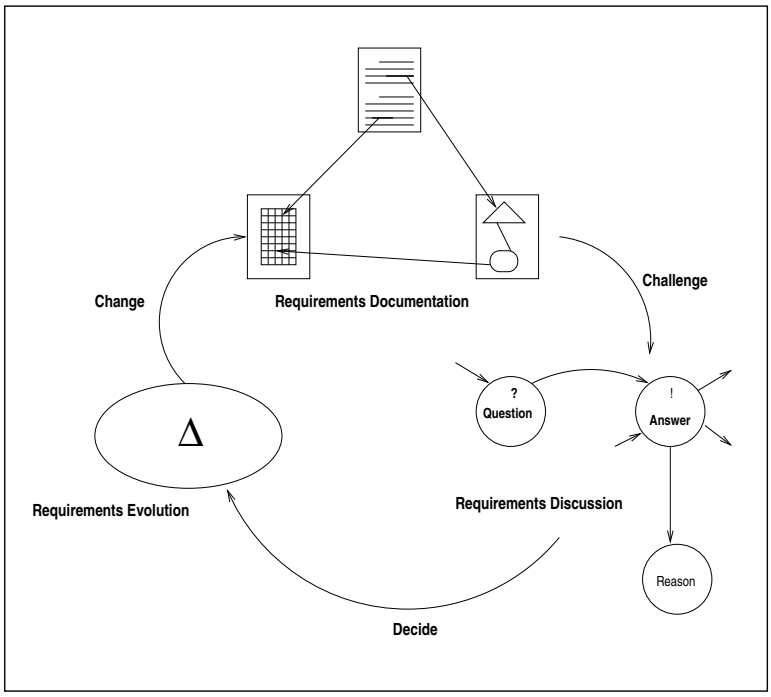

Figure 2.1. The Inquiry Cycle Model

The use of hypermedia technology in conjunction with inquiry-driven analysis has been advocated and is receiving attention within the requirements engineering community. Two tools have been developed to support this approach: Tuiqiao [85] and Ecolabor [86]. Tuiqiao is a hypertext tool for elaborating system requirements and Ecolabor is a networked hypermedia system for collaborative elaboration of system requirements. Several features of these tools are found in the GBRAM (for example, the ability to annotate requirements with questions, answers and alternatives).

2.3 Analysis of Scenarios

In order to adequately validate system requirements and specifications, stakeholders must be able to visualize the actual system in use and imagine how the system will support their work. In the requirements community, different authors understand scenarios to be different things, such as: a sequence of events [55, 73]; a general 'use case' [48]; one or more end-to-end transactions involving the system and its environment [64]; a description of behavior of the system and its environment arising in restricted situations [8]; or a trace of execution (such as in OMT [75]). In this dissertation, scenarios are understood to be behavioral descriptions of the system and its environment.

Scenarios are represented in a number of different ways. Jacobson represents scenarios as natural language prose descriptions and employs the use of 'sociograms' which represent the interactions between different system components during the execution of a system [48]. Rubin and Goldberg represent scenarios as 'scripts' in a linear tabular format with columns for agents and actions [73]. Potts et. al. [64] have augmented the scenario representation in [73], demonstrating how agents and actions are mapped onto the classes and services of an object-oriented design. Scenarios have also been shown to exhibit an episodic structure [64], indicating that scenarios may be constructed from scenario fragments (or 'episodes').

Scenario analysis entails the consideration of actions and behaviors arising from restricted situations in proposed systems, facilitating the identification of hidden requirements. The role of scenarios in the requirements process has received increased attention in recent years. The literature on the use of scenarios indicates that scenarios are useful for uncovering [7, 25, 56, 60, 64], elaborating [48, 52, 51, 55], refining [83], and validating

requirements [3, 8, 45]. Studies have shown that 'concrete' scenarios are essential for an understanding of the customer's needs and the operational concept of a system [56], suggesting that scenarios are an expressive representation scheme which allow stakeholders to clearly relate to their experiences, ultimately facilitating the validation of requirements.

The tendency among practitioners has been to consider only routine and "expert error-less performance" scenarios, disregarding frequent exception cases [14]. Different levels of abstraction may be more useful for the identification of exception cases. In [64], the simple enumeration of different kinds of cases (use cases), in a sense guaranteed that exceptional cases were not overlooked in the scenario analysis. The approach and techniques presented in this dissertation support the elaboration of goals using scenarios (Refer to Chapter 4.3).

2.4 Viewpoints and Negotiation

Traditional systems analysis focuses on *what* features (i.e. activities and entities) a system will support. Goal-based approaches focus on *why* systems are constructed, providing the motivation and rationale to justify software requirements. The notion of focusing on the *why* is not new;* organizing requirements around goals is new.

The problems associated with communication and agreement are not central to the work in this dissertation. However, it is important to note that these issues have been addressed by methods based on speech act theory. A summary of these methods is presented in [84].

*Context analysis, introduced in [70], determines the reasons *why* a system is to be created and the *purpose* of the system according to different viewpoints.

Much of the conflict identification and resolution literature centers around the concept of viewpoints. A viewpoint captures a particular responsibility performed by a participant at a particular stage in the development process. The ability to represent viewpoints and distinguish between them is especially useful during requirements analysis. The representation of viewpoints allows conflicts to surface early in the requirements process, rather than burying or overlooking them, so that they may be resolved during the process rather than ex-facto.

Easterbrook [28] argues that modeling multiple perspectives separately and then examining and explicitly resolving conflicts between them will result in a more precise and representative specification. A model of the requirements process which provides guidance for identifying and developing descriptions of the perspectives, and the resolution of conflicts between them is presented in [28]. A perspective is a consistent view of the world arising from the context of a particular role. Perspectives are represented using viewpoints, which are formatted descriptions in an appropriate representation scheme.

It has also been observed that it is possible to allow the resolution of inconsistencies to be delayed [29]. In delaying resolution, the relationships between partial specifications are explicitly recorded, representing both the resolved and unresolved inconsistencies. This record is then used to reason about the evolution of the requirements. The GBRAM relies on this notion of early identification and delayed resolution by capturing the rationale, assumptions, and questions that keep a requirement from being agreed upon by the stakeholders.

Finkelstein and Fuks [31] use viewpoints as a basis to study some of the communicational problems surrounding conflict. They propose a formal model in which specifications

are constructed from multiple viewpoints as a dialogue between two agents. The model allows agents to share, and thus detect inconsistencies in, their knowledge. Specifications are treated as dialogues in which the viewpoints negotiate, establish responsibilities, and cooperatively construct an overall specification. Agents can then query chains of reasoning made by other agents, and request information which they need to verify the conclusions. Conflicts based on misunderstandings and incomplete knowledge can thus be detected and resolved.

Finkelstein et. al. [32, 59] propose the use of viewpoints as both an organizing and a structuring principle in software development. In this analysis, viewpoints provide a complimentary means for acquiring goals, objects, and actions. The approach is especially useful for applications where the interdependency between human and automated agents is high. In the GBRAM, the role of viewpoints is most prominent when analyzing stakeholders priorities and agent responsibilities. It is important to note that a goal may be represented by multiple viewpoints. In light of this research, this dissertation presents goals as a higher-level abstraction which can incorporate the multiplicity of these viewpoints.

Leite and Freeman [53] propose viewpoint resolution as a way to provide early validation of requirements for complex systems. Their semi-automated technique is based on the premise that requirements should reflect multiple viewpoints and that requirements can be validated early in the process by studying the differences in these viewpoints. Using this technique facilitates the identification of missing and/or conflicting information. Leite and Freeman's work treats the resolution of viewpoint discrepancies as a static activity. The Static Analyzer [53] validates facts acquired during the process of requirements elicitation. The main focus is on very early fact-validation.

Robinson [65] describes a tool that integrates software specification components by evaluating preferences expressed from different viewpoints or perspectives. The tool guides the search for solutions which satisfy the different viewpoints to the greatest degree possible. Robinson observes that goals provide the "roots" at which conflicts should be resolved and multiple viewpoints should be reconciled. Robinson's work in negotiation seeks ideal solutions while balancing conflicting goals and/or beliefs [65–67]. The three main tenets of this work are: conflict detection, resolution search, and resolution generation. Conflict detection identifies and characterizes differences between perspectives into syntactic and semantic categories. Once a conflict is detected, a search for a solution that is acceptable to all perspectives is invoked. The characterization of the initial conflicts is used as a starting point and new alternatives are generated under the guidance of all available perspectives. Resolution generation methods may then be applied to generate new alternatives, given an alternative abstraction space and agent goal hierarchies. Methods include: compromise, substitution, compensation, and dissolution.

As evidenced by the discussion in this chapter, a rich collection of work exists in the research community on the representation of requirements, the elicitation process, the validation of requirements as well as goal-based approaches to the refining requirements. The next section discusses the influence of goals in other disciplines.

2.5 Cross Disciplinary Influences

Goals are not unique to software engineering. Goals and goal-based approaches are appropriate for processes other than simply specifying software requirements, and are central

to work in non-computing intensive and human-interaction fields. A prime example is strategic planning, the process of devising a course of action to achieve long term goals for an organization and positioning the organization to achieve those goals. This section provides an overview of these multidisciplinary influences.

2.5.1 Human-Computer Interaction

Goals have been addressed in-depth within the human-computer interaction community. The definition of a goal provided by Card, Moran, and Newell* asserts that a achievement of goals is attempted by doing those things the task itself requires to be done. The tasks should be analyzed in order to understand the course of human behavior. A framework for this analysis is provided by in the GOMS model [13], a model of human cognition typically used for understanding and measuring human-computer interaction. The GOMS model is comprised of four components:

- a set of Goals,
- a set of Operators,
- a set of Methods for achieving the goals, and
- a set of Selection rules for choosing among competing methods for goals.

The GOMS model provides a basis for describing how experts perform routine tasks, and is particularly useful for purposes of predicting performance (usually at the keystroke level).

*"A goal is a symbolic structure that defines a state of affairs to be achieved and determines a set of possible methods by which it may be accomplished."

The goal hierarchy representations used in requirements engineering [7,24] are not new. In GOMS, goals are organized into a hierarchy of goals and operators; goals are shown as square boxes and operators as round boxes with each goal ultimately terminating at a set of operators. A set of alternative methods by which the goal can be achieved and a set of selection rules for selecting among the methods are associated with each goal. The GBRAM employs a goal hierarchy representation for expressing topographies which may be mapped to Software Requirements Documents, as relayed in Chapter 4.

2.5.2 System and Process Reengineering

Software reengineering technologies begin the process of understanding existing systems, or parts of existing systems, at the abstraction specification level of the original system rather than at the strategic planning, requirements specification, and analysis levels [1]. While currently available reengineering tools and methods are useful for abstracting information from existing implementations, the tasks associated with analysis are not presently supported. This helps to perhaps explain why many systems seem to be reengineered with no consideration to redesigning the process for improvements. Reengineering efforts often result in the strict automation of existing outdated processes.

The process of reconciling existing processes with corporate goals is often viewed as a Herculean task by analysts charged with reengineering an enterprise. An in-depth analysis of existing processes and systems as well as analysis of corporate goals and missions is required. Often it is not clear how a particular process supports the corporate goals or mission, or why a particular process is even necessary.

The existing literature in this field is rich with claims of what makes a Business Process Reengineering (BPR) project successful. These keys to successful projects are typically based on attitudes within an organization, the need for a strong commitment from top management, and an openness to radical change. However, the need for clear systematic methods to employ in this process is typically overlooked or continues to go unaddressed. The ability to reengineer processes requires an in-depth requirements analysis of the objectives, processes, and procedures. Many of the same methods used for specifying software system requirements may be employed; however, the reengineering analysis process requires the ability to also consider business rules, goals, and policies. Several researchers have begun to address the issue of business rules as requirements that arise from the business objectives of an enterprise [69,90].

2.5.3 Goal-Based Learning

Goals exist in many settings. At the University of Chicago, researchers are employing goal-based scenarios and case-based reasoning in the design of a cardiac auscultation system [30]. Auscultation is the process of listening to the sounds in a person's body for diagnostic clues. The Cardiac Auscultation Diagnosis Instruction (CADI) environment teaches medical students diagnostic reasoning and the perceptual skills to support their reasoning. This system requires students to understand the complex anatomy and mechanics of the heart. The goal-based scenario approach used in the system has proven to be well suited for traditional medical education. Goal-based scenarios guide the framing of the learning task to motivate students, allowing educators to focus the learning task on the skills that

students must acquire. Case-based reasoning provides insights into what should be taught. This cognitive model helps students understand new experiences by referring to previous experiences, stored in the system's memory, and comparing the new cases to previous cases.

Goal-based scenarios for teaching cardiac auscultation is an example of the use of a goal-based approach. Goals are also prominent in product planning and strategic planning.

2.5.4 Product Planning

Product planning entails the development of an objectives document, which consists of three elements: goals, functions and objectives [72]. The goals may be philosophical in nature and usually express the overall statements of benefit sought. Examples of these goals include 'productivity increased,' 'costs reduced,' and 'revenues increased.' The document also outlines the functions needed to reach those goals; the objectives specify what is needed in order to enable the functions. The objectives document does not specify the 'how,' but rather 'what' the product must include. Similarly, a software requirements document specifies 'what' is needed, not 'how' it should be implemented. The product plan goals are analogous to non-functional requirements in a software requirements document. In strategic planning, goals are used to focus on the 'what' and the 'why.'

2.5.5 Strategic Planning

Evidence demonstrates that the involvement of stakeholders provides a broader picture and understanding during the strategic planning process. Ohio-based J.M. Smucker Co. recruited a team of 140 of their employees to devote nearly 50% of their time to a six month

strategic planning effort. By expanding the process to include 7% of their workforce, J.M. Smucker Co. developed a broader perspective, as evidenced by the generation of a dozen new initiatives which could double the company's $635 million revenues during the next five years [12]. J.M. Smucker Co. chose to validate their goals by including stakeholders in their strategic planning process. Similarly, in requirements analysis, it is important to involve stakeholders so that system goals may be validated early in the design process.

Whereas strategic planning was formerly limited to small groups of strategic 'thinkers,' the evolution of this paradigm in the 1990's closely resembles the evolution of the requirements engineering process [12]. Organizations are now adopting more inclusive strategic planning models. Instead of limiting the process to small groups, organizations are 'democratizing' the process by involving individuals from different disciplines and with differing levels of experience; some are even involving their stakeholders in the process. The goal of requirements engineering is to develop software which satisfies the needs of the stakeholder(s). The new 'democratic' approach to strategic planning shares this focus on the stakeholder desires, being centered on the notion that in order to produce what the stakeholder wants, the stakeholder must be involved in the process. Organizations have discovered that if they are not in tune with the needs of their stakeholders, the results of poor strategic planning efforts can be disastrous [40]. Likewise, when the requirements for an information system are 'wrong,' for whatever reason, the resulting software can fail to meet its ultimate objectives.

2.6 Summary

Clearly, the notion of goals and objectives spans many fields. The Goal-Based Requirements Analysis Method, presented in Chapter 4 is directed at the identification of enterprise and system goals to allow for their transformation into operational requirements. However, it is clear that the method may be generalized in such a way as to render it useful during other activities such as identifying learning objectives for a course or strategic objectives for an organization. The next chapter discusses three formative case studies which served as the origin of the ideas presented in this dissertation.

CHAPTER 3

Case Studies

The improvement of understanding is for two ends: first, our own increase of knowledge; secondly, to enable us to deliver that knowledge to others.

John Locke

There are three traditional research paradigms: mathematical, scientific, and engineering. In mathematics, research is derived from constructing concepts, often in the form of formal proofs and reflexive induction and reasoning. In ' social science and scientific fields, research assumes an experimental or empirical slant. The paradigm adopted in this dissertation is the engineering approach, which typically involves studying a problem, proposing solutions, and testing the solution on real problems. Specifically, the paradigm employed is one of conceptualization, empirical exploration, and testing. The Goal-Based Requirements Analysis Method, introduced in the following chapter, was developed and evaluated while working on real problems. This contrasts with other approaches to method development in software engineering research in which methods are developed and later tested on conceptualizations formed in isolation from real applications.

This chapter discusses development of the Goal-Based Requirements Analysis Method (GBRAM) in the context of its application to real case studies. The approach taken has concurrently led to the development of an integrative goal-based requirements method and

analyses of real problems. The initial case studies discussed in this chapter enabled the development of a systematic approach to goal identification and refinement, as discussed in Chapter 4; subsequent studies, discussed in Chapter 6, enabled evaluation and refinement of the method. Thus, the case studies presented in this chapter were formative, serving as the origin of the ideas and concepts presented in this thesis. The case studies discussed in Chapter 6 are summative; this distinction is key in that the summative cases previously developed methods were being validated, whereas the formative cases involved the evolution of the methods simultaneously coupled with validation. These case studies unfolded over time and GBRAM evolved as a result of its application to the case studies discussed in this chapter.

Each of these case studies involves a process or system:

- Financial Services Office (FSO) business process (Section 3.1);
- Career Track Training System (CTTS) (Section 3.2); and the
- Meeting Scheduler System (Section 3.3).

These projects are discussed in the following sections. Synopses of each project are followed by discussions of the methodology employed and the lessons learned through application of the method.

Table 3.1 summarizes the availability of the data for each of the case studies discussed in this chapter. Two of the case studies, the FSO and CTTS, are government confidential projects; thus, the raw data cannot be made available. However, the results from both of these studies have been published and are available in [7] and [4], respectively. The raw

data for the meeting scheduler case study is currently available via anonymous ftp*.

Table 3.1. Availability of Case Study Data

Case Study	Raw Data	Requirements Document
Financial Services Office	*	*
Career Track Training System	*	*
Meeting Scheduler	√	√
Key: √ Available from author upon request * Government Confidential		

3.1 Financial Services Office Case Study

The College Financial Services Office (FSO) case study focused on redesigning the FSO business process which is responsible for all of the College's finances. The FSO employs four full time employees, two part time student assistants, and requires the integral involvement of three College administrators.

This study involved the application of goal decomposition and scenario analysis in the context of Business Process Reengineering (BPR) [42, 41, 76, 78]. BPR attempts to avoid simply automating existing processes or tasks in organizations to introduce process efficiencies by questioning the reasons why specific processes and activities are linked together in

*The meeting scheduler data is available via anonymous ftp at ftp.cc.gatech.edu in /pub/groups/SERC/scenario.tar.gz.

support of a business entity [78]. Hammer and Champy observe that BPR requires "discontinuous thinking" so that dramatic performance improvements may be achieved [41]. The requirements for software systems that support reengineered processes must be understood in the context of the goals of the BPR project.

Due to the role of goals and objectives in business organizations and enterprises, the availability of financial processes in need of redesign and the necessary analysis of organizational goals made this project well suited for the development of GBRAM applicable ideas. The FSO case study exemplifies and validates the process of using scenarios in refining business process descriptions.

Methodology and Case Study Artifacts

The FSO case study was conducted for approximately 15 hours a week over a period of 3 months. Two College administrators and four FSO employees, a total of six stakeholders, participated in initial interviews for this case study. These interviews were tape recorded on site and later transcribed for future reference. The interviews served as information gathering sessions, providing an understanding of the organizational structure, the lines of communication within the organization, and the business processes for which the FSO is responsible. The initial elicitation of scenarios was unguided; each stakeholder was asked to explain the business processes for which they are responsible. Stakeholders expressed information in the form of scenarios or illustrations, often freely expressing the processes as scenarios which illustrate a process goal or demonstrate exceptional cases. The transcripts served as a source for goal and scenario identification.

The College administration initially provided a set of very high-level prescriptive goals emphasizing the deliverables for which the unit was responsible. However, upon interviewing the FSO employees, it was clear that employees in non-administrative positions outside the FSO were not aware of the existence of these prescriptive organizational goals and objectives; thus, these goals were not representative of the perceived goals of the stakeholders. The prescriptive goals in this study were analyzed using a top-down approach in order to develop a goal-hierarchy. The interview transcripts were analyzed using a bottom-up approach which also culminated in a goal hierarchy. The characteristics of the different identified goals in the bottom-up descriptive goal hierarchy and the top-down prescriptive goal hierarchy were then compared.

Lessons Learned

This section summarizes the lessons learned from the Financial Services Office case study and addresses the integration of these lessons into the Goal-Based Requirements Analysis Method.

Descriptive goals relate to operational activities

A pragmatic goal classification scheme, which differentiates between prescriptive and descriptive goals, emerged during this case study. *Prescriptive goals* are typically expressed by management-level stakeholders and account for organizational structures and processes that should be observed. Stakeholders are usually aware of goals in terms of operationalizations, expressed in the form of actions performed on a regular basis. *Descriptive goals* are

found in the current operational processes of an organization. The prescriptive goals in this case study were provided by management and were codified in the organization's written procedures, whereas the stakeholders responsible for carrying out the FSO processes tended to express goals in a descriptive fashion.

GBRAM recognizes that stakeholders possess different viewpoints and that their manner of expressing those viewpoints varies as well. Scenarios elicited from stakeholders support the descriptive goals and respective operations; however, it was difficult to elicit scenarios to describe activities which support the prescriptive goals. Chapters 4 and 5 explain how GBRAM provides guidance for analysts in identifying the goals as expressed by different stakeholders.

Descriptive goals are a better source of requirements

This is the only case study in this thesis in which a distinction is made between prescriptive and descriptive goals. The interest in descriptive goals for the FSO study stems from the need to analyze descriptive goals. This is salient in that descriptive goals are more indicative of what's going on, whereas management in this study was not cognizant of the real life operational strategies. Thus, descriptive goals were helpful during this case study as a source from which functional requirements could be derived.

Scenario analysis yields concrete process goals

The scenario fragments elicited during the stakeholder interviews were reviewed to identify a concrete set of process goals. Each scenario was analyzed by asking: *"What goal does this scenario fragment either support or satisfy?* and/or *"What goal does this*

scenario fragment prevent the achievement of?" A goal hierarchy was then constructed using the representation scheme shown in Figures 3.1 and 3.2. This bottom-up approach to goal identification focuses on stakeholder descriptions of activities and operations. It is less structured than other approaches since interview transcripts lack an organizational structure and are very much characterized by stream of consciousness; thus, transcripts do not offer a complete set of goals. Figure 3.1 shows the goal hierarchy constructed using a bottom-up approach.

These goals were defined primarily by the current operating procedures and did not offer a direct correspondence to the prescriptive goal set. However, the use of scenarios and inquiry was beneficial in that the analysis raised exceptional cases and goals which were not apparent in the prescriptive goal set.

The goals shown in Figure 3.1 were formulated upon examination of the scenarios elicited from the stakeholders and documented in the interview transcripts. For example, goal #2 in Figure 3.1 was extracted from the scenarios in NLD #4.2 on page 86. Every goal in Figure 3.1 was identified by examining the entire set of available scenario fragments; therefore, each goal has at least one supporting scenario.

The goal hierarchy in Figure 3.2 is based on the set of prescriptive goals, which were systematically decomposed into subgoals. The scenario transcripts were then reviewed in an effort to identify supporting and/or non-supporting scenario fragments such as those found in the bottom-up approach.

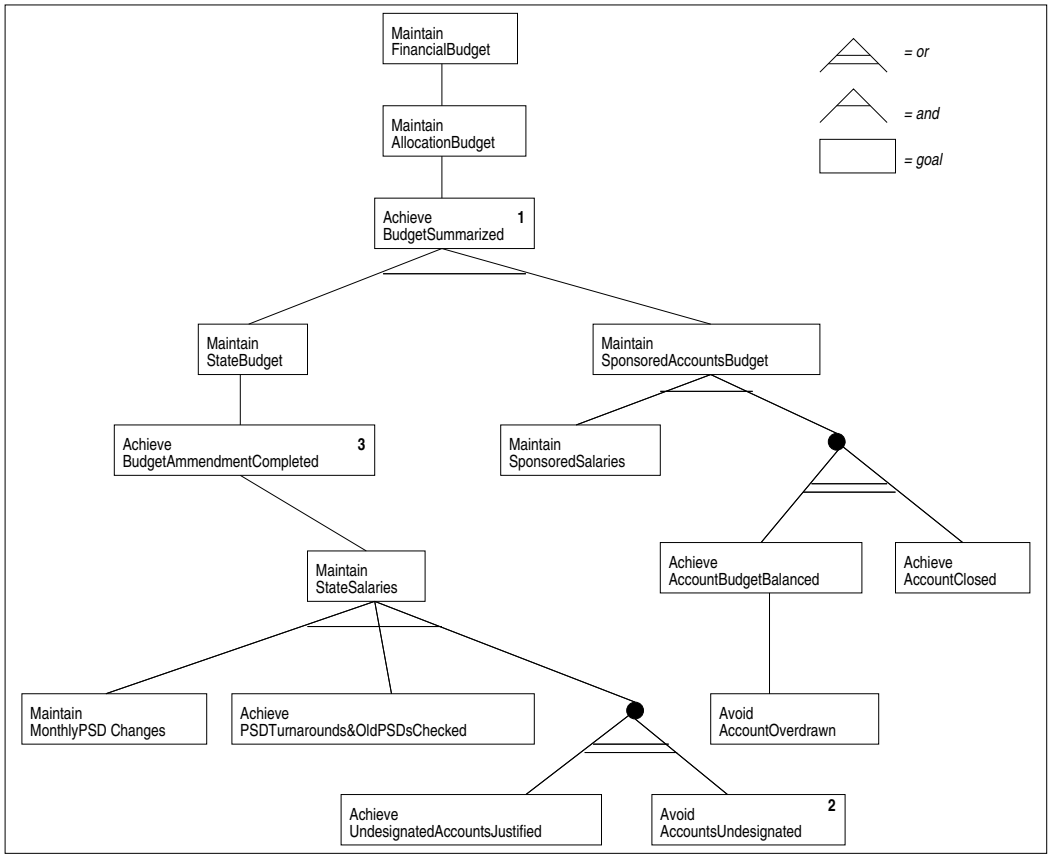

Figure 3.1. Goal Hierarchy for FSO Sponsored and State Accounts. This hierarchy was constructed using a bottom-up approach.

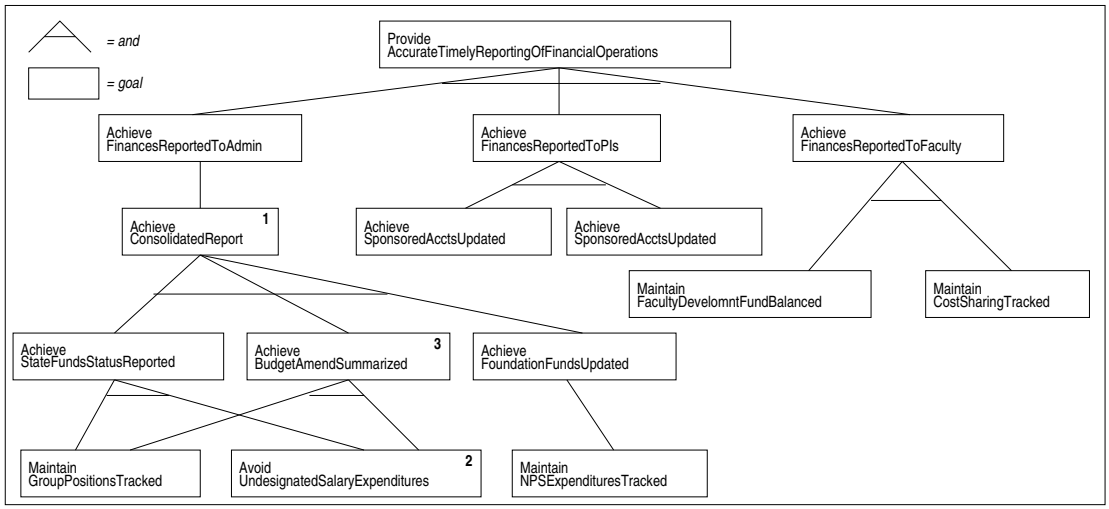

Figure 3.2. Goal Hierarchy for FSO Prescriptive Goal Set. This hierarchy was constructed using a top-down approach.

Scenarios were identified for three of the prescriptive goals: 1) `Achieve Budget Summarized`, 2) `Avoid Accounts Undesignated`, and 3) `Achieve Budget Amendment Completed`. These goals correspond to goals 1, 2, and 3 in Figure 3.1. The top-down approach requires an initial set of prescriptive goals. In the FSO, these goals displayed a definite emphasis on deliverables. Only four goals in Figure 3.2 had supporting scenarios.

Taking a bottom-up and top-down approach yields multiple viewpoints

The FSO goals were first identified by analyzing a list of process inefficiencies identified during the initial bottom-up analysis of the FSO business processes. Each process inefficiency was examined by asking *"What goal is prevented from being satisfied by this inefficiency?"* While this approach is structured and facilitated the identification of scenarios that satisfy system goals, it did not provide a complete set of goals; furthermore, the

goals identified in this manner did not offer a direct correspondence to the set of prescriptive goals. Thus, the results of this analysis were not very useful for the consideration of information system requirements.

Employing only a bottom-up approach in goal identification fails to provide a sufficiently high-level view for reorganizing/restructuring the organization. Similarly, sole employment of a top-down approach prevents analysts from developing an understanding of the actual current processes in an organization. However, a top-down approach coupled with a bottom-up approach offers a more complete view of an organization and its processes. These views complement each other, resulting in the identification of multiple viewpoints which may then be resolved.

Synonym identification facilitates viewpoint resolution

Contrasting both of the goal hierarchies gives rise to issues pertaining to synonymous goals and their identification. Consider what the FSO employees refer to as the "Allocation Budget" in Figure 3.1. The College administration refers to the allocation budget as the "monthly consolidated report" as shown in Figure 3.2. Clearly, this is indicative of conflicting viewpoints. If this difference in nomenclature were resolved, it would probably be possible to identify a few more supporting scenarios for the prescriptive goals in Figure 3.2. Some of the scenarios which support the goal `Achieve Monthly Report Consolidated` in Figure 3.1 would also support the goal `Maintain Allocation Budget` in Figure 3.2.

This case study was initially motivated by the study of scenario analysis for the meeting scheduler [64]. The goal and scenario analysis of the FSO business process exemplifies the desirability of combining both a top-down and bottom-up view. The goals from the bottom-up analysis were much more concrete, process/task oriented, and descriptive in nature, while the goals from the top-down approach were prescriptive in nature. This difference led the analysis to focus on understanding the relationships between the two sets of objectives. Since scenarios played a role in refining the FSO goals, subsequent case studies sought to understand the relationship between goal refinement and scenarios.

The following section discusses an analysis of an Air Force base training acquisition process.

3.2 Career Track Training System

The Career Track Training System (CTTS) case study focused on an Air Force Base (AFB) responsible for sending employees to Air Force and Department of Defense (DoD) training. The training acquisition process was observed to be extremely fragmented, necessitating an enormous amount of time and effort involving dozens of people across numerous organizational boundaries. Approximately 155 AFB employees are enrolled in official training each year. The Goal-Based Requirements Analysis Method was employed to analyze the goals for a Career Track Training System (CTTS).

The CTTS business process was appealing and well suited for investigation of the GBRAM for several reasons. First, the requirements for the continuing education process and desired system illustrate requirements problems typically exhibited by information systems. For example, the need to register persons for training sessions in the military is analogous to the need to register students for university classes in a course registration system. Second, the problem can also be considered a resource management (and scheduling) problem since it concerns the provision of courses for employees. As such, it addresses issues which are relevant to many systems and is, thus, widely applicable. Third, due to the involvement of different units in the AFB, and the DoD, the likelihood of conflicting goals among those units is high, making it an appropriate case to consider since one of the GBRAM goal refinement approaches employs conflict identification strategies. The following section explains how the case study was performed and provides an overview of the project artifacts.

Methodology and Case Study Artifacts

Two analysts, the author of this thesis and an analyst from the AFB, conducted the CTTS case study for approximately 10 hours a week over a period of four months. One member of the team was familiar with the application domain and was also a stakeholder in the system; the other possessed a knowledge of analysis methodologies and was not a stakeholder in the system. It was observed that having both areas of expertise was essential for performing an effective analysis. Given a one page textual description of the current training acquisition process, the analyst and stakeholder conducted an analysis of

the process in order to identify the system goals. The information from the analysis was reviewed, system goals were identified using inquiry and action word identification, and a goal structure was constructed using the representation scheme presented in [8]. In-depth interviews were conducted with AFB personnel and professionals in the training acquisition process to develop a comprehensive understanding of the current process. This interaction resulted in the construction of a detailed, informal flow chart model of the current process, from which an artifact of the redesigned/reengineered model with a statement of the system goals and requirements was derived. The following subsection discusses the lessons learned during this case study.

Lessons Learned

As with the previous case, the lessons learned from this case study served as an incubator for the ideas that influenced the evolution of the method. This section discusses the lessons learned during the CTTS case study and is a high-level explanation of how these lessons were integral to the formation of the method. Extensive examples from this case study are presented in the exposition of the method in Chapter 4; thus, to avoid repetition, the discussion in this section is brief.

Stakeholders express activities more readily than they express goals

Stakeholder identification allows the consideration of activities in which agents stand to gain or lose, thereby facilitating the identification of conflicting viewpoints. However, in the CTTS, stakeholders rarely gave consideration to the actual goals of a system without

some form of prompting; instead, they were more likely to focus directly on operationalizations and actions which they rely on the system to perform. Since stakeholders did not volunteer goals, goals had to be elicited through prompting. This illustrates the importance of allowing stakeholders to state goals descriptively so that the analyst may synthesize these viewpoints and incorporate them into a representative set of goals. GBRAM allows analysts to share knowledge about the different stakeholders, their viewpoints, and their responsibilities so that inconsistencies may be detected. The role of stakeholder viewpoints is elaborated upon in Chapters 4 and 5.

Multiple sources yield better goals

The goals for a desired system are not always clear at the outset of analysis and must be extracted from diverse representations of information (e.g. stakeholders are not considered to be "representations" in the context of this discussion). It is unlikely that analysts will produce a complete goal set for a system given only one information source, but the combination of goals extracted from various information sources does produce a *more* complete set of goals. This is especially probable when analysis of both the current and desired systems is incorporated into the goal set.

During this case study a direct correlation was observed between the different types of goals identified and the nature of the available documentation. Three sources of information were available for this analysis: a textual introductory statement, a textual process scenario, and flow charts of the existing process. Analysis of these three sources yielded the identification of 36 goals. The introductory statement and process scenario comprise a one-page textual description detailing the high-level mandates that drive the system and

organization, and the employee certification acquisition process. The flow charts represent the current process, described during interviews with stakeholders. It is important to note that the medium, or language, in which the information is expressed is less important than the actual information provided. The introductory statement for this case study is the most declarative of the three sources, while the process scenario exhibits commonalities with flow charts in its step-by-step description of the process.

The heuristics for identifying goals in GBRAM depend on the various sources with which an analyst works. Process descriptions in flow charts tend to be much more clear and succinct than are transcripts of someone speaking, since transcripts lack an organizational structure which minimizes tangential comments. These observations all played a formative role in the development of the heuristics presented in Chapter 5. Consequently, GBRAM provides guidance for analysts to identify and extract goals from information sources such as process diagrams and interview transcripts.

Categorizing goals suggests operationalizations

It is useful to differentiate among types of goals by noting the target conditions of the goals. In GBRAM, goals are classified as either achievement or maintenance goals. An *achievement goal* is satisfied when the target condition is attained. A *maintenance goal* is satisfied as long its target condition remains true*.

*The classification of achievement and maintenance goals is discussed in greater detail in Chapter 4.2.

While both achievement and maintenance goals were identified in the CTTS, maintenance goals are more likely to appear in organizational and policy level descriptions than are achievement goals since maintenance goals tend to delineate the objectives of an organization. Consider, for example, that nine maintenance goals were extracted from the CTTS prescriptive description, while none were extracted from the process description or identified from the current process descriptions. This juxtaposition supports the findings reported in [7] that organizational goals are often not reflected in operational strategies. Achievement goals are more likely to arise when exploring the process descriptions, which tend to contain more action words than prescriptive descriptions. Thus, GBRAM uses action word identification as a technique for identifying goals; the role of action words in goal identification is discussed in Chapters 4.2 and 5.2. In the CTTS study achievement goals were more readily extracted from flow charts than from, for example, the interview transcripts of the Financial Service Office study (See Section 3.1).

Given that high-level maintenance goals are not often reflected in operational strategies, it is useful to differentiate them from achievement goals. An efficient enterprise and use of information technology/systems depends on a close correspondence between the supporting systems and the enterprise goals and objectives; this thesis subdivides enterprise goals and objectives into maintenance goals, representing organizational goals, and achievement goals, representing operational strategies.

As hypothesized prior to this study, achievement goals are best mapped to actions that occur within the system, while maintenance goals tend to be nonfunctional (e.g. constraints that prevent things from occurring). Maintenance goals are helpful when operationalizing achievement goals because they can point to previously overlooked goals. In the CTTS,

analysis of new relevant information providing additional descriptions of maintenance goals resulted in refinements to the set of achievement goals, thus yielding more meaningful operationalizations. By categorizing goals, analysts may also begin bridging the communication gap between stakeholders and developers. The integration of these concepts and experiences is evidenced through the discussion in Chapters 4 and 5.

Diversity of goal information gives a rich picture

System requirements information tends to be much more specific and more implementation-dependent than domain theory information. Application domain information may appear more useful for identifying the high-level goals, since the main concern is the overall problem and not a proposed solution. However, more concrete goals and goal obstacles* can be identified from system requirements, which primarily focus on actual functionality and system performance. This identification ultimately leads to a better understanding of the system due to the availability of more concrete descriptions which allow analysts to more clearly envisage the desired system. Consider, for example, the different types of goals identified in the CTTS. In the CTTS, maintenance goals are extracted from policy level descriptions and achievement goals are extracted from both policy level and process descriptions; thus, when operationalizing† an achievement goal, it is helpful to analyze the goals extracted from different sources. This analysis enables analysts to clarify their understanding of the goal they seek to operationalize so that it may be specified in more detail.

---

* *Goal obstacles* prevent or block the achievement of a given goal.

† *Operationalization* refers to the process of translating a goal into an operational requirement.

Constraints indicate requirements and point to new goals

Constraints provide additional information about requirements that must be met for a given goal to be completed, providing insight into issues that must be considered when goal priorities change. For example, a constraint may indicate *when* a goal can be completed. Consider a constraint which specifies that a meeting must be scheduled on a specific day. If a room is not available or no one can attend the meeting on that day, the goal priorities must be reexamined. By examining the constraints in the CTTS, new goals were identified that would otherwise have been overlooked. GBRAM provides heuristics which aid analysts in identifying and uncovering hidden goals and requirements via constraint analysis. This process is further elaborated in Sections 4.3 and 5.2.

Exceptions can be explained via goal obstacles

Goal obstacles are an effective mechanism for the anticipation of exception cases that must be handled by system operations. Some requirements arise from analysis of obstacles and are thus not obvious to stakeholders. In the CTTS, obstacle analysis forced the consideration of reasons that could prevent an agent from achieving a goal. In Chapter 4, Examples 4.14 and 4.15 on page 98 illustrate how GBRAM aids in obstacle analysis by indicating the need for the proposed system to be prepared to handle possible obstacles and goal failures. Another benefit of obstacle analysis is the ability to more easily identify scenarios* to determine why a goal could be blocked and when the relationship between goals and scenarios deserves further research. This is useful in that obstacles indicate which scenarios, if elaborated, would ensure coverage of exception cases. This relationship be-

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

tween goal obstacles and scenarios is explained in Chapter 4 and is detailed in Chapter 5.

Scenarios play a major role in uncovering issues

Scenarios offer a natural and concrete way to describe the circumstances in which a goal may fail or be blocked, facilitating the discovery of new goals and the consideration of alternative mappings from goals to operations. In the CTTS, the use of scenarios led to the uncovering of hidden goals and obstacles as well as the identification of circumstances leading to the occurrence of goal obstacles. By considering the CTTS obstacle `Submitted paperwork not reviewed`, issues were uncovered that may have otherwise been overlooked. For example, if paperwork is not complete, the employee must be notified and asked to resubmit their paperwork or risk losing their ability to improve their certification status. It was clear during the CTTS that the identification of pre- and post-conditions for each goal is important to allow each goal to be operationalized into the requirements, as discussed in Chapter 4 on page 88. Analyzing scenarios in the CTTS enabled the identification of possible postconditions for various behaviors and goals. Experiences and observations from both the meeting scheduler and the CTTS demonstrate that scenarios are useful for uncovering and elaborating requirements, checking for completeness and conflicts, and communicating with stakeholders. These concepts are addressed and supported by GBRAM, as explained in Chapter 4.

Goal Evolution

Goal evolution concerns the refinement of goals from the moment they are first identified to the moment they are translated into operational requirements for the system specification. Evolution of the CTTS goal set was marked by modifications and additions to the goals themselves. In order to examine the evolution of the CTTS goal set, the size of the goal set was tracked during each stage of the analysis. The results of the goal analysis are seen in Figure 3.3. The ovals show the number of identified instances of achievement and maintenance goals, constraints, goal obstacles, and scenarios. The alpha-numeric combinations represent the evolution of the initial goals into a more refined and elaborated set of goals.

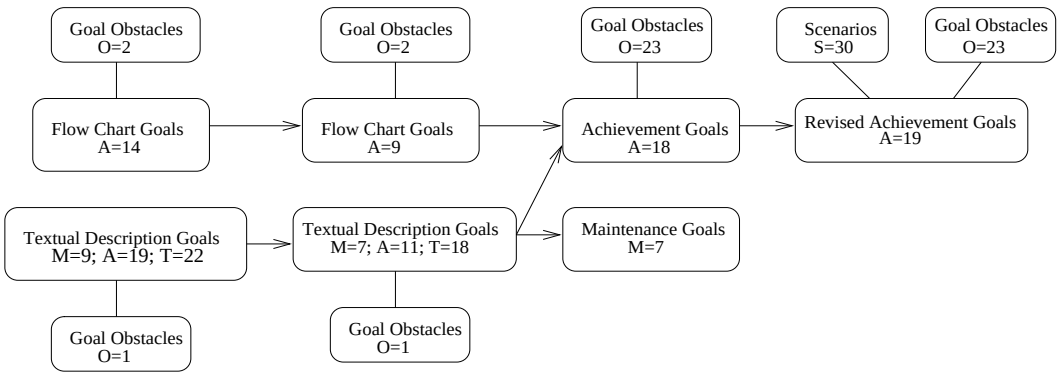

Figure 3.3. Evolution of CTTS Goal Set

Given 36 lines of text (470 words) in the CTTS textual description (Introduction and Process Scenario), 22 goals were identified. During the case study, three approaches for

reducing the size of the goal set were uncovered: eliminating duplicate goals, refining goals based on system entities, and consolidating nearly synonymous goals. Using these refinement techniques, discussed in Chapter 3, the size of the goal set, derived from the textual description, was reduced from 22 to 18 goals. The flow chart goals were refined and reduced from 14 to 9 goals. Once the duplicate goals were eliminated and the synonymous goals reconciled, the goal obstacle and scenario analysis elaboration techniques were employed, yielding the identification of 23 obstacles and 30 scenarios for the achievement goals. Finally, as a result of refinements during operationalization, the size of the achievement goal set (18 goals) increased in size from 18 to 19 goals. This analysis of the evolution of the goal set facilitated early reasoning regarding the scope of effect using the refinement and elaboration techniques. More detailed discussion of these refinement and elaboration techniques is found in Chapters 4 and 5.

3.3 The Meeting Scheduler

An initial meeting scheduler case study [64] was conducted in an effort to validate the inquiry cycle model [61]; this analysis predates both the FSO and the CTTS. The meeting scheduler is a theoretical problem which may be applied to real development projects. The system resulting from the application of this problem provides a rich combination of challenging features (e.g. interfering goals, real-time aspects, and multi-agent cooperation and communication). Since the starting point for this analysis effort was a short requirements document that had to be understood, clarified, and refined, the meeting scheduler system also illustrates contractual requirements. Additionally, the meeting scheduler illustrates

market-driven project issues, providing a great deal of latitude to make decisions regarding which features or components developers should implement. This study entailed an investigation of the types of questions that analysts ask about a set of written requirements and how they tend to be answered as well as the role scenarios play in the process.

The Goal-Based Requirements Analysis Method was not complete after the FSO and CTTS case studies leaving questions to be answered. At this juncture, we revisited the meeting scheduler from a different angle for observation of goal identification and refinement.

Methodology and Case Study Artifacts

The goal-based method was applied during a reevaluation of the meeting scheduler problem. Two sources of information were available for this analysis: the existing two-and-half page "Preliminary definition at the meeting scheduler system" written by Axel van Lamsweerde and his students at the Catholic University at Louvain [87], and transcripts of interviews* with three administrative assistants responsible for scheduling departmental meetings and maintaining the calendars of several professors in an academic college. Efforts were focused on two (of three) specific sections in the preliminary definition: Meeting Scheduling Domain Theory and System Requirements. Due to the preliminary nature of this study, we did not focus on the information in the third section, which pertains to system extensions. The domain theory, system requirements, and interview transcripts served as information sources for goal and scenario identification.

*These interviews were conducted by Idris Hsi, a graduate student at Georgia Tech.

Lessons Learned

This section summarizes the lessons learned from the Meeting Scheduler case study and addresses the integration of these lessons into the Goal-Based Requirements Analysis Method.

Goals may be identified using an inquiry-driven approach

The initial meeting scheduler case study included an investigation of the types of questions that analysts ask about a set of requirements [64]. In the meeting scheduler revisited, the focus was on instantiating this inquiry model for a goal-based approach. During this case study, it was evident that an inquiry-driven approach aids in goal identification.

The domain theory description explains how meetings are typically scheduled. It contains 22 lines of text which comprise the domain theory portion of the preliminary definition. Goal identification yielded 12 goals which were identified by asking *"What goal(s) does this fragment exemplify?"* The identification of these goals was relatively straightforward. Consider the following description:

> Domain Theory Description: *Meetings are typically arranged when the meeting initiator asks all potential meeting attendees for the following information based on their personal agenda:*
>
> - *a set of dates on which they cannot attend the meeting (hereafter referred to as "exclusion set")*
> - *a set of dates on which they would prefer the meeting to take place (hereafter referred to as "preference set")*

The goals G_1, G_2, G_3, and G_4, shown in Table 3.2, were identified from the Domain Theory Description. It should be noted that GBRAM does not require that goals be ex-

pressed in a specific format; throughout this thesis, goals will be displayed in a tabular format, as in this table, or in a hierarchical format, as shown in Figures 3.1 and 3.2.

Table 3.2. Goals Identified from "Scheduling Meetings: Domain Theory"

Goals	Agent
G_1: Meeting requested	Initiator
G_2: Potential attendees specified	Initiator
G_3: Exclusion set requested	Initiator
G_4: Preference set requested	Initiator
G_5: Date range specified	Initiator
G_6: Equipment requirements requested	Initiator
G_7: Location preference requested	Initiator
G_8: Exclusion set provided	Attendees
G_9: Preference set provided	Attendees
G_{10}: Equipment requirements provided	Active participant
G_{11}: Location preference provided	Important participant
G_{12}: Meeting arranged	Scheduler

Multiple sources yield a more complete set of goals

In the FSO, it was observed that stakeholders tend to describe goals in a descriptive fashion. As observed in the CTTS system, the goals that are extracted from process descriptions when used as the exclusive source of information are insufficient for achieving throughness and completeness. Therefore, the 'meeting scheduler revisited' example relies on additional transcripts of interviews conducted with three persons responsible for coordinating meetings in a university department. Table 3.3 shows the goals identified from one of the three interview transcripts. Several of the goals (e.g. G_2: `Decentralized participants coordinated` and G_7: `Overhead time reduced`) suggest areas of personal concern to individual stakeholders. In a BPR effort, such goals suggest process inefficiencies

which are candidates for redesign in the new system.

Table 3.3. Goals Identified from Interview #1 (JM)

Goals	**Agent**
G_1: Meeting scheduled	Scheduler
G_2: Decentralized participants coordinated	Scheduler
G_3: Potential attendees specified	Initiator
G_4: Participant schedules requested	Initiator
G_5: Participant schedules received	Participants
G_6: Meeting requested	Initiator, Scheduler
G_7: Overhead time reduced	Scheduler
G_8: Date range specified	Scheduler
G_9: Preference set requested	Scheduler
G_{10}: Preference set provided	Attendees
G_{11}: Preference sets compiled and formatted	Scheduler
G_{12}: All preference sets consolidated	Scheduler
G_{13}: Important attendees preference sets requested	Scheduler

Goal obstacles & conflicts may be identified by using an inquiry-driven approach

By asking *"What goal(s) does this fragment obstruct or block?"* it was possible to identify goal obstacles. Table 3.4, shows that two conflicts were identified which may potentially block goal G_{12} (`Meeting Arranged`).

Table 3.4. Goal Obstacles for G_{12} from "Scheduling Meetings: Domain Theory"

Goals	Agent	Goal Obstacles
G_{12}: Meeting arranged	Scheduler	1. Date conflict 2. Room conflict

Both of these conflicts (`Date conflict` and `Room conflict`) were identified by analyzing the following description:

> Date range and conflict description: *The proposed meeting date should belong to the stated date range and to none of the exclusion sets; furthermore, it should ideally belong to as many preference sets as possible. A "date conflict" occurs when no such date can be found. A conflict is strong when no date can be found within the date range and outside all exclusion sets, but no date can be found at the intersection of all preference sets.*

For each goal obstacle (conflict) identified for the meeting scheduler, an additional table was constructed. The date conflict and room conflict tables are shown in Tables 3.5 and 3.6. Each table describes the conflict and enumerates the possible resolution strategies. In the Date conflict case (Table 3.5), four resolution strategies were specifically abstracted from the text.

Table 3.5. Date Conflict Description and Resolutions

Meeting Arranged: Date conflict
Strong Conflict: no date can be found within the date range & outside exclusion sets
Weak Conflict: dates found within the date range & outside all exclusion sets; but at intersection of all preference sets
Resolutions:
R_1: Initiator extends date range
R_2: Some participants remove some dates from their exclusion sets
R_3: Some participants withdraw from the meeting
R_4: Some participants add some new dates to their preference set

In the room conflict case (Table 3.6) two possible resolution strategies were extracted from the domain theory. The identification of such conflicts suggest candidate scenarios for analysts to elaborate. For example, consider R_2 in Table 3.5; if some participants remove

dates from their exclusion set after the meeting is scheduled, this may require the meeting to be rescheduled. The inquiry approach thus aids in identifying exceptional cases which the system must be prepared to handle. Chapters 4 and 5 discuss how GBRAM supports this process and provide a catalog of questions to guide analysts through this inquiry process.

Table 3.6. Room Conflict Description and Resolutions

Meeting Arranged: Room conflict
Strong Conflict: No room available at meeting date
Strong Conflict: No room available with proper equipment meeting date
Weak Conflict: No room available at intersection of Location preference sets
Resolutions:
R_1: New round of negotiation
R_2: Active participant modifies Equipment requirements

Concepts expressed in concrete terms are easily understood by stakeholders

The most significant difference observed between the domain theory and the system requirements was the number of goal obstacles. Two potential goal conflicts (shown in Table 3.4) were identified from the domain theory, whereas sixteen goal conflicts were identified from the system requirements (shown in Table 3.7). The identification of 16 concrete goal obstacles, or conflicts, supports the theory that concepts expressed in concrete terms are more understandable to stakeholders and analysts. Although more goals and goal obstacles were identified from the system requirements, it was observed that at times it was more difficult to abstract functional goals from the system requirements by considering each statement and simply asking *"What goal(s) does this statement exemplify?"* Many of

Table 3.7. Goals Identified from "System Requirements"

Goals	Agent	Goal Obstacles
G_1: Meeting organization supported	Scheduler	
G_2: Meeting date determined	Scheduler	
G_3: Meeting location determined	Scheduler	
G_4: Meeting arranged	Scheduler	1. Participant modifies preference set(s) 2. More-important meeting called
G_5: Potential participants notified	Scheduler	
G_6: Overhead time reduced	Scheduler	
G_7: Typical meeting management reflected	Scheduler	
G_8: Conflict resolution supported according to client's resolution policies	Scheduler	
G_9: Interactions among participants supported	Scheduler	1. Participants ignore requests 2. Notification not delivered
G_{10}: Interactions among participants minimized	Scheduler	
G_{11}: Multiple meeting requests satisfied in parallel/concurrently	Scheduler	1. Meeting requests compete for overlapping time 2. Meeting requests compete for space
G_{12}: Decentralized requests satisfied	Scheduler	
G_{13}: Physical constraints maintained	Scheduler	1. Meeting room allocated to more than one meeting 2. Participant scheduled for two meetings at same time
G_{14}: Drop-dead date kept short	Initiator	
G_{15}: Participants notified soon after meeting arranged	Scheduler	1. Notification not delivered 2. Notification ignored
G_{16}: Lower bound between drop-dead date & date range specified	Initiator	
G_{17}: Privacy maintained	Scheduler	1. Non-privileged participant becomes aware of another's constraints
G_{18}: Usable by non-experts	Scheduler	
G_{19}: Customizable	Scheduler	
G_{20}: Evolving data accommodated	Scheduler	1. Participant failed to send notification of new address
G_{21}: Explicit priorities among dates in preference set satisfied	Scheduler	1. Priorities not specified
G_{22}: Participants represented by substitute	Participant	1. Delegate unavailable
G_{23}: Meeting attended in full	Participant	1. Participant arrived late 2. Participant left early

the statements exemplified goal obstacles; from those obstacles it was possible to infer the actual goals.

Conditions imposed on goals suggest constraints

Some of the statements neither exemplified a goal or obstructed a goal. Instead, they exemplified constraints which must be met in order for a goal to be achieved. For example, several constraints were identified from the 'Date Range & Conflict Description', shown on page 60 (e.g. C_3, C_4, C_5, & C_6 shown in Table 3.8). These statements are classified as constraints and not as goals. Ideally, goals are high level objectives such as 'Meeting Scheduled.' The textual fragments of the description impose constraints on the objective of scheduling a meeting. The description delineates certain requirements and conditions that must be met. It is interesting to note, however, that G_5 (`Date range specified`) in Table 3.2, which had already been identified in a previous description fragment, was identified for a second time in the Date range & conflict description. Since the goal was already specified, the second occurrence was considered redundant and was thus not added; Chapters 4 and 5 explain GBRAM heuristics that aid analysts in the identification of redundancies. The nine constraints identified for goal G_{12} (`Meeting Arranged`) are shown in Table 3.8. It should be noted that the constraints did not stem from the goals, but were instead identified from the textual descriptions in parallel to the goal identification process.

Extraordinary circumstances can be identified by considering scenarios

Scenarios facilitate the identification of special or extraordinary circumstances which occur so that goal and requirements information may be elaborated. Scenarios are identified

by considering the goals and goal obstacles previously identified in order to determine the reasons why a goal may fail and the circumstances under which a goal may fail. By asking *"Why?"*, *"What are the circumstances under which this obstacle can occur?"*, *"Why did this obstacle occur?"* and *"Why was this goal not achieved?"* scenarios which address the reasons for and consequences of failure may be identified. Consider goal G_{16} (`Meeting date determined`) in Table 3.9. By asking *"What are the circumstances under which this obstacle can occur?"* four scenarios were identified for obstacle #1 (`Meeting date not determined`).

Table 3.8. Constraints for Goal G_{12}: Meeting Arranged

Constraints
C_1: Meeting date defined by pair: calendar date and time period
C_2: Exclusion sets are all contained in the date range
C_3: Preference sets are all contained in the date range
C_4: Proposed meeting date belongs to date range
C_5: Proposed meeting within no exclusion set
C_6: Proposed meeting belongs to as many preference sets as possible
C_7: Meeting room available at Meeting date
C_8: Meeting room meets Equipment requirements
C_9: Meeting room belongs to as many as possible Important Participants' Location Preferences

Table 3.9. Scenarios for Meeting Scheduler Goal G_{16}

Goal	**Goal Obstacles**	**Scenarios**
G_{16}: Meeting date determined	1. Meeting date not determined	1.a Date conflict
	2. Preference sets not consolidate	1.b Room conflict
	3. Preference sets not provided	1.c Equipment not available
		1.d Important participant pref set not provided

Goals were also extracted from the problem definition document produced by Axel van Lamsweerde et. al. [87] which is referred to as the "System Requirements." The textual statements of need provided in the System Requirements were of a different flavor than those provided in the domain theory. The domain theory information is more widely applicable in that it is much more generalizable than the system requirements information. The system requirements information was much more specific and more implementation dependent. The argument may be made that domain theory information is more useful for identifying the high-level goals for the system due to its concern with the overall problem; in counterpoint, it may be said that more concrete goals and goal obstacles were identified from the system requirements, ultimately leading to a better understanding of the system. This may be due to the System Requirements' foundation on principles of actual functionality and system performance, which leads to more concrete descriptions and allows the system to be more clearly envisaged. However, it may also be the case that more goals and goal obstacles were identified from the system requirements simply due to the difference in length (52 lines of text versus 22). Table 3.7 shows the 23 goals identified from the system requirements.

One important issue in the meeting scheduler problem is the notion and occurrence of conflicts. From a general perspective, the concept of conflicts is specific to the domain of meetings and is not particular to the ontology of goals. However, in the domain of scheduling meetings, conflicts are analogous to goal obstructions in the general ontology of goals.

Chapters 4 and 5 demonstrate how the lessons learned from the meeting scheduler revisited were integrated into the Goal-Based Requirements Analysis Method.

3.4 Summary

This chapter presented the three case studies which served as the conceptual origin for the Goal-Based Requirements Analysis Method. Each case study detailed in this chapter involved a particular system:

- the Financial Services Office;
- Career Track Training System; and
- the Meeting Scheduler.

In the Financial Services Office process, goal decomposition and scenario analysis were investigated in the context of business process reengineering. The Career Track Training System (CTTS) required the reengineering of business processes spanning several inter-organization boundaries within a large enterprise. The meeting scheduler case required an analysis of user needs for a multi-user office application.

A synopsis of each project detailed in this chapter was followed by a discussion of the methodology and the lessons learned. The case studies served as a source of early validation, shaping the GBRAM through the lessons learned. The next chapter introduces the Goal-Based Requirements Analysis Method in detail.

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.

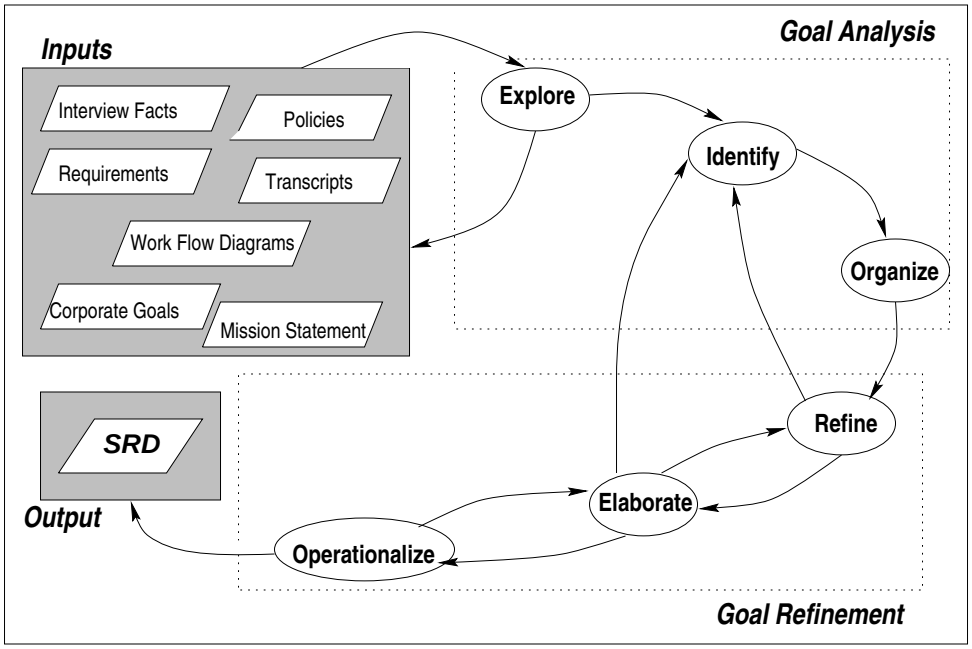

Figure 4.1. Overview of GBRAM Activities

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.

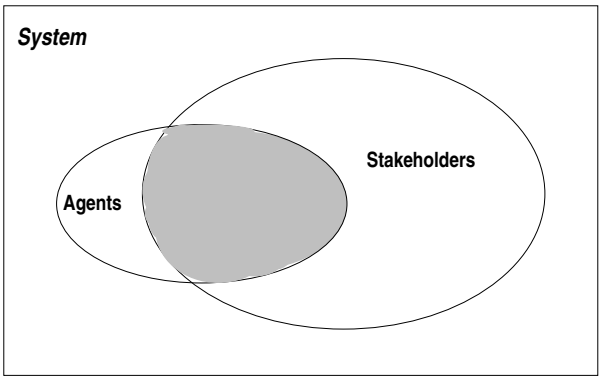

Figure 4.2. Venn Diagram Distinguishing Agents and Stakeholders

[†]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.

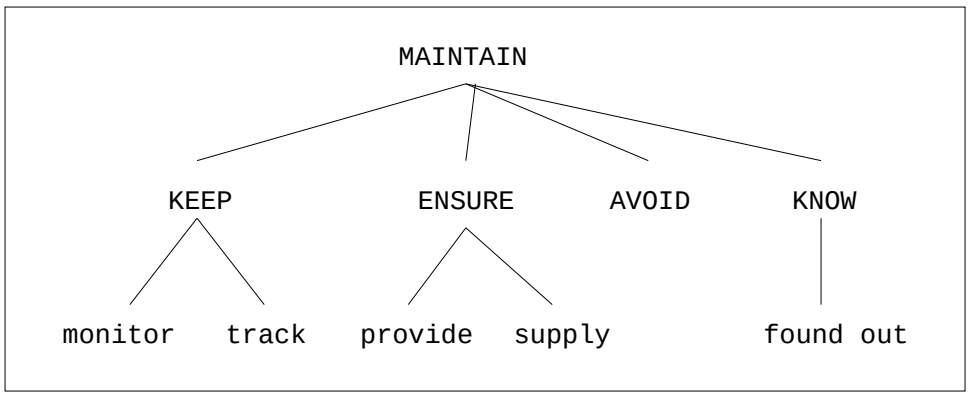

Figure 4.3. Useful Key Words for Classifying Maintenance Goals

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.

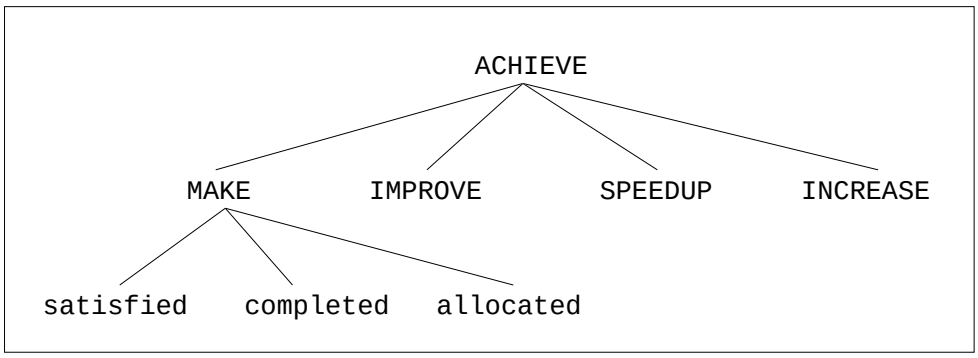

Figure 4.4. Useful Key Words for Classifying Achievement Goals

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).

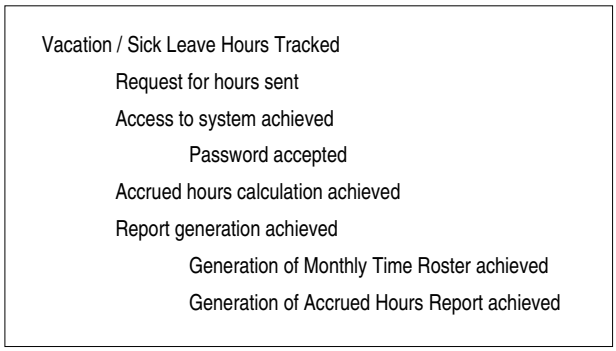

Figure 4.5. Goal Topography in Outline Form for Vacation/Sick Leave System

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.

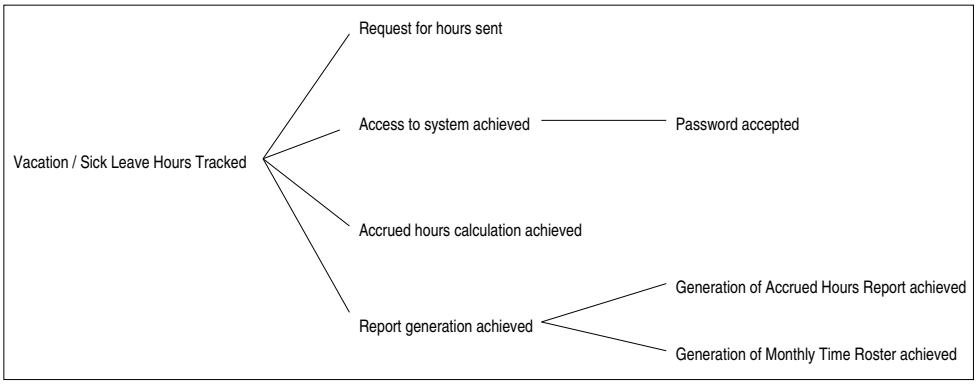

Figure 4.6. Goal Topography in Hierarchy Form for Vacation/Sick Leave System

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.

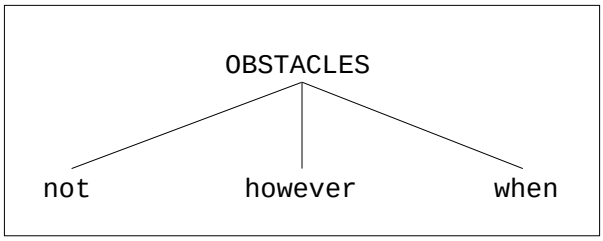

Figure 4.7. Some Useful Key Words for Identifying Obstacles

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 $G_1 \not<_P G_2$) (see Example 4.15). When a goal fails because the responsible agent is irresponsible, it is considered an *agent failure* (A) obstacle (expressed $G_1 \not<_A G_2$) (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 $G_6 \not<_P G_8$ 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.

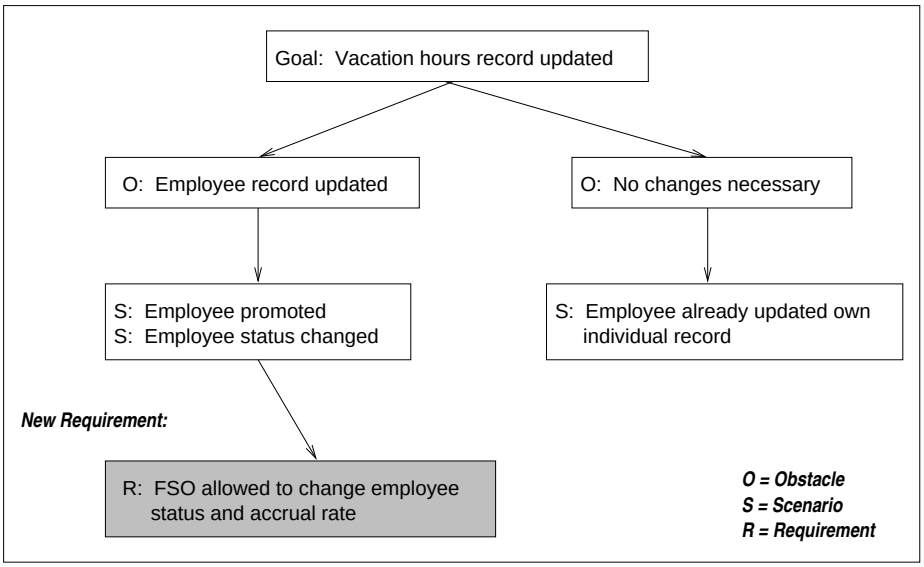

Figure 4.8. Uncovering New Requirements for Vacation/Sick Leave System

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).

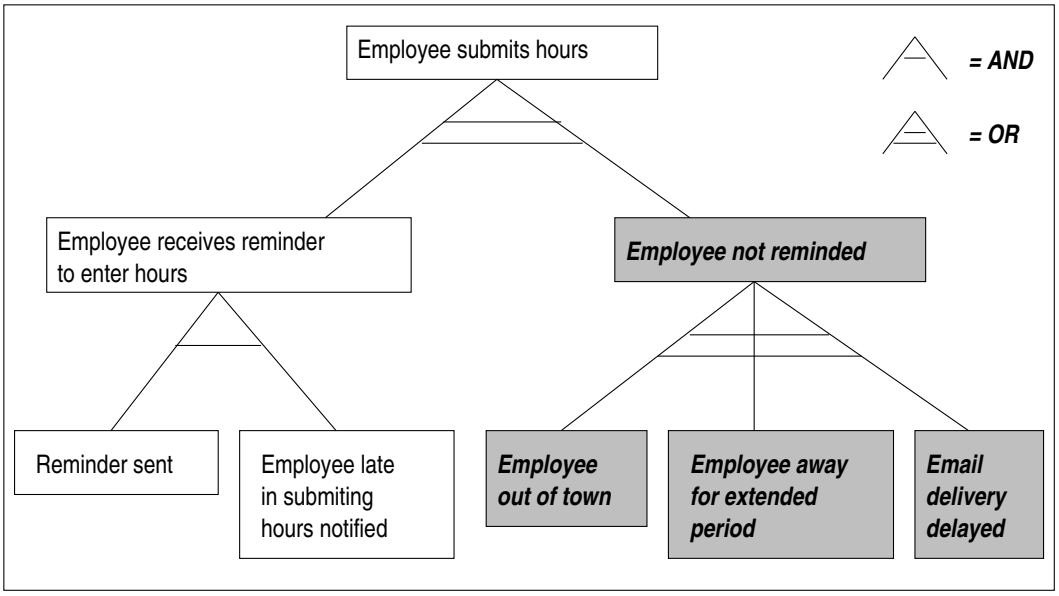

Figure 4.9. Considering Exceptional Cases

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.

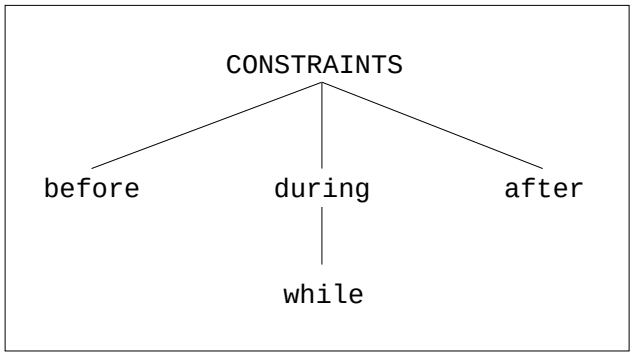

Figure 4.10. Some Useful Key Words for Identifying Constraints

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.

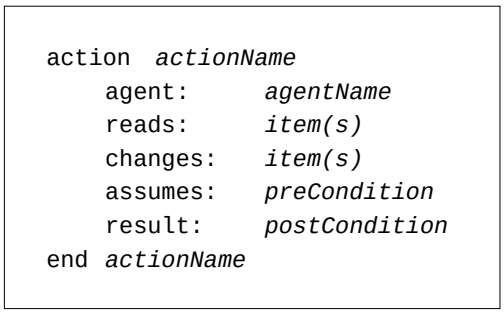

Figure 4.12. Operational Definition Syntax

- **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 $\mathfrak{E}$ 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

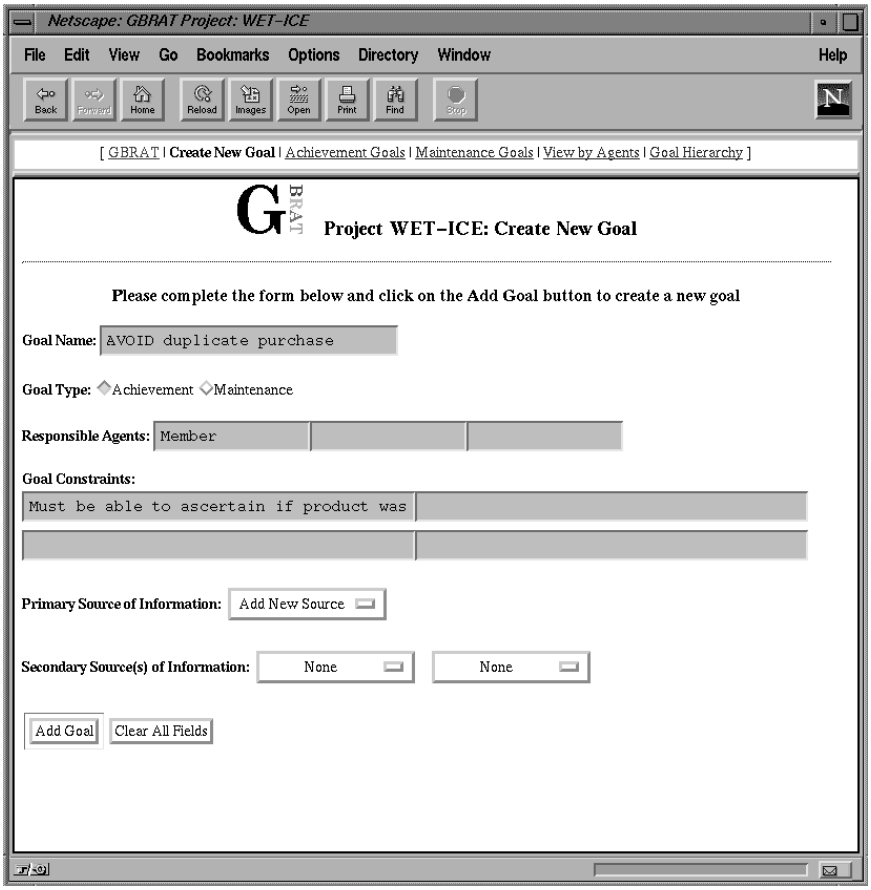

Figure 4.20. GBRAT Form for Creating Goals

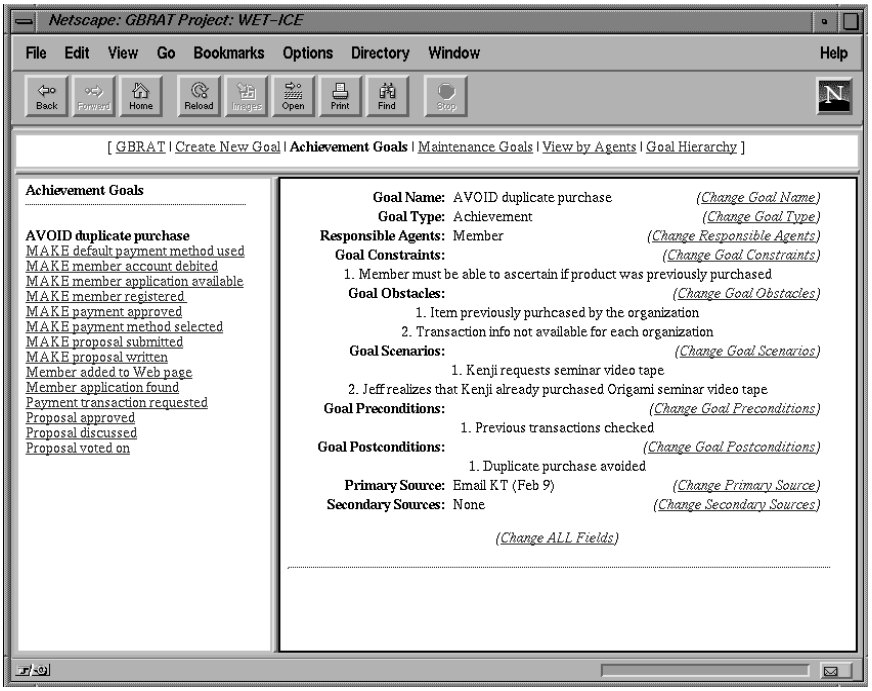

Figure 4.21. Viewing Achievement Goals

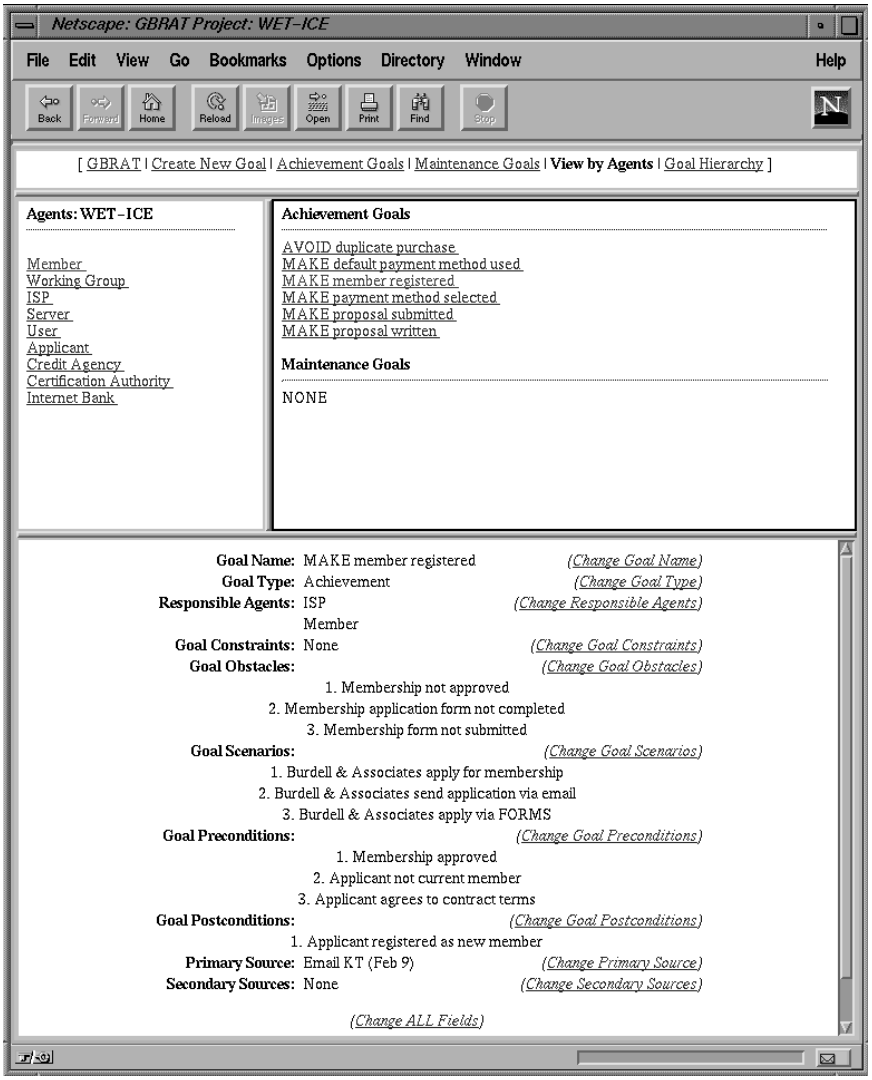

Figure 4.22. Viewing Goals by Agent

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

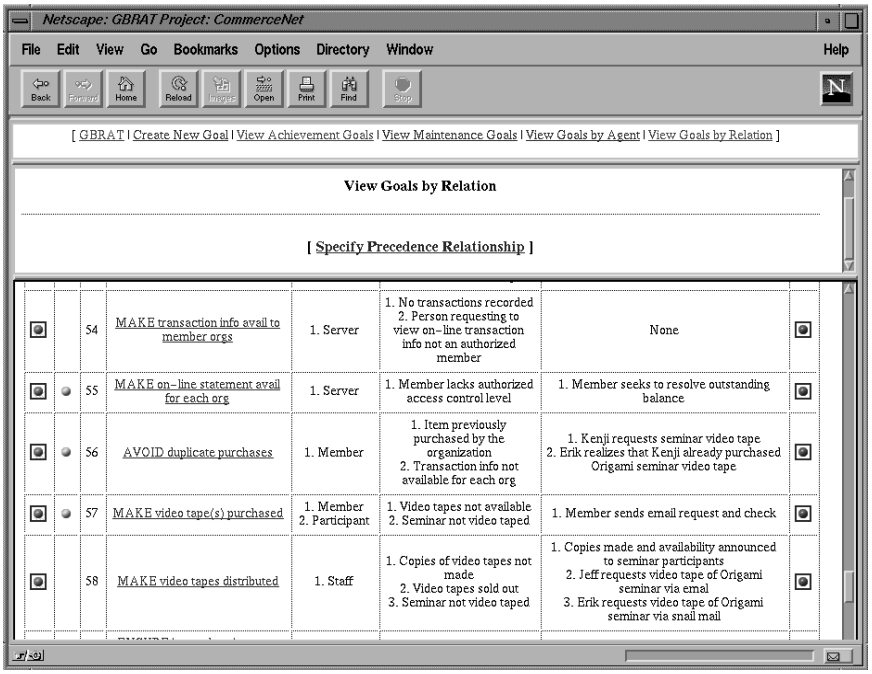

Figure 4.23. Viewing Goals by Precedence Relation

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.

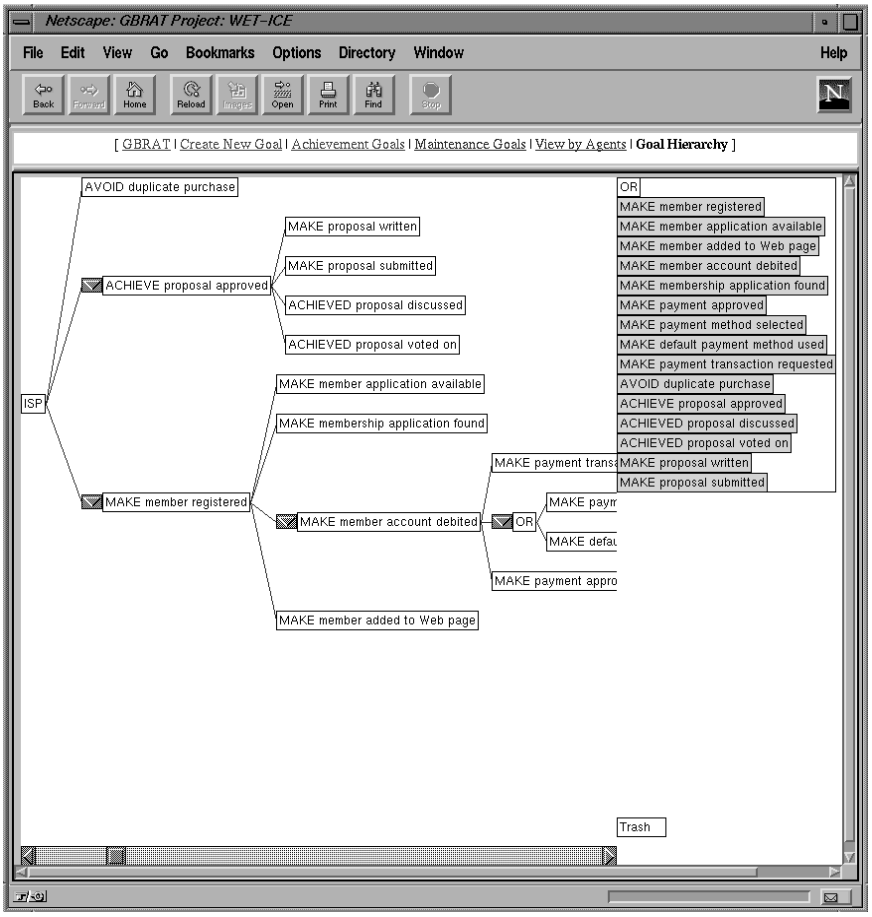

Figure 4.24. Goal Hierarchy

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.

CHAPTER 5

Heuristics and Guidelines

The most important function that software builders do for their clients is the iterative extraction and refinement of the product requirements.

Frederick P. Brooks, Jr.

The objective of this chapter is to describe some typical heuristics used by analysts applying the Goal-Based Requirements Analysis Method. In the GBRAM, heuristics are rules which guide analysts towards a high probability of success while avoiding wasted efforts. The GBRAM provides sets of heuristics and guidelines for the identification of goals to be used in requirements specifications. The selection of heuristics depends upon the type of system desired and on the information available to the analyst.

There are four general types of heuristics employed by analysts using GBRAM:

- identification heuristics;
- classification heuristics;
- refinement heuristics; and
- elaboration heuristics.

Identification heuristics assist analysts in identifying goals, stakeholders, agents, and constraints from multiple sources. The objective of goal classification heuristics is to aid

analysts in determining the type of each goal identified. Refinement heuristics employ a series of questions and techniques to reduce the size of the goal set. Elaboration heuristics address the need to acquire more detailed information by considering goal dependency relations, suggesting the goal obstacles for which scenarios should be constructed and which scenarios to elaborate. Although there is a distinction between refinement and elaboration, as discussed in Section 5.4, GBRAM recognizes the concurrent and overlapping nature of these activities. As discussed in Chapter 3, the lessons learned in the initial case studies served as the origin for the ideas which formulated the Goal-Based Requirements Analysis Method. The heuristics in this chapter were derived from these experiences and observations.

Section 5.1 presents a set of recurring question types to aid analysts in applying an inquiry-driven approach to goal-based analysis. Since the utilization of the heuristics depends upon the particular GBRAM activity with which the analyst is involved at any given time, each heuristic set (5.2, 5.3, 5.4, and 5.5) is presented in its own section, with discussion of the application of the heuristics to specific GBRAM activities.

5.1 Goal-Based Instantiation of Inquiry Cycle

The GBRAM heuristics detailed in this chapter are a set of rules. Some of these heuristics are straightforward, not requiring the analyst to employ any specific inquiry process. In contrast, other heuristics are meaningless without the accompanying questions to guide analysts in developing a deeper understanding of the system and uncovering hidden goals and requirements. Often a simple answer to a question is insufficient; for example,

justification or rationale for a particular response may be needed to understand the system. Thus, GBRAM offers a set of recurring question types which follow the inquiry cycle approach [61, 64] instantiated for goal-based analysis, and guide analysts in applying an inquiry-driven approach as they comprehensively investigate the available artifacts for descriptive answers. This section discusses these question types; subsequent sections in this chapter suggest appropriate applications and resolutions based upon the answers derived from the questions asked.

Recall from the discussion in Chapter 2.2 that the Inquiry Cycle model is a formal structure for describing discussions about requirements [61]. The types of questions asked about a set of goals and requirements were investigated during the case studies discussed in Chapter 3. These recurring question types are summarized below and discussed throughout the remainder of this chapter.

- *What-is:* These questions request specific information regarding terminology which is unclear to someone with no knowledge of the application domain. For example, analysts may ask a stakeholder, *"What is the process for deciding how to organize and structure information in an intranet server?"* In any analysis effort, participants in the process have to develop a common understanding of the concepts and terms. What-is questions enable analysts (and stakeholders) to do precisely this. What-is questions clarify a situation or scenario enabling analysts to ensure that something is understood correctly. This type of question is most useful when the analyst is able to interact directly with the stakeholders.

- *Who-is:* These questions request specification of the agent responsible for the given task, process or goal. In Business Process Reengineering (BPR) cases it is often the case that one of the prevailing inefficiencies in an organization is the number of approvals and authorizations [levels] that are required. It is helpful to ask questions such as, *"Who is responsible for the ultimate decision?"* and *"Who is responsible for this task?"* By asking who-is questions, analysts can acquire information about the various agents and tie the information to the corresponding goal based on implied or explicit responsibility.

- *Why:* These questions request *reasons* which underlie work activities. This information is essential when redesigning processes. Analysts must ask questions such as *"Why is this information routed?"* While it is relatively simple for analysts to determine *what* information is required to route, the reason why the information is routed can be very difficult (if not impossible) to ascertain without direct stakeholder inquiry. These questions may be application specific, but may be generalized since it is critical in BPR efforts to ask *why?* questions.

- *What-if:* It is beneficial to ask questions that enable analysts to further examine cases in which an unexpected action occurred in order to explore how other system features may be affected. An example is *"What happens if an individual drops or cancels out of a class?"* This kind of question prompts consideration of whether this new opening in a class has any bearing on, for example, students who were previously turned away from the class because the course had reached its registration capacity. What-if questions mandate the consideration of the other agents and processes that would be affected in the event of such an unexpected cancellation.

- *When:* These questions request timing constraints for a given event or events. For example, it may be unclear when an electronic credit payment approval request is subsequently reviewed again after previously failing to gain approval. It may be important to know when requests for payment approvals are reviewed. To ascertain this type of information, analysts ask questions such as, *"When is an electronic credit payment approval request reviewed?"* which may be followed up with a clarification question such as: "Is it reviewed immediately or at the beginning of the next working day?" In Business Process Reengineering efforts, this type of information is critical because it enables analysts to identify candidates for redesign so that unnecessary delays may be eliminated.

- *Relationship:* These questions ask how one agent is related to another or how one goal is related to another goal so that the dependency relations may be established. For example, analysts may consider each goal and ask: *"What goals are prerequisites for this goal?"*, *"What goals must follow this goal?"*, and *"What agent depends on this goal for completion of their responsibilities?"*

These question types assist analysts in knowing when and how to apply the GBRAM heuristics by providing a guide as to how much detail is needed before one can be reasonably confident that the goals have been fully elaborated and that any hidden goals or requirements have been uncovered.

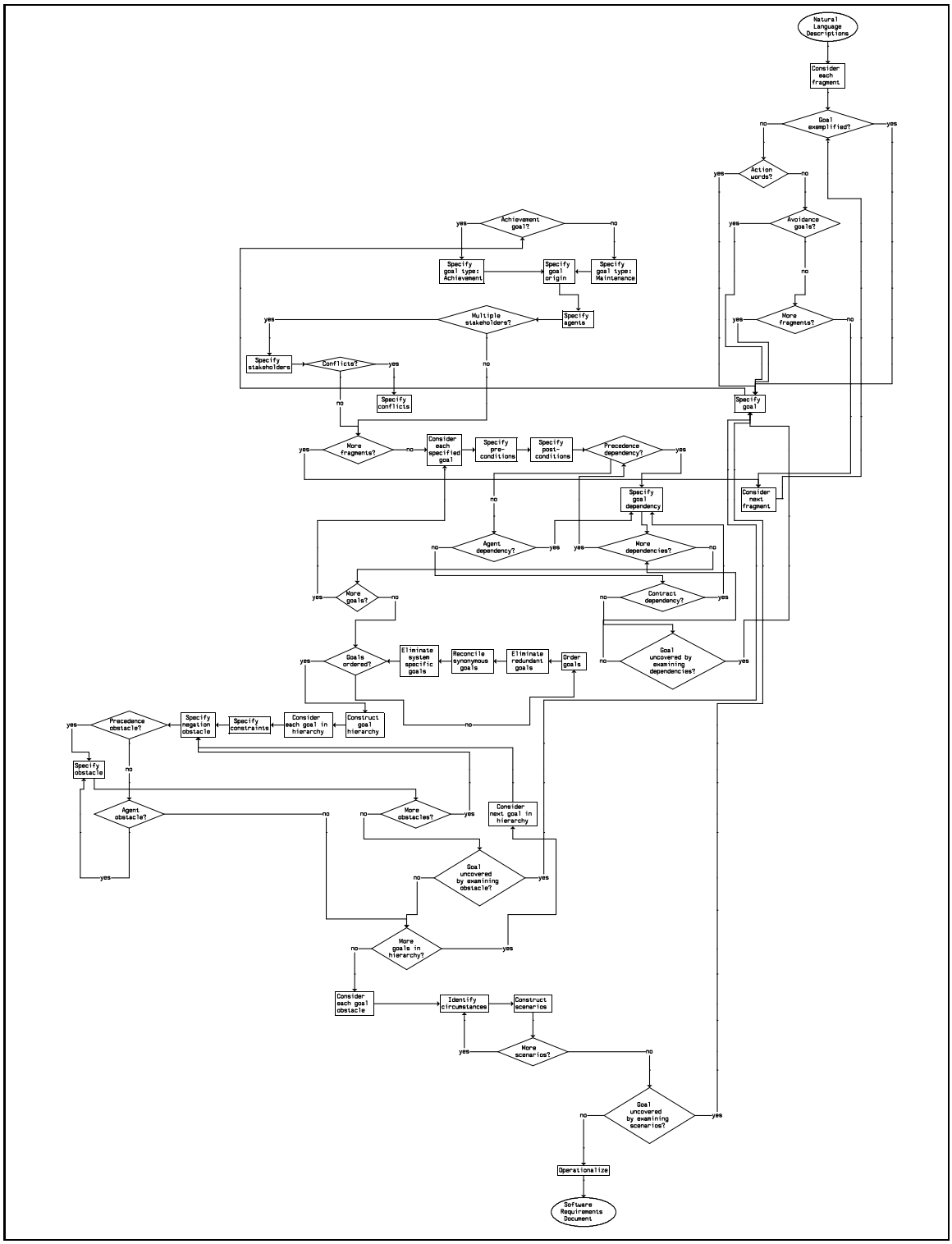

Figure 5.1. GBRAM Control Flow Chart

Figure 5.1 provides a reduced overview of the Goal-Based Requirements Analysis Method, represented in a control flow chart. The flow chart is an approximation of the method; it is presented here to show the overall scope of the method. Each part of this flow chart is presented in expanded form throughout the remainder of the chapter and is discussed within the context of the specific inquiry points for each set of GBRAM heuristics. Each of the activities shown in Figure 4.1 on page 69 are represented in this control flow chart. The boxes in Figure 5.1 represent the activities that an analyst must perform. The diamonds represent the various inquiry points for an inquiry-driven approach throughout the GBRAM process, while the ellipses represent the input (natural language descriptions) and output (a SRD) of the method. The arrows denote the flow throughout the process as well as the iterative nature of goal-based analysis. Boxes and ellipses have only one output edge; however, since the diamonds represent inquiry points, a decision is required of the analyst. Thus, diamonds have two outgoing edges; one edge represents an answer of 'yes,' and the other edge indicates an answer of 'no'. A detailed discussion of the types of questions analysts ask during inquiry is provided in earlier in this section. The decision points, as shown by the diamonds in Figure 5.1, are discussed throughout the remainder of this chapter. Since Figure 5.1 is compressed to allow for an overview of the complete process, the discussion of each set of heuristics is accompanied by the relevant portions of Figure 5.1 for the elucidation of the possible sequences of heuristic and inquiry application detailed in this chart. Again, it must be emphasized that the flow charts throughout this chapter provide an approximate representation of the method; a more detailed description of the GBRAM activities is available in Chapter 4. Since many of the activities may be performed concurrently, the flow charts simply serve to demonstrate how the GBRAM instantiates the

inquiry cycle [64].

Each heuristic set addresses a problem space, as discussed in the text preceding each set of heuristics in this chapter. Many examples have already been presented in Chapter 4; however, some additional supporting examples for the heuristics not previously discussed throughout this thesis are provided in this Chapter. The remainder of this chapter presents the four sets of GBRAM heuristics and provides guidelines for the appropriate time and manner for application. Table 5.1 provides a glossary for the labels which serve as tags for the identification of each of the four heuristic sets.

Table 5.1. Glossary of Heuristic Identifier Codes

Code	Definition
HIG	Heuristic for Identifying Goals
HIS	Heuristic for Identifying Stakeholders
HIA	Heuristic for Identifying Agents
HIC	Heuristic for Identifying Constraints
HCM	Heuristic for Classifying Maintenance
HCA	Heuristic for Classifying Achievement
HRR	Heuristic for Refining Redundancies
HRS	Heuristic for Refining Synonymous
HRSS	Heuristic for Refining System-Specific
HED	Heuristic for Elaborating Dependencies
HEO	Heuristic for Elaborating Obstacles
HES	Heuristic for Elaborating Scenarios

5.2 Identification Heuristics

This section presents four sets of identification heuristics in GBRAM which may be summarized as follows:

- the *goal identification heuristics* provide strategies for identifying and extracting goals from various information sources;
- the *stakeholder identification heuristics* guide the process of identifying all parties claiming an interest in the proposed system;
- the *agent identification heuristics* provide strategies for allocation of goal responsibilities; and
- the *constraint identification heuristics* offer strategies for identifying conditions that must exist or be met for a goal to be realized or completed.

As discussed in Chapter 4, the GBRAM identification activities (Figure 4.1, page 69) involve the identification of goals, stakeholders, and agents. Another GBRAM identification activity involves constraints. Although constraint identification was introduced late in Chapter 3, this activity is a recurring task for the analyst. As such, analysts may identify constraints as early as during the initial exploration of documentation or as late as during the operationalization of goals into requirements. The constraint identification heuristics are included in this section of the chapter because this activity is not limited to any one phase of GBRAM.

The following subsections provide a discussion of these identification heuristics within the context of the associated activities shown in Figure 4.1.

Heuristics for Goal Identification

The goal identification heuristics address the problem of how to extract goals from the documentation and resources available to the analyst. As discussed in Chapter 4, goal analysis is the process of exploring gathered documentation, ranging from information about the organization (i.e., enterprise goals) to system-specific information (i.e., requirements),

for the purpose of identifying, classifying, and organizing goals. It is often assumed that software systems are constructed with a purpose or goal(s) in mind [27]; the heuristics in this section address the origin of goals when their origin is initially unknown and what happens when the goal or purpose is not clear to the analyst.

As previously discussed in Chapter 3 on page 74, analysts must work with various information sources. When reengineering legacy systems, an existing requirements document may be used as a starting point for extracting goals and objectives. However, such a document may not be available, requiring the goals to be deduced from the obtainable sources. Goals may be extracted or identified from a number of different sources, as shown in Figure 5.2.

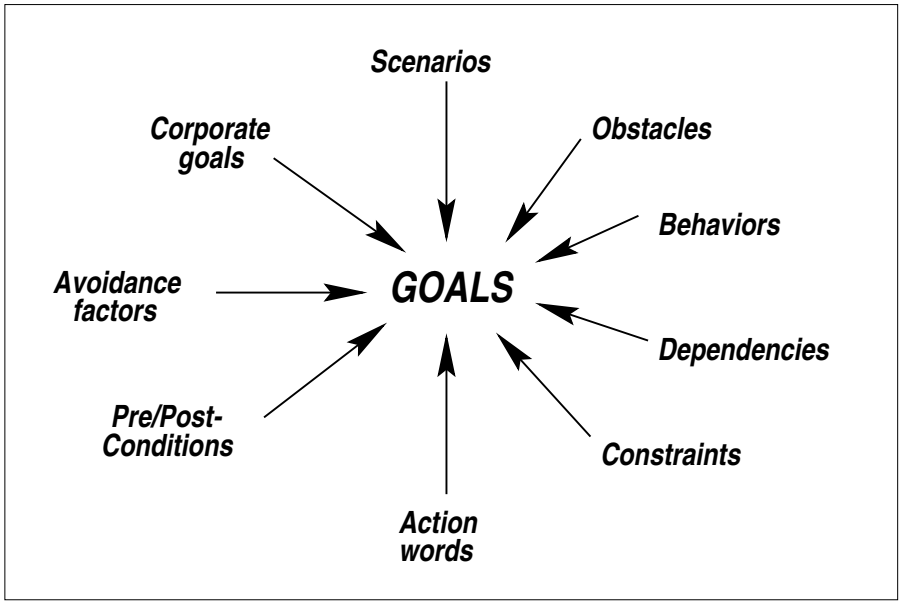

Figure 5.2. Origin of Goals

Goal identification is not limited to the initial activities of the Goal-Based Requirements Analysis Method; goals may be identified throughout the entire analysis process (e.g. during the identification of goal dependencies, obstacle analysis, and scenario analysis). Figure 4.1 on page 69 illustrates how goals may be identified during goal refinement and elaboration. The goal identification heuristics address this 'delayed' uncovering of new goals. For example, goals may be identified by considering constraints, as shown in Example 5.13 on page 150, or scenarios, as shown in Example 4.19 on page 103.

The general heuristics for identifying goals and objectives, presented below, should be considered by analysts when exploring existing documentation. Those heuristics which are not explicit are accompanied by a series of questions to guide the analyst. Figure 5.3 illustrates a possible series of inquiry points for an analyst applying the goal identification heuristics to extract goals from available documentation.

HIG 1. Goals are named in a standardized subset of natural language in which the first word is a verb that describes the kind of goal being named. For example, `AVOID` denotes one kind of goal. Goals of this kind are satisfied as long as their target conditions remain false.

Example 5.1 In the electronic commerce WWW server discussed in Chapter 6.2, stakeholders expressed the need to prevent duplication of purchases. The analyst must specify a goal name such as `AVOID duplicate purchase`.

HIG 2. Abstraction mechanisms may be employed to extract goals from available documentation by asking:

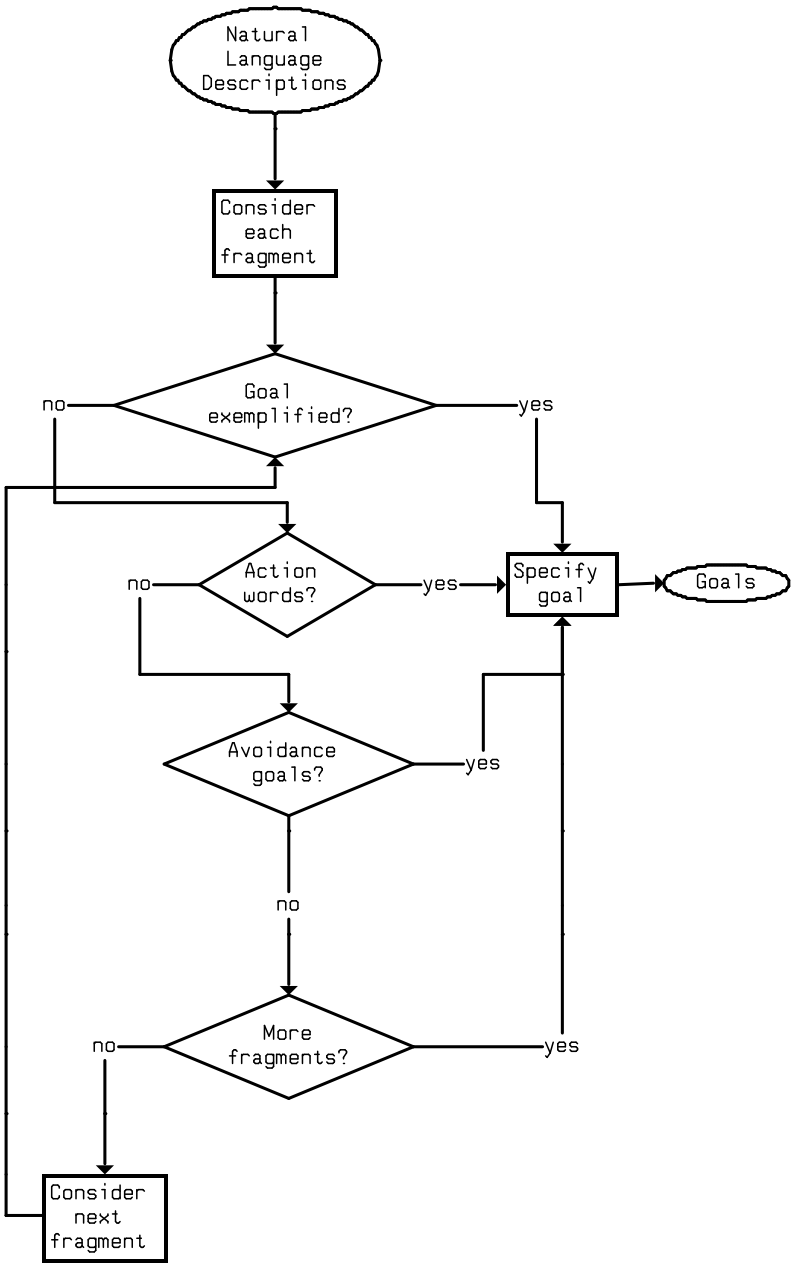

Figure 5.3. Control Flow Chart for Goal Identification

(a) What goal(s) does this statement exemplify?

(b) What goal(s) does this statement block or obstruct?

If the answer to either of these questions is yes, then express the statement as a goal which represents a state that is desired or achieved within the system. The inquiry points are shown in Figure 5.3.

Example 5.2 (Using abstraction mechanisms to identify goals) In the bug evolution example (See Appendix B; page 241), the goal `Population stabilized` is identified by using abstraction mechanisms to identify goals.

HIG 3. Action words (see Figure 5.3) that point to some state that is or can be achieved once the action is completed are candidates for goals in the system. They are identified by considering each statement in the available documentation by asking:

(a) Does this behavior or action denote a state that has been achieved, or a desired state to be achieved?

If the answer is yes, then express the answer to these questions as goals which represent a state that is desired or achieved within the system.

Example 5.3 (Using action words to identify goals) Consider the bug evolution description in Appendix B. *Move*, *eat*, *die*, and *reproduce* are all action words. Recall that if a statement denotes an action or behavior, it must be restated so that it denotes a state that has been achieved or is desired (a goal). For example, `bug moves` should be restated as `bug moved`.

HIG 4. An effective way to uncover hidden goals is to consider each action word and every description of behavior and persistently ask *"Why?"* until all the goal have been 'treated' and the analyst is confident that the rationale for each action is understood and expressed as a goal. The action words should be restated so that they denote a state that has been achieved or a desired state.

Example 5.4 (Questions for action words) Given the statement *weight gained*, the analyst would inquire as to why the bug gained weight. Given the description in Appendix B, one can determine that the bug gained weight because the bug ate bacteria (*bacteria eaten*). While this technique may seem obvious, it has been applied successfully to several system analysis efforts (Refer to Chapter 3; page 35).

HIG 5. Key action words such as: *track, monitor, provide, supply, find out, know, avoid, ensure, keep, satisfy, complete, allocate, increase, speedup, improve, make,* and *achieve* are useful for pointing to candidate goals.

Example 5.5 Example 4.2 on page 76 illustrates how the identification of the action words *coordinate*, *track*, and *improve* led to the identification of the goals `Training coordinated`, `Progress tracked`, and `Qualifications improved`.

HIG 6. If a statement seems to guide design decisions at various levels within the system or organization, express it as a goal.

Example 5.6 Example 4.1 on page 75 illustrates how the statement "DoD must ... spend tax payers' money ... more effectively and efficiently" affects decisions through-

out the organization, and can be expressed as the goal `Tax payers money spent efficiently`.

Analysts may find this mode of express awkward for this particular goal. This is mainly a consequence of GBRAM stating goals as state expressions. If this creates confusion for the user, the goal may be expressed in a manner with which they feel comfortable.

HIG 7. Goals may be uncovered by examining available information fragments to identify avoidance goals*. Avoidance goals are found by identifying bad states or states that should be avoided within the system.

Example 5.7 (Identifying goals by considering avoidance factors) In a budget system, `Avoid overdrawn account` is an avoidance goal.

HIG 8. Goals can be uncovered or discovered by considering the goal dependencies for the previously specified goals by asking:

(a) What are the pre-conditions† of this goal?

(b) What are the post-conditions‡ of this goal?

Since preconditions and postconditions are expressed as goals in GBRAM, it is possible to identify new goals that had not been previously considered or identified by considering each goal's dependencies.

---

*_Avoidance goals_ are satisfied as long as their target conditions remain false.
†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.

Example 5.8 Example 4.22 on page 107 illustrates how goals may be discovered by considering the potential postconditions of a goal.

HIG 9. Stakeholders tend to express their requirements in terms of operations and actions rather than goals [7, 25]. Thus, when given an interview transcript, it is beneficial to apply the action word strategy to extract goals from stakeholders' descriptions.

Example 5.9 (Extracting goals from stakeholder descriptions) In a meeting scheduler system, stakeholders may use action words such as 'schedule' and 'reserve,' which give rise to goals such as `Meeting scheduled` and `Room reserved`.

HIG 10. Analysts should first seek to understand the stakeholder's application domain and goals before concentrating on the actual or current system so that the system requirements may be adequately specified. Previous research indicates that customers tend to express their goals within the context of their application domain, not in terms of an existing or desired system [7].

Example 5.10 The goal of a college financial services system is not to maintain a financial ledger/database (system goal) as typically described by management level stakeholders, but to ensure that, among other requirements, the budget remains balanced; sponsors are charged according to their contracts; and faculty are paid according to state research contracts, as typically described by customers using their application domain vocabulary.

HIG 11. Goals are also identified by considering the possible goal obstacles for previously specified goals.

Example 5.11 Example 4.20 on page 105 illustrates how the goal `Course closed` was identified by considering the possible obstacles for the goal `Available course slots announced`.

HIG 12. Goals may be identified by considering possible scenarios.

Example 5.12 Example 4.21 on page 106 illustrates how an analysis of the scenario "Employee not reminded" facilitates the identification of new goals to handle the exceptional cases represented by the scenarios.

Given each goal obstacle, the analyst should determine whether or not the occurrence of the goal obstacle would cause the system to fail. If the occurrence of the goal obstacle would initiate system failures, these obstacles are key candidates for scenario construction and analysis since the analyst must be sure to specify the goals and requirements to enable the system to handle exceptional cases. Goals may also be identified by considering the normal non-exceptional scenarios.

HIG 13. Goals may be identified by considering constraints.

Example 5.13 (Identifying a goal from a constraint) Several statements in the Career Track Training System (Chapter 3.2; page 45) describe requirements that have to be met in order for a goal to be achieved. Consider the circumstances illustrated in Example 4.25 on page 110. A review of NLD #3 on page 81 leads to the identification of the constraint `Course must qualify employee to advance to another level`; the system must 'know' which courses an employee can take. Thus, the goal

`Courses which employee qualifies for identified`) can be identified from this constraint.

Example 5.14 In the CTTS case study, several statements describe requirements that have to be met in order for a goal to be achieved. For example, consider the CTTS Process Scenario: a review of NLD #3 (shown on page 81) points to the identification of the constraint `Course must qualify employee to advance to another level`. The word 'qualify' was the key indicator that a requirement must be met. Before an employee can advance to a certification level, the course taken must officially qualify them for advancement.

HIG 14. Goals may be extracted from process diagrams by searching for actions and behaviors, as well as by consistently applying the Inquiry Cycle [64] to clarify the goals and requirements.

Example 5.15 (Extracting goals from process diagrams) In the CTTS example (Chapter 3.2; page 45), in-depth interviews were conducted with AFB personnel, as well as with professionals in the training acquisition process, to develop an understanding of the current process. These interactions resulted in the construction of an informal, but detailed flow chart model of the current process; a portion of the flow chart for a career training system is shown in Figure 5.4. This particular flow chart provides a good bit of detail so it is possible to extract goals using action word location as well as by identifying behaviors in the system. Given the level of detail provided in this flow chart, 14 goals were extracted, as detailed in Table 5.2.

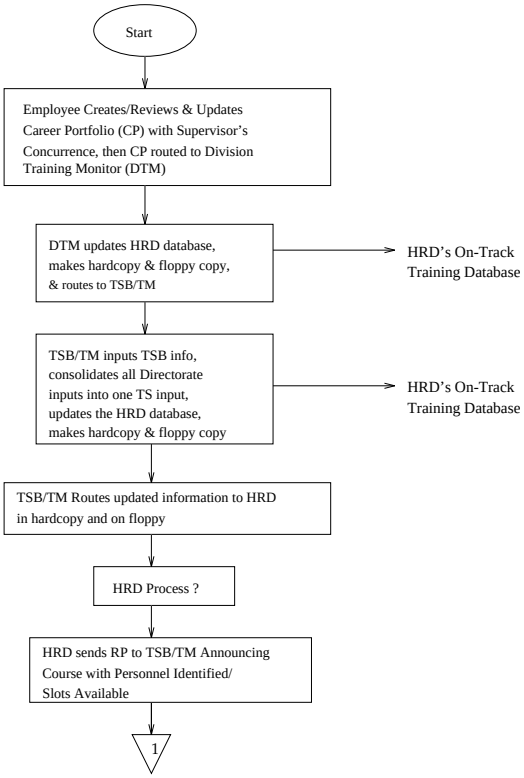

Figure 5.4. Flow Chart for a Portion of the CTTS

Table 5.2. Goals Extracted from the CTTS Flow Chart

Goals	Agent
G_1 : Career portfolio created	employee
G_2 : Career portfolio reviewed	employee
G_3 : Career portfolio updated	employee
G_4 : Career portfolio routed to DTM	employee
G_5 : Database routed to TSD	HRD
G_6 : Database routed to HRD	TSD
G_7 : IP sent to TSD	HRD
G_8 : Database updated	HRD, TSD
G_9 : Database updated with TSD info	TSD
G_{10} : Database hard copied	HRD, TSD
G_{11} : Database floppy copied	HRD, TSD
G_{12} : Directorate inputs consolidated	TSD
G_{13} : HRD process completed	HRD
G_{14} : Course & personnel matched	HRD

The goal identification heuristics offer analysts several approaches for extracting goals, based upon the resources available to the analyst. A summary of these heuristics is provided in Tables C.1 and C.2 (Appendix C; pages 244 and 245). The stakeholder and agent identification heuristics guide the process of considering the stakeholders of the proposed system and the agent responsibilities. The constraint identification heuristics allow analysts to identify conditions that must exist or be met for a goal to be realized. These heuristics are discussed in the following subsections.

Heuristics for Stakeholder Identification

As previously discussed in Chapter 4 on page 78, anyone who claims an interest in a proposed system is a stakeholder, while anyone or anything responsible for the actual completion of goals within an organization or system is an agent. The heuristics for stakeholder identification address the problem of determining who and what parties claim an interest in the proposed system. A stakeholder need not only be a user in the classical sense; rather, any representative affected by the completion or prevention of a goal may be considered a stakeholder. A stakeholder may be a customer, actor, owner, or representative of an organization (Refer to Examples 4.3 and 4.4, found on page 79).

Although goal identification is discussed prior to stakeholder identification, the focus in this thesis and GBRAM is on the activities, not on the sequence of activities. However, there are times when stakeholders must be identified before the goals and objectives can even be established. For example, if the manager of a department in a telecommunications company requests an information system, there may be no existing documentation or description of

the desired system. The starting point for the analyst thus becomes an in-depth interview with the telecommunications department manager or "customer" to determine who the stakeholders are so that the analyst may begin interviewing those persons. Requirements elicitation is not the focus of this dissertation; however, this has been addressed extensively in other literature [22,23,28,33,35].

Stakeholders are identified to determine which agents claim a stake in each goal and to develop an understanding of the different viewpoints involved in the system for conflict resolution. Stakeholder identification is important for conflict resolution and for prioritizing goals in that capturing stakeholders' viewpoints allows conflicts to be detected early. Analysts should consider the role that stakeholders play in prioritizing goals during negotiation [65]. In the event of conflicting requirements, the respective goals should be used to guide the process of resolving the conflict. While a balanced resolution is not always possible, the goals may have different priorities or 'importance' ratings that may guide the process to determine the most feasible trade-off.

As illustrated in Figure 5.5, stakeholders may be identified before and/or after goal identification, enabling conflicts to be surfaced early. When a conflict is detected, or when there are multiple stakeholders, the analyst should apply the inquiry cycle, using some of the questions in this section to clarify the goals and the reason for conflict.

HIS 1. Systems or subsystems which do not involve multiple stakeholders may not require stakeholder identification; analysts may choose to skip stakeholder identification entirely in these systems.

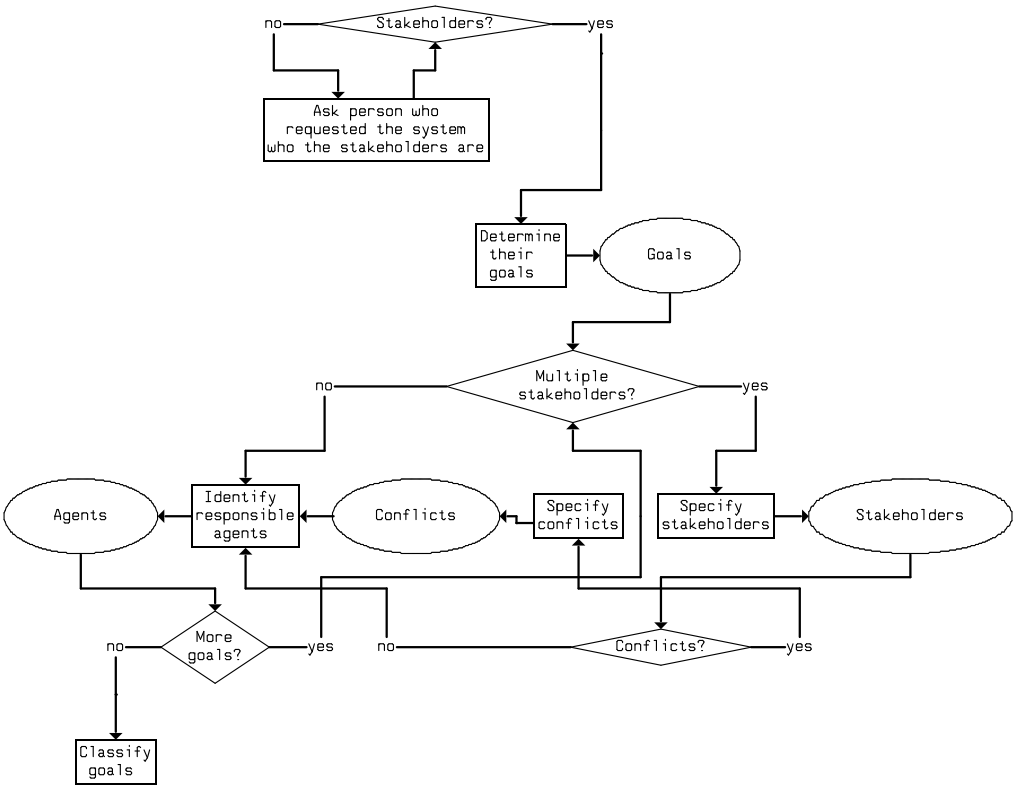

Figure 5.5. Control Flow Chart for Stakeholder and Agent Identification

Example 5.16 In GBRAT (Detailed in Chapter 6.1; page 187), there was only one stakeholder at the time of development. Thus, stakeholder identification was deemed unnecessary.

HIS 2. Multiple stakeholders may be associated with one goal.

Example 5.17 Example 4.3 on page 79 illustrates this association of multiple stakeholders with one goal. In this example, the goals `Skills improved` and `Career tracks provided` both have two stakeholders (AFB and employee).

If different stakeholders are associated with a goal but their associations occur at different times within the life of the system, the analyst should document these variances to ensure that the role of stakeholders throughout the lifetime of a goal or the system is well understood.

HIS 3. Any representative affected by the completion or prevention of a goal is a stakeholder. A customer or person representing the enterprise requesting the system or an analysis effort is a stakeholder. Users of the proposed system are stakeholders. Stakeholders are thus identified by asking:

(a) Who or what claims a stake in this goal?

(b) Who or what stands to gain or lose by the completion or prevention of this goal?

(c) Who will use the system?

Example 5.18 Consider the vacation/sick leave problem description in Appendix A. `Hour tracked` has more than one stakeholder and these stakeholders are identified by

asking the questions above. The stakeholders for this goal are: academic institution, employee, financial services office, and payroll office.

The goal identification heuristics allow analysts to extract goals from various sources of information; stakeholder identification heuristics allow analysts to consider all parties who claim an interest in the system. A summary of the heuristics for stakeholder identification is provided in Table C.3 (Appendix C; page 246). Agents must be identified so that responsibility for ensuring the achievement of a goal at any given time may be determined and assigned. The following subsection presents the agent identification heuristics.

Heuristics for Agent Identification

Every goal has at least one responsible agent, be it a person, organization, or even the system. The agent identification heuristics presented below address the problem of determining who or what the analyst should allocate responsibility to for the goals.

HIA 1. At least one agent must be responsible for the completion of each goal. If the analyst is unable to allocate responsibility for a goal to any agent, then the analyst can assume that the goal lies outside the scope of the proposed system. If the analyst believes there is a responsible agent, but doesn't know who or what, then the inquiry cycle should be applied using the Who-is question.

HIA 2. Responsible agents may be identified by considering each goal and asking:

(a) Who or what agent *is*, *could be*, or *should be* responsible for this goal?

The answer to this question will be the name of the responsible agent. The agent's name should be 'attached' to the goal for which it is responsible; Table 4.4 on page 81 illustrates how agents can be attached to goals using a tabular notation.

Example 5.19 Example 4.5 on page 81 illustrates how the responsible agent (employee) was identified for the goals `Course completed` and `Proof of course completion submitted` by asking the questions in HIA 2 for the goals

Example 5.20 Consider the bug evolution description in Appendix B. For each goal, answer the question: *Who or what agent is responsible for this goal?* For example, `bug` is responsible for the goal `Bacteria eaten`.

HIA 3. Different agents can be responsible for the completion of the same goal at different times.

HIA 4. Agents may be either the system, organization, or a human agent.

Example 5.21 In a meeting scheduler system the goal `Meeting scheduled` is the responsibility of the meeting scheduler. Depending on the desired implementation, the agent may be either the automated system or a human agent. In an email-based implementation of this system the responsible agent could be someone other than the automated meeting scheduler system such as a member of the clerical staff.

Agents must be identified so that responsibility for ensuring the achievement of a goal at any given time may be determined and assigned. A summary of the heuristics for agent identification is provided in Table C.3 (Appendix C; page 246). The following subsection presents the heuristics for constraint identification.

Heuristics for Constraint Identification

A constraint, as discussed in Chapter 4 on page 110, places a condition on the achievement of a goal. In GBRAM, statements which seem to be independent of other goals or requirements should be stated as constraints. The heuristics for identifying constraints are presented below.

HIC 1. Constraints can be identified by considering each statement and asking:

(a) Does this fragment impose some constraint on the goal(s)?

(b) Does this statement specify some requirement that must be met?

Given an answer of 'yes' to either of these two questions, restate as a constraint every statement that exemplifies or states a requirement which must be met to achieve some goal.

Example 5.22 Example 4.25 on page 110 illustrates a such a case where a statement specifies a requirement that must be met. In this example, the word *qualifies* is a key indicator of a condition that must be met.

HIC 2. Constraints can be identified by searching for temporal connectives (i.e., *during, before, after,* etc.). Restate statements that describe *when* some condition is true or *when* a goal can be completed as a constraint.

Example 5.23 Example 4.24 on page 110 illustrates a how the identification of the temporal connective *during* led to the identification of the constraint: Meeting room must be available during the meeting date/time.

HIC 3. Constraints can be identified by searching statements which place limits on the completion of a goal.

Example 5.24 Interview transcripts with stakeholder for the meeting scheduler system were analyzed to identify possible constraints. The constraint `At least 35 participants should be handled efficiently` places a limit or minimum threshold for the system. This constraint can eventually be mapped into a non-functional requirement for the system.

HIC 4. Since constraints may place a condition on the achievement of a goal, they should be restated as goal obstacles to allow for subsequent elaboration of the obstacle using scenario analysis. This enables the consideration of exception cases which the system is required to handle.

Example 5.25 A constraint places a condition on the achievement of a goal. For example, the constraint `Member must be able to ascertain if product was previously purchased` in the Electronic Commerce Web Server, discussed in Chapter 6.2 on page 193, places a condition on the achievement of the goal `AVOID duplicate purchase`.

A summary of the heuristics for constraint identification is provided in Table C.3 (Appendix C; page 246). The following section presents the heuristics for goal classification.

5.3 Goal Classification Heuristics

The goal classification heuristics in this section address the problems of determining the type of each goal. GBRAM differentiates among types of goals according to their target conditions, classifying goals as either *achievement*[*] or *maintenance goals*[†]. In general, it is best to classify a goal at the moment it is initially identified, since it is convenient and more efficient to examine each goal once, upon identification, instead of revisiting each goal at a later date for classification. Thus, analysts are encouraged to perform the classification activity concurrently with the goal identification activity. When analysts experience difficulty in classifying a goal as either an achievement or a maintenance goal, it may be because the goal is either an organizational/policy level goal or a quality goal (e.g. `SPEEDUP time required to process claim`). This section presents the heuristics for classifying goals as either achievement or maintenance goals.

Classifying Achievement Goals

Achievement goals generally map to actions that occur in the system. Thus, they are helpful in identifying functional requirements in the system. The heuristics for classifying goals as achievement goals are presented below.

HCA 1. Goals are classified as achievement goals by considering each goal and asking:

(a) Is completion of this goal self-contained?

[*] *Achievement goals* are satisfied when their target conditions are attained.

[†] *Maintenance goals* are those goals which are satisfied while their target condition remains constant or true.

(b) Does this goal denote a state that has been achieved or a desired state?

(c) Does the completion of this goal depend on the completion of another goal?

(d) Is the ability of another goal to complete depend upon the completion of this goal?

Given an answer of 'yes' to any of the questions above, classify the goal as an achievement goal.

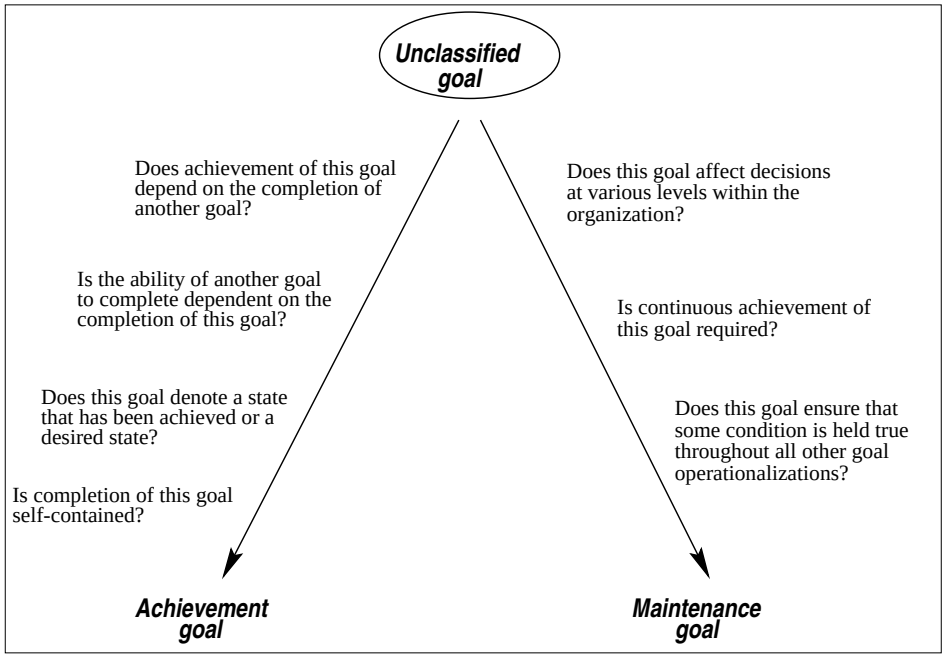

Figure 5.6. Classifying Goals

HCA 2. Achievement goals can be identified by searching for key words representing desired behaviors within the system (i.e., *make*, *improved*, *speed up*, *increase*, *satisfied*, *completed*, *allocated*, etc.)

Example 5.26 Recall that achievement goals are objectives of the system that are named by verbs such as *make*. Consider a seminar registration system that may need to satisfy the goal of enrolling organization members for a seminar before the actual seminar begins. The object of the goal is seminar registration, thus the goal would be named `MAKE member registered`.

HCA 3. Achievement goals are relatively self-contained. While other goals may depend on the completion of the given goal, achievement goals rarely impose constraints upon an entire class of goals (e.g. a group of security and access goals). In contrast, a Maintenance goal is likely to impose a constraint upon an entire class of achievement goals.

A summary of the heuristics for classifying achievement goals is provided in Table C.5 (Appendix C; page 248). The following subsection presents the heuristics for classifying maintenance goals.

Classifying Maintenance Goals

A maintenance goal is satisfied as long as its target condition remains true. Since maintenance goals suggest a continuous state within the system, they may generally be mapped to nonfunctional requirements*. Not all maintenance goals map to nonfunctional requirements, some generally map to safety requirements. The heuristics for classifying goals as Maintenance goals are presented below.

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

HCM 1. Goals are classified as maintenance goals by considering each identified goal and asking:

(a) Does this goal ensure that some condition is held true throughout all other goal operationalizations?

(b) Does this goal affect decisions at various levels within the organization?

(c) Is continuous achievement of this goal required?

Example 5.27 In the Career Track Training System (Chapter 3.2, page 45), the goal G_2 (`Tax payers money spent efficiently`) must be achieved on a 'continuous' basis. The system mandates that career tracks be provided in order to ensure that tax payers' money is spent efficiently. This goal characterizes a condition which must be held true.

HCM 2. Maintenance goals can be identified by searching for key words that suggest a continuous state within the system (i.e., *keep, ensure, avoid, know, monitor, track, provide, supply,* etc.).

Example 5.28 Example 4.1 on page 75 illustrates how the word *provided* allowed for the identification of two maintenance goals, one pertaining to the provision of training courses and another pertaining to the provision of career tracks.

Example 5.29 Example 4.2 on page 76 illustrates how the words *tracked* served as an indicator for the maintenance goal `Progress tracked`.

HCM 3. Maintenance goals tend to be operationalized as actions that prevent certain states from being reached within the system. Since maintenance goals are those goals which are sat-

isfied while their target condition remains true, they are named using the verbs `MAINTAIN`, `KEEP`, `AVOID` and `ENSURE`.

Example 5.30 The goals in Table 4.6 (page 87) are examples of maintenance goals.

A summary of the heuristics for classifying maintenance goals is provided in Table C.5 (Appendix C; page 248). The following subsection presents the GBRAM goal refinement heuristics.

5.4 Goal Refinement Heuristics

GBRAM refinement heuristics allow analysts to prune or refine the size of the goal set and to clarify stakeholders' goals. GBRAM provides three sets of refinement heuristics:

- heuristics for the elimination of redundancies;
- heuristics for the reconciliation of synonymous goals; and
- heuristics for the refinement of system-specific goals.

Figure 5.7 illustrates a likely sequence of goal refinement activities. In this control flow chart, the goals are refined after the goal set has been ordered according to precedence relations. When analysts have a small goal set to work with it may be convenient to apply the refinement heuristics before applying the dependency heuristics. However, for large goal sets it may be more convenient to apply the refinement heuristics after ordering the goals, so that the investment of time can be greatly reduced due to the clustering effect exhibited by ordered goals, as detailed in Chapter 4. This section presents the three sets of heuristics for refining the goal set.

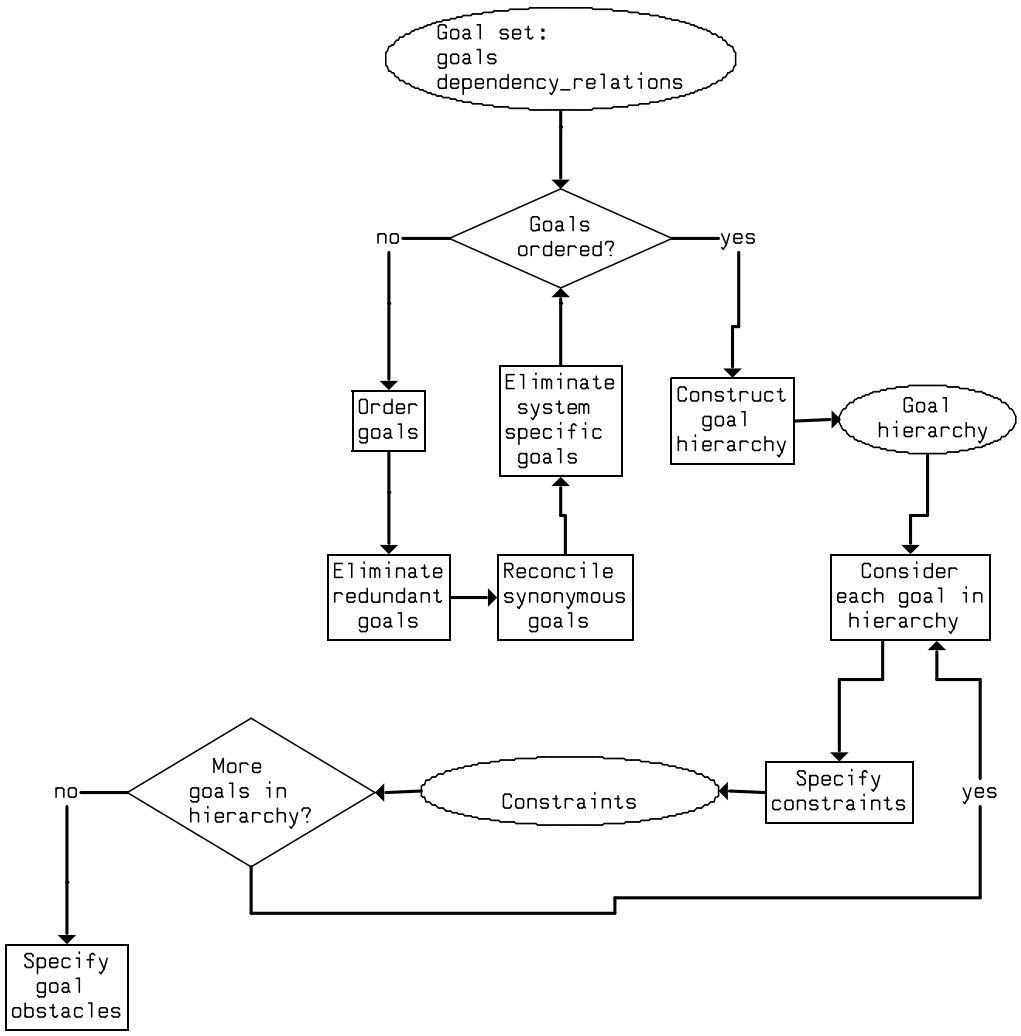

Figure 5.7. Control Flow Chart for Refining Goal Set

Heuristics for the Elimination of Redundancies

Analysts should examine each occurrence of redundant goals according to the following heuristics and resolution strategies.

HRR 1. If the same goal appears more than once AND the same agent is responsible for the goal on each occurrence, then all but one of the goals may be eliminated.

HRR 2. If the same goal appears more than once but two or more different agents are responsible for the same goal at different times, then all but one occurrence of the goal should be eliminated. However, to prevent the loss of information the analyst must keep track of all current and future agents who assume responsibility for the goal.

A summary of the heuristics for refining goals by eliminating redundancies is provided in Table C.6 (Appendix C; page 249). The following subsection presents the GBRAM heuristics for reconciling synonymous goals.

Heuristics for the Reconciliation of Synonymous Goals

The objective of the heuristics for the reconciliation of synonymous goals is to ensure that a common terminology is agreed upon by the stakeholders and to clarify, whether or not there is a shared understanding for goals which stakeholders may refer to by different names. Analysts can apply the inquiry cycle and the questions discussed in Section 5.1 to determine which goals are candidates for subsuming other synonymous goals. The heuristics for reconciling synonymous goals are presented below.

HRS 1. If two goals are synonymous, eliminate one of them.

Example 5.31 Example 4.6 on page 83 illustrates how the goal `Skills improved` can be semantically subsumed by the synonymous goal `Qualifications improved`.

HRS 2. If two synonymous goals are heterogeneous (e.g. one is an achievement goal and the other is a maintenance goal), then it is likely that the maintenance goal was classified incorrectly. If a maintenance goal is synonymous with an achievement goal, then the maintenance goal should be decomposed into an achievement goal. If the maintenance goal is decomposed into more than one achievement goal, then at least one of the goals should be synonymous with the originial achievement goals, and thus synonymous with the maintenance goal.

HRS 3. Consolidate and refine goals by merging synonymous goals.

Example 5.32 Example 4.7 on page 84 illustrates how the goals `Training coordinated` and `Training provided` in the Career Track Training System are synonymous and can be consolidated into one goal.

HRS 4. Since synonymous goals tend to share precedence relations, they appear clustered together when ordered. Ordering goals according to their precedence relations thus facilitates the identification of synonymous goals.

A summary of the heuristics for reconciling synonymous goals is provided in Table C.6 (Appendix C; page 249). The following subsection presents the GBRAM heuristics for refining system-specific goals.

Heuristics for the Refinement of System-Specific and Information Dissemination Goals

A goal-based model is an abstraction of *what* the desired system must do, not *how* it will be done. The goals in the model should be application-domain concepts and not computer implementation concepts, such as data structures. A good model may be readily understood by both programmers and application experts who are not programmers. The analysis model should not contain implementation decisions. Recall that GBRAM is a requirements analysis method, not an information modeling method; thus, GBRAM focuses on what the goals and requirements are and their attributes and operations rather than on specific data entities. For example, a workstation windowing system would be described in terms of attributes and operations visible to the user, not in terms of the actual implementation solution. The heuristics for refining system-specific goals are presented below.

HRSS 1. Individual information dissemination goals may be refined by asking:

(a) What is 'information' and why is it significant or important?

(b) Do any goals depend on the availability of this information for goal achievement?

Restate goals based on system-specific entities to capture the essence of the goal without including any system-specific information. If a goal is based on system-specific entities, it should be eliminated (e.g. `database updated` is a system-specific goal).

Example 5.33 Goals based on system entities should be refined. We initially identified the goal `HRD process completed` which clearly called for refinement due to the need to develop an understanding of what the 'HRD process' entails. By asking the

stakeholder *"What is 'HRD process,' and why is it significant?"* it was possible to define `HRD process` and determine that the goal (`HRD process completed`) can be decomposed into two goals: `Available course slots announced` and `Qualified personnel identified`. Thus, in this case, one goal was eliminated and replaced with two new goals.

When an implementation bias exists (i.e., if the customer has requested a certain implementation platform) then it may not be possible to ignore system-specific information.

Example 5.34 Due to the nature of the meeting scheduler, many of the goals in this system pertain to dissemination. For example, if the meeting scheduler were implemented as a messaging system in which meetings are scheduled via email messages, then the use of system-specific or information dissemination goals cannot be avoided.

HRSS 2. Restate routing goals to avoid emphasizing the receiving party and so that the underlying process and activity is represented.

Example 5.35 In the Career Track Training System (Section 3.2; page 45), the goal `Career portfolio routed to DTM` concerns the dissemination of some information. This goal is restated so that it reflects the underlying process: `Employee prefs made available`. In order to determine what the contents of a Career portfolio are to define DTM, the analyst must apply a few of the questions discussed in Section 5.1. Notice that all references to information disemination have been eliminated from the goals.

Example 5.36 Similarly, the goal `IP sent to TSD` is restated so that the goal reflects the underlying process: `IPs made available`. As illustrated in the previous example,

the questions discussed in Section 5.1 must be applied in order to clarify what IP and TSD are. By asking the stakeholder *"What is 'TSD info' and why is it significant?"* it is learned that `TSD info` and `directorate inputs` are the collection of IPs that have been submitted by employees. Each IP contains two lists: a list of all courses taken by an employee and a list of the additional training courses the employee wishes to take.

A summary of the heuristics for refining system-specific goals is provided in Table C.6 (Appendix C; page 249). The strength of GBRAM lies in its focus on goals and objectives and the derivation of operational requirements from those goals. The next section focuses on the heuristics which aid in the elaboration of goals.

5.5 Goal Elaboration Heuristics

Goal elaboration is the process of adding practical detail to the goals identifed from intially simplified descriptions. Analysts begin the goal elaboration process by considering the dependency relationships that exist between goals; this process facilitates the consideration of possible ways in which goals may be blocked or failed so that exceptional cases may be anticipated. There are three sets of GBRAM elaboration heuristics employed by analysts:

- heuristics for considering goal dependencies;
- heuristics to guide obstacle analysis; and
- heuristics to guide scenario analysis.

The objective of the goal dependency heuristics is to guide the analyst in considering goal relationships so that the goals may be ordered in the goal hierarchy. The goal obstacle heuristics allow analysts to consider the possible ways for goals to fail, facilitating the anticipation of exception cases. Scenario analysis heuristics allow analysts to evaluate changing goal priorities. Goals are further elaborated by considering the possible ways in which they may be blocked and by identifying scenarios to develop an understanding of how the goals may be operationalized. These three sets of goal elaboration heuristics are presented in the following subsections.

Heuristics for Considering Goal Dependencies

To gain a better understanding of the 'big picture,' it is helpful to model the relationships between goals. In Chapter 4 the identification of goal obstacles and scenarios to enable the consideration of such relationships was discussed. Recall that *refinement* refers to the process of pruning the goal set, while *elaboration* refers to the process of acquiring more detailed information about the goals and identifying or uncovering new goals for the goal set.

GBRAM recognizes three kinds of dependency relations among goals: *precedence dependency*[*], *contract dependency*[†], and *agent dependency*[‡]. Identification of these dependency relationships facilitates ordering the goals en route to constructing a goal hierarchy. Goal dependencies also assist analysts in the essential task of identifying pre- and post-

[*]A *precedence dependency* exists between goals G_1 and G_2 if goal G_1 must be completed before goal G_2.
[†]A *contract dependency* exists between goals G_1 and G_2 when goal G_2 must be achieved if goal G_1 occurs.
[‡]An *agent dependency* exists between goals G_1 and G_2 when the agent responsible for goal G_1 must complete the goal in order for the agent responsible for goal G_2 to complete G_2.

conditions during goal elaboration. As previously mentioned, the GBRAM goal dependency heuristics are considered to be both refinement and elaboration heuristics. Figure 5.7 on page 166 illustrates how goal refinement relies on the goal dependency heuristics depending on the analysts' chosen order of activities when applying GBRAM. A set of questions guides the identification of goal dependencies through this process. Figure 5.8 illustrates, at a high level, a possible sequence of application of these questions. Again, this flow chart is an approximation of the method. Each of the diamonds in this figure represents an inquiry point during which the analyst can apply the questions discussed in below in association with the heuristics as well as follow-up questions as suggested in Section 5.1. Goals must be organized according to their precedence relations for construction of the goal hierarchy. Specifying goal dependencies is thus a prerequisite activity for ordering the goals. The goal dependency heuristics are presented below.

HED 1. An effective way to discover precedence dependencies between goals is to consider each goal and ask:

(a) What goals are the prerequisites for this goal?

(b) What goal(s) must follow this goal?

The answers to these questions indicate the given goal's precedence relations, and should be documented by the analyst so that the goal may be subsequently ordered in accordance with these relationships. Table 5.3, found on page 175 and Chapter 4, 88 detail methods which analysts may employ for these annotations.

Example 5.37 Example 4.11 on page 89 illustrates how consideration of the precedence dependencies for the goal `Course completed` in the Career Track Training System

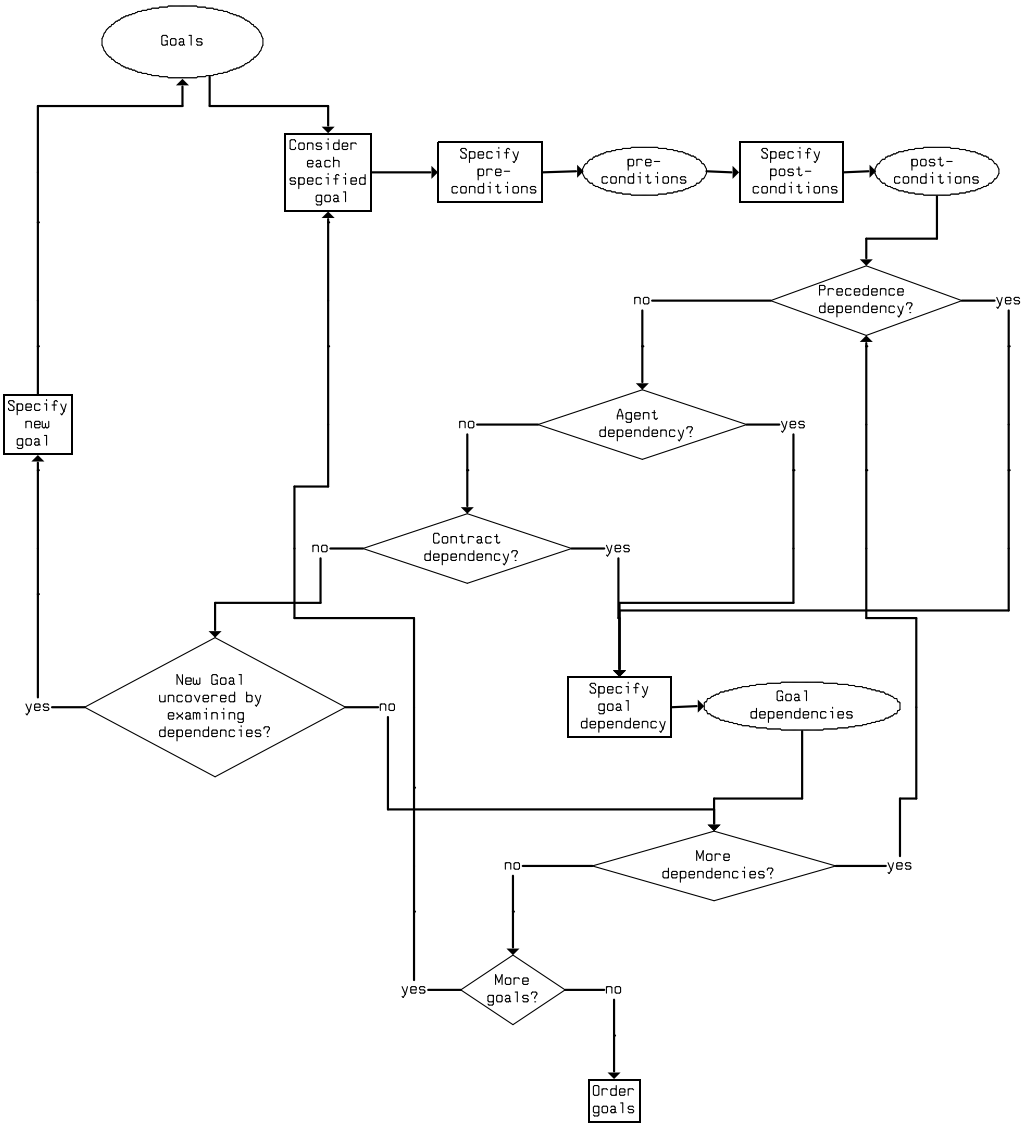

Figure 5.8. Control Flow Chart for Specifying Goal Dependencies

leads to the clarification of the relationship between this goal and the goals `Skills improved` and `Certification granted`.

Example 5.38 In the CTTS the prerequisites for each goal were considered. For example, before a career portfolio can be reviewed, or updated, it must be created. Table 5.3 shows the ordering that resulted from the analysis of the dependency relations for G_2 (`Career portfolio reviewed`). By asking *"What are the prerequisites for this goal?"* it becomes evident that $G_1 < G_2$.

HED 2. An effective way to identify contract dependencies is to consider each goal and ask:

(a) What goal(s) *must* be completed if this goal is achieved?

Example 5.39 Example 4.12 on page 89 illustrates how a contract relation may be observed between the goals `Course completed` and `Skills improved`. If the course is not completed, the employees skills are not improved and both goals fail; thus there exists a contract dependency between the two goals.

Table 5.3. Ordered Achievement Goals

Achievement Goals	Agent	Stakeholders
G_2 : Career portfolio reviewed	employee	employee, supervisor
G_3 : Career portfolio updated	employee	employee, supervisor
G_4 : Career portfolio made available	employee	employee, DTM
G_5 : Employee course prefs ready	employee	employee
G_6 : IPs made available	HRD	HRD, employee

Example 5.40 Example 4.12 on page 89 illustrates how contract dependencies exist between the goals `Course completed` and `Skills improved` in the Career Track Training System (Chapter 3.2; page 45).

HED 3. An effective way to identify agent dependencies between goals is to consider each goal and ask:

(a) What agent must complete the goal(s) they are responsible for before the agent responsible for this goal can achieve this goal?

Example 5.41 Consider a payroll system; before an employee can be paid, the employee's supervisor depends on the the employee to provide him with a record of the number of hours the employee worked (e.g. a time sheet).

Example 5.42 Consider goal G_2 in Table 5.3. From this goal, it is clear that an agent dependency between the supervisor and the employee exists. The supervisor depends on the employee to provide him the necessary information which enables the supervisor to decide whether or not to approve the employee's career portfolio course preferences. IP availability is similarly contingent upon this flow of information. Thus in Table 5.3, $G_3 < G_4 < G_5$.

The inquiry points for HED 2, HED 3, and HED 4 are represented by the diamonds in Figure 5.8; the questions discussed in Section 5.1 also assist the analyst in determining the goal relevant information pertaining to goal dependencies. A summary of the heuristics for considering goal dependencies is provided in Table C.7 (Appendix C; page 250). The goal

dependency heuristics presented in this section enable analysts to elaborate the goal set by determining the relationships between goals and agents so that goals may be ordered and refined. Additionally, by considering the goal dependencies in a methodical manner, it is likely that analysts will recognize the need for additional goals that had not been previously identified and which may be necessary for the given goal to be realized.

Once the goal dependencies are established, the process of identifying obstacles is facilitated because each goal that fails directly affects the goals which depend upon it. The obstacle and scenario analysis heuristics serve to guide analysts as they elaborate the goal set and uncover new goals and requirements. The following section focuses on the heuristics which aid in the elaboration of goals via the analysis of goal obstacles.

Heuristics to Guide Obstacle Analysis

The objective of the obstacle analysis heuristics is to uncover hidden goals and requirements by considering the possible ways in which goals can fail. Figure 5.9 illustrates, at a high level, the relationship between obstacle and scenario analysis. The possible ways that goals may be blocked are considered by assigning a trivial obstacle to each goal. The obstacle analysis process forces analysts to consider specific cases that must be handled due to activities which prevent goal completion.

HEO 1. There is at least one goal obstacle for every goal. This is informally referred to as the trivial obstacle and formally referred to as the normal first case goal obstacles. These obstacles are worded by negating the verb in the goal name.

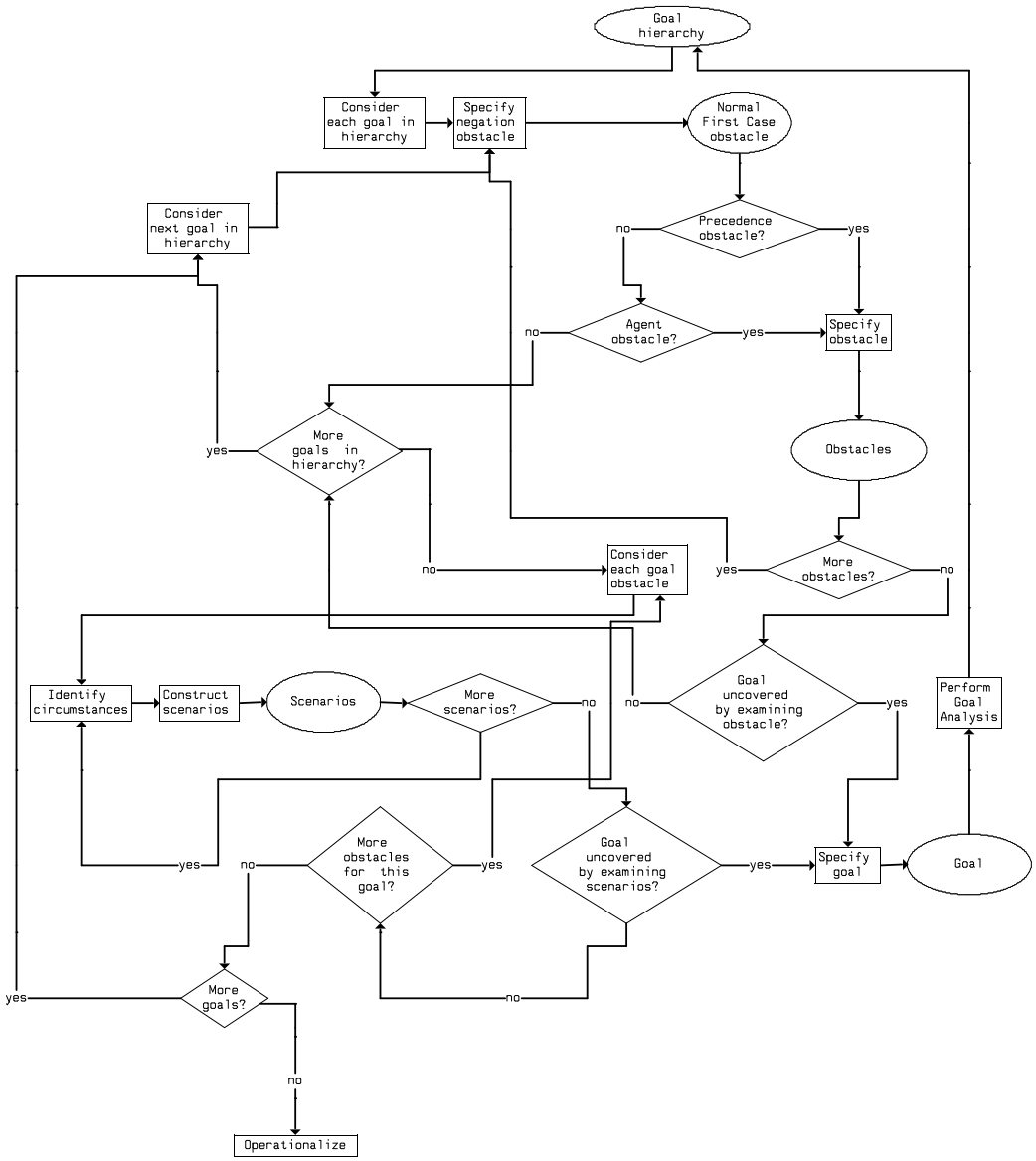

Figure 5.9. Control Flow Chart for Obstacle and Scenario Analysis

Example 5.43 Example 4.18 on page 101 illustrates how trivial obstacles enables analysts and stakeholders to consider specific cases that must be handled due to activities which prevent goal completion.

HEO 2. A statement that illustrates a condition which prevents the completion of a goal or which illustrates an example of a goal being blocked by another goal is indicative of an obstacle and should be expressed as an obstacle.

Example 5.44 Example 4.13 on page 97 illustrates how a statement which imposes a condition on an identified goal provides insights into potential goal obstacles.

HEO 3. An effective way to identify goal obstacles is to consider each goal and ask:

(a) What other goal(s) or condition(s) does this goal depend on?

(b) What other goal(s) must be completed or achieved in order for this goal to be achieved? (precondition)

(c) What goal(s) depend on this goal? (postcondition)

(d) What goal(s) must follow from this goal? (postcondition)

(e) Can the failure of another goal to complete cause this goal to be blocked?

(f) If this goal is blocked, what are the consequences?

The answer to the questions above should be worded to emphasize the state that is true, thereby denoting a goal obstacle.

Example 5.45 By asking *"What other goal(s) or condition(s) does this goal depend on?"* it was possible to determine that the goal `Career portfolio made available`

in the Career Track Training System is dependent upon whether or not the supervisor's concurrence is obtained. Thus, using the "state as true" naming convention, `Supervisor's concurrence not obtained` gives rise to the obstacle `Career portfolio not made available`.

HEO 4. A prerequisite failure obstacle occurs when a goal having a precedence relation is obstructed because the precedence goal fails. Prerequisite failures are identified by considering each goal and asking:

(a) What other goal(s) does this goal depend on?

Example 5.46 Example 4.15 on page 99 illustrates how, prerequisite obstacles for the goal Career Track Training System `Qualified personnel identified` may be determined by examining the goal's precedence relations (e.g. `Preferences made available`).

HEO 5. An agent failure obstacle occurs when a goal fails because the responsible agent fails to achieve the goal. Agent failures are identified by considering each goal and asking:

(a) Can the failure of an agent to fulfill their responsibilities cause this goal to fail?

Example 5.47 Example 4.16 on page 100 illustrates how an Air Force Base (AFB) employee failing a course is an agent failure which prevents the goal `Certification granted` from being completed in the Career Track Training System.

Example 5.48 Example 4.14 on page 98 illustrates how the failure of an agent to perform a goal he/she is responsible for can prevent another goal from being completed.

HEO 6. An contract failure obstacle occurs when a goal which holds a contract with another goal fails. Contract failure obstacles are identified by considering each goal and asking:

(a) Does this goal share a contractual relation with another goal?

Example 5.49 In example 4.17 on page 101, the goals `Course completed` and `Skills improved` in the Career Track Training System share a contractual relationship. The example illustrates how a contract failure may cause a contractual goal to fail (i.e., if `Course completed` is not achieved, the goal `Skills improved` is not achieved).

A summary of the heuristics for goal obstacle analysis is provided in Table C.7 (Appendix C; page 250). While each of these activities (obstacle and scenario analysis) were treated separately in Chapter 4, the activities are complementary. Obstacle identification begins to identify ways in which goals can fail; this information is elaborated further via scenarios. The following section focuses on the heuristics which aid the analyst in constructing and analyzing scenarios.

Heuristics to Guide Scenario Analysis

Scenario analysis heuristics assist analysts in uncovering hidden goals by considering both non-special cases in the system and the circumstances under which goals may fail within the system. Obstacles denote the *reason* why a goal failed. Scenarios denote concrete *circumstances*, for example, those under which a goal may fail. Some scenarios thus instantiate goal obstacles. This section presents the heuristics to guide scenario analysis.

HES 1. An effective way to identify candidate scenarios for construction is to consider each goal and ask:

(a) What happens if this goal is not achieved?

(b) What are the circumstances under which this obstacle can occur?

The identified scenarios are elaborated by listing the activities that must occur should the scenario actually take place. The scenarios may be represented either as a simple list of actions or as the list of actions accompanied by the agent responsible for each action.

Example 5.50 Consider the Bugs problem description in Appendix B. One of the possible goals to elaborate is: `Weight gained`. The goal is elaborated by constructing a scenario which lists the activities that must occur in order to achieve the goal:

```
1. Bug moves to new grid location
2. Bacteria available
3. Bug eats bacteria
4. Bug gains weight
```

Example 5.51 In the CommerceNet Web Server (Chapter 6; page 193), the scenarios were represented as a list of agents and actions as shown below in the Scenario entitled "Member Navigates Web Pages":

```
1. Member:    Logs in members-only web page
2. Member:    Visits personal "What's New" and the
              "search" Web page
3. Member:    Submits query about payment negotiation
              protocol
4. CN Server: Gets member's preference data
5. CN Server: Searches for related web pages based on
              the preference
```

```
 6. CN Server: Generates search results web page
 7. CN Server: Responds with search results web page
 8. Member:    Visits web page X from search results web
               page
 9. Member:    Logs out
10. CN Server: Records that web page X has been visited
               by member in preference data
```

HES 2. Another effective way to identify candidate scenarios is to consider each obstacle and ask:

(a) Why did this obstacle occur?

(b) Why was this goal not achieved?

(c) Under what circumstances would this obstacle occur?

Example 5.52 Example 4.20 on page 105 illustrates how asking the questions above, with respect to the obstacle `No slots available`, led to the identification of two scenarios: `2.a All courses closed (max capacity reached)` and `2.b Course cancelled (no slots available)`.

HES 3. The scenarios which analysts should provide particular or special attention to are those which violate goals or obstacles. Scenarios should be analyzed by considering the possible ways in which goal obstacles could be prevented. This process leads to the identification of new goals and requirements for the system.

5.6 Summary

This chapter presented four sets of GBRAM heuristics and a set of recurring question types which aid analysts in applying an inquiry-driven approach. The GBRAM heuristics detailed in this chapter were:

- identification heuristics;
- goal classification heuristics;
- refinement heuristics; and
- elaboration heuristics.

The identification heuristics assist analysts in identifying goals, stakeholders, responsible agents, and constraints. Goal classification heuristics aid analysts in determining whether a goal is an achievement or maintenance goal. Refinement heuristics assist analysts in pruning the size of the goal set by eliminating redundant goals and reconciling synonymous goals. GBRAM elaboration heuristics address the need to consider goal dependencies, suggesting the goal obstacles for which scenarios should be constructed and which scenarios to elaborate. A summary of four heuristics sets is provided in Appendix C, beginning on page 243.

The following chapter discusses validation of the Goal-Based Requirements Analysis Method via its application to a large industrial case involving the reengineering of an electronic commerce Web server and through utilization of the method in an empirical investigation.

CHAPTER 6

Validation

Practice proves more than theory, in any case.

Abraham Lincoln

While it is easy to propose a method, validation is required before the usefulness of the method may be determined. Software methods, such as the Goal-Based Requirements Analysis Method, need early validation while under development. The initial case studies which seek this validation, discussed in Chapter 3, are best characterized as the formative case studies due to their central role in shaping the method. In contrast, the case studies discussed in this chapter are best characterized as summative; their primary role in this thesis is to validate the method developed during the formative case studies.

Validation of the Goal-Based Requirements Analysis Method involved three applications of the method:

- use of the GBRAM to specify the requirements for tool support, followed by construction of a prototype based on the resulting requirements;
- a case study involving a large commercial application and multiple stakeholders to evaluate the scalability of the method; and
- an empirical evaluation whereby use of GBRAM by one group of subjects was compared to the use of alternative analysis methods by other subjects.

This chapter discusses the Goal-Based Requirements Analysis Method in the context of these three efforts. While theoretically different in research approach taken, each of these efforts seeks to validate the method. The GBRAM was used to specify the requirements for a tool to support the method*; consequently, a prototype, referred to as the Goal-Based Requirements Analysis Tool (GBRAT), described in Chapter 4.5, was developed. An overview of this validation effort is provided in Section 6.1. In order to investigate the scalability of the method, GBRAM was applied to a large commercial intranet web server application†, discussed in Section 6.2. A synopsis of an empirical evaluation performed to validate the Goal-Based Requirements Analysis Method is presented in Section 6.3‡. Table 6.1 summarizes the data availability for each of the validation studies discussed in this chapter.

Table 6.1. Availability of Case Study Data

Case Study	Raw Data	Requirements Document
GBRAT	√	√
Electronic Commerce WWW Server	**	**
Vacation/Sick Leave System	√	√
Key √ Available from author upon request ** Company Confidential		

*The data is available on the WWW at http://www.cc.gatech.edu/computing/SW_Eng/GBRAT/case.ps.

†This case study was sponsored by NTT Software Labs in Palo Alto, California, and is a company confidential project.

‡Available from the author upon request.

6.1 The Goal-Based Requirements Analysis Tool

The first summative effort to formally validate the efficacy of the Goal-Based Requirements Analysis Method involved using the GBRAM to identify the functional modules necessary for a tool to support the method. The objective of the Goal-Based Requirements Analysis Tool (GBRAT) is to provide analysts with the procedural support needed for the analysis and refinement of goals for software-based information systems, supporting and guiding analysts as they identify, capture, and structure requirements information in the form of goals.

Methodology and Case Study Artifacts

The Goal-Based Requirements Analysis Tool (GBRAT) case study was conducted for approximately 20 hours a week over a period of one month. One analyst was assigned to review the available documentation: a four-page description of the goal-based method (106 lines of text) and a one-page research abstract (20 lines of text). Both sources of information were analyzed using the GBRAM to identify system goals and translate those goals into operational requirements. Subsequently, a design document* was produced and a prototype† of the tool was built.

Goals were documented for the GBRAT using tables created in LaTeX for document preparation. The GBRAT requirements document was composed in Hyper Text Markup Language (HTML), which provided the ability to cross-reference between auxiliary notes

*http://www.cc.gatech.edu/computing/SW_Eng/GBRAT/design_doc.html
†http://www.cc.gatech.edu/computing/SW_Eng/GBRAT/

and requirements throughout the document. The hypertext links thus afforded a general level of requirements traceability. Since the subsequent design document was also written using HTML, traceability was ensured among the various document artifacts which led to the production of the final system.

Lessons Learned

This section summarizes the lessons learned from the GBRAT case study. In Chapter 3, the presentation of each case study includes a discussion of the lessons learned. The discussion of lessons learned in this chapter takes a different bent; the lessons learned during the summative case studies outlined in this chapter are much more validation oriented than development oriented, whereas the case studies detailed in Chapter 3 are primarily formative with a concurrent bent towards validation. Thus, the following discussion seeks to provide examples which confirm and augment the method presented in Chapter 3 and the heuristics presented in Chapter 5.

Redundant goals may be identified by looking for synonyms

During this case study, heuristic HRS 1. (page 168) was solidly confirmed by its application in numerous instances. For example, three synonymous maintenance goals were extracted from the research abstract: `strategies provided`, `techniques provided`, and `guidelines provided`. The three words *strategies*, *techniques*, and *guidelines* may imply different meanings. However, in the context of GBRAT, the three terms referred to the tool's provision of prescriptive advice. Since the word *strategies* encompassed the stakeholders'

intentions and was the most general of the three terms, and since one agent, GBRAT, was responsible for the three goals, these goals could be merged into one goal. This confluence is illustrated in HRS 1 (page 168): `strategies provided`.

The system is usually responsible for maintenance goals

Heuristic HCM 2. (page 164) suggests keywords which signal a continuous state within the system and indicate candidate maintenance goals. In this case study, the keywords 'provision of' and 'provide' facilitated the identification of several maintenance goals from the research abstract. This is of particular note in that the words 'provision of' and 'provide' in the research abstract directly correspond to the system as the responsible agent in this analysis. This suggests that maintenance goals may also be identified by considering who or what is the responsible agent. However, some goals had both the system and the user as shared agents; these goals with non-human and human agents were typically achievement goals. The following section further discusses the lessons learned from this occurrence.

Goals with two responsible agents may require two operational definitions

Heuristic HIA 3. (page 158) indicates that multiple agents may share responsibility for a goal. In the GBRAT case study, it was observed that viewpoints may be expressed via the operational definitions in the requirements document. Consider the GBRAT goal `Goals organized`. Two agents, the analyst, or user, and the system itself share responsibility for this goal, but the actions required of each agent for the organization of goals are different. Depending on the implementation, the analyst may need to specify formal dependency relations so that goals may be ordered, or use a drag and drop tool to order the goals (as

supported by the GBRAT goal hierarchy tool, discussed in Chapter 4.5). Alternatively, the tool may either parse some dependency rules or invoke sorting algorithms to order the goals based on the analyst's input. To address the different roles played by agents sharing responsibility for goals, two operational definitions were constructed and specified in the requirements document. Since there were only two agents in GBRAT, the user and the system, each GBRAT functional requirement has a *user-operational definition* and a *tool-operational definition*. Example 6.1 discusses the two operational definitions for the GBRAT goal `Specify agent`.

Example 6.1 (GBRAT functional requirement: Specify Agents) This goal requires a division of labor due to the shared responsibility between the user and the tool for its completion. The user must specify the agent responsible for the completion of each goal. The system must make the goals selectable to the user and display any previously specified agents associated with the selectable goals. The user, in turn, must select a goal and determine the responsible agent(s) for the selected goal. Thus, the user is responsible for selecting goals, determining the responsible agent(s), and specifying those agents. The tool is responsible for presenting the goals and agents, making them selectable, and then storing the responsibilities which the user specified for that goal. This separation of responsibilities in the operational definitions makes the role of each agent much more clear for the design and implementation of the system.

Inquiry surfaces implementation issues and alternatives

Several questions pertaining to implementation-specific issues were raised during the GBRAT case study. One of these issues addressed the need for a flag to indicate unresolved issues that require discussion among the analysis team and/or stakeholders. If there are several conflicting issues related to a specific goal, these issues must be managed by the tool. Possible alternatives were identified through an inquiry-driven approach. For example, one alternative posed during the investigation suggests that if one person has the authority to make the final modification based on a set of assumptions, a configuration management flag may not be needed since only one person/user may make modifications to the project repository. However, if multiple users will share responsibility for a project repository, the implementation must support auxiliary notes and provide a mechanism for both tracking revisions and unresolved issues. Due to the design and implementation ramifications, both of these options require decisions; the analyst must discuss the actual goals with the stakeholders since the decision will ultimately affect the systems' implementation.

Constraints point to questions which should be directed to the stakeholders

Constraints extracted from the method overview pointed to follow-up questions which the analysts should ask the stakeholder. Consider the constraint: `Goals are identified from two sources: textual statements of need and descriptions or models of current processes`. During the analysis of this GBRAT constraint, the goal was annotated with two questions: *Are you developing a completely new system?* and *Is a current system in place but in need of repair?* These questions may be generalized since they are not specific to one application. The answers to these questions will help the analyst determine

if there are, in fact, other sources of information available which the stakeholder(s) may have overlooked.

Discussion

Goal schemas were constructed for 45 GBRAT goals. However, the requirements document focused solely on the Goal Editor Module; thus, only the goal editor requirements and schemas appear in the requirements document. Nineteen functional requirements, 14 user operational definitions, and 14 tool operational definitions were identified for the Goal Editor module.

Requirements produced using the GBRAM enable effective development of a software system. The efficacy of a software method is best tested in practice; examination of the resulting artifact assist analysts in determining effectiveness and compliancy with the stated objectives. The requirements document produced using the GBRAM enabled the analyst to determine functional modules for the proposed system and translated the goals identified for these modules into operational requirements for a Goal Editor. Another analyst, the developer, was able to use the requirements document as the basis for the production of the GBRAT design document and for the construction of the GBRAT prototype. Thus, the GBRAM facilitated the development of the proposed system which was specified by employing the GBRAM.

While the requirements specified focused on *what* the system must do, the operational definitions which support the requirements favored *how* the system could work (e.g. design elements). While this facilitated the design phase, the analyst experienced the lack of

support GBRAM offers for information modeling. Since GBRAT is an information-intensive system, the ability to better model information requirements would have been beneficial to the study. Additionally, the lack of high-level organizational goals in this analysis was a limiting factor in this study; thus, while organizational issues and high level objectives played a role in the analysis of the FSO and CTTS studies, the GBRAT case study lacked the ability to develop more non-functional and organizational requirements.

The following case study was conducted due to the need for validation of the method's ability to handle high-level enterprise goals and the method's scalability for systems supporting a large organization comprised of multiple users holding conflicting goals.

6.2 CommerceNet Web Server

The objective of this case study was to further validate the GBRAM by broadening the base of practical experience through application of the method to an Intranet electronic commerce application. Application of the method to a large commercial application provided insights into how goals are used to identify and refine system requirements as well as the applicability of the method's strategies in reengineering efforts involving teams of analysts.

The electronic commerce World Wide Web (WWW) server case study involved the reengineering of a Web server which supports various consortium member organizations participating in electronic commerce over the Internet. The Web server, hereafter referred to as CommerceNet, must support secure payment and transactions, different access levels, membership and seminar registrations, and project and proposal status tracking. The

CommerceNet study also explores the issues which arise when applying the GBRAM in a collaborative effort. This section summarizes the knowledge garnered from using the GBRAM in the specification of requirements for a commercial web-based application.

Methodology and Case Study Artifacts

The CommerceNet case study was conducted by four analysts, the author, and various stakeholders, for approximately 30 hours a week over a period of four months. The author of this thesis served as the principle analyst conducting weekly video conference meetings over the MBone* with a group of three to four stakeholders in the EColabor room at NTT Multimedia Communications Laboratories in Palo Alto, California. These meetings were each one to three hours in duration. Some additional meetings were conducted with larger groups of stakeholders (CommerceNet consortium members). The meetings were primarily goal elicitation sessions during which the goals of the system were fleshed out and specified collaboratively using tools such as vic, vat, and whiteboard.

The principle analyst produced a requirements document based on the elicited goals; the document was comprised of six major sections pertaining to the six functional areas within the CommerceNet server. Each of these sections contains four subsections: Goals, Functional Requirements, Nonfunctional Requirements, and Organizational Requirements. The goals are the objectives of the system. The functional requirements are the behavioral descriptions of the system and thus define what the system does. The nonfunctional requirements define the attributes of a system, such as the levels of efficiency, reliability, se-

*MBone stands for the Multicast Backbone on the Internet and provides audio and video connectivity.

curity, maintainability, portability, visibility, capacity, and standards compliance. Although reporting procedures do not belong in a requirements document, they were included in the CommerceNet requirements document in the form of organizational requirements due to their affect on the continued analysis of the system. The main analyst requested elaborated scenarios from the CommerceNet analysts and stakeholders for those goals which were not readily understood. Goals were elaborated on a scenario-by-scenario basis.

The CommerceNet Web Server requirements document was made available to stakeholders via the WWW using HyperMail. In contrast to the GBRAT requirements document which utilized hypertext to support requirements traceability, the CommerceNet requirements document employed hypertext links to support requirements discussions and to capture auxiliary notations. The use of auxiliary notations is discussed below within the context of lessons learned in the CommerceNet case study.

Lessons Learned

This section summarizes the lessons learned from the electronic commerce web server case study.

Stakeholders who are well-trained analysts may express goals in terms of functions

Heuristic HIG 9. (page 149) discusses stakeholders' tendencies to express requirements in terms of operations and actions; during the CommerceNet case study, stakeholders expressed requirements in terms of functions that the system had to perform. During the first video conference meeting, which lasted for 1 hour and 45 minutes, four participants

(two analysts from Georgia Tech and two analysts representing CommerceNet) met for a brainstorming session. During this session the CommerceNet representatives expressed their goals for the system in terms of the functions the system must support; after some initial brainstorming, the functions were categorized into four broad categories (shown in Table 6.2). The function categories ultimately served as the four CommerceNet goal classes previously discussed in Chapter 3. Since functions correspond to actions or behaviors within a system, they were easily expressed as goals by the primary analyst at a later date.

Table 6.2. CommerceNet Function Categories

Function Category	Number of Functions Identified
Information Display & Organization	14
Process Support	11
Usage Levels (public or member)	9
Electronic Commerce	4

Several factors contributed to stakeholders' expression of system goals in terms of functions. The objective of the CommerceNet Web Server project reflected in the terminology used by stakeholders was the reorganization and redesign of an existing system. For the CommerceNet example, stakeholders were asked *how-to* questions as the requirements evolved. One of these questions was *"How is the public key entered on the application form?"*; the stakeholder responded *"Public keys are loaded in."* This resulted in a new goal `MAKE public key loaded in`. The use of computerese by the stakeholders was quite common in this case study because the stakeholders assumed the use of a medium/mechanism for the goals with which they were well versed and experienced. Had such a goal arisen in,

for example, the FSO, the analyst would have needed to abstract away from said terminology to ensure that the goal vocabulary employed was easily understood by all stakeholders. In another example of a *how-to* question which surfaced the need for new goals within the system, stakeholders expressed that membership kits are sent to CommerceNet users. The follow-up question asked by the analyst was *"How is the membership kit sent to the user?"* the stakeholders responded "Via email."

Stakeholders almost always refer to processes they want operationalized in the system as "automatic." In the CommerceNet case study, stakeholders identified several goals needing to be performed "automatically" by the system:

- `Web server history changed automatically`;
- `Web resources automatically rated by key word occurrences, accesses, and what's hot`; and
- `What's New automatically generated`.

Such goals imply "knowledge" on the part of the system. It was apparent throughout this project that the CommerceNet Web server was the responsible agent for all "automatic" goals as well as all "know" goals. "Know" goals pertain to states or conditions which the system must be able to distinguish or recognize (e.g. `KNOW user edit-authorization level`). This technique for the identification of automatic and know goals' responsible agents may simplify analysts' identification of those goals for which the system is responsible in other analysis efforts.

Stakeholders may be hesitant to focus on agents

During this study, participating stakeholders were initially hesitant to focus on agents due to their belief that the responsible agents would change as a result of the redesign and reengineering of the CommerceNet Web server. Of particular note, is the proximity of the specific/detailed goals to the operations which will be operationalized in the CommerceNet system; these goals will be performed by people, CommerceNet, or the server/system. As with most reengineering efforts, the primary focus is on the process(es), not on the people, since the employees involved with the system may change as a result of the newly designed process. However, since there were subtle policy issues which were only understood by the persons currently responsible for functions and goals, it was important to identify those persons in the event that follow-up interviews were needed later on. Also, if the newly designed system will alter stakeholders' work, it is important to track the people who will ultimately be affected.

Goals are named using a verb as the first word

Heuristic HIG 1. (page 144) states that goals may be named in a standard subset of natural language in which the first word is a verb describing the kind of goal being named. This naming convention, adopted during the CommerceNet study, facilitates the ordering of goals based on the knowledge that certain kinds of goals may have a pervasive affect on other goals. In this case study, all CommerceNet goals begin with verbs formed from the following set of words: `AVOID`, `ENSURE`, `IMPROVE`, `INCREASE`, `KEEP`, `KNOW`, `MAINTAIN`, `MAKE`, and `SPEEDUP`. Table 6.3 provides an example of each of these words which describe the kind of goal being named. `AVOID` implies a state which must be prevented within

the system. `ENSURE` refers to making certain that a particular state is achieved. `IMPROVE` is a 'quality' goal (discussed in Chapter 5, page 161) and which implies bettering the quality of some portion of the system and/or organization and increasing some level of productivity. `INCREASE` goals concern the amount or rate by which something is increased or made greater. `KEEP` implies the continuation of some state or event within the system at a steady level or pace, and may refer to saving information or maintaining some state within the system. `KNOW` implies the ability to distinguish or recognize some state within the system. `MAINTAIN` implies the provision for or sustainment of some existing condition. `MAKE` implies the formation or attainment of some state within the system. `SPEEDUP` implies the acceleration of production.

Table 6.3. Examples of CommerceNet Goal Verbs

Goals	Goal Type
AVOID obsolete information	Maintenance
ENSURE secure transactions	Maintenance
IMPROVE content maintenance and administration	Improvement
INCREASE profits from seminars	Improvement
KEEP soliciting participation	Maintenance
KNOW member access privileges	Achievement
MAINTAIN two servers	Maintenance
MAKE member registered	Achievement
SPEEDUP time required to procure approval for modifications/updates	Improvement

Maintenance and achievement goals need to be differentiated

In the CommerceNet server, maintenance goals are basically high-level goals; the associated achievement goals should comply with the maintenance goals. It was important

to differentiate between maintenance and achievement goals since a majority of the discussions which transpired with the stakeholders addressed "maintaining the server" and "content maintenance"; while the conversational references regarding server maintenance goals and meta-level goals were clear to the analyst, the were not clear to the participating stakeholders.

It is possible that an achievement goal may share a relation with more than one maintenance goal. Consider the CommerceNet achievement goal: `What's New filtered according to member's personal preferences`. This goal shares a relation with two maintenance goals: `Personal preferences managed` and `User-level privileges enforced`. Thus, while it is beneficial to differentiate between achievement and maintenance goals, it is also important to consider the relationships which may exist among the goals.

Auxiliary notes help analysts track unresolved issues & requirements discussions

Within each subsection of the CommerceNet requirements document, various classifications of auxiliary notes appear, serving to document requirements discussions [61, 64]. *Questions* are reminders of unresolved issues pertaining to a particular requirement. *Answers* describe solutions or provide a clearer understanding of the requirements; when a question generates more than one answer, the answers are listed as *Alternatives*. *Reasons* provide justification for answers or requirements which are not immediately obvious. *Scenarios* serve to document issues and elaborate the requirements. At times a requirement or auxiliary note is followed by a parenthetical reference; some requirements address several functional areas and meet several goals, while the parenthetical cross-references imply that the same or substantive similar requirements are found elsewhere in the document. An

annotated requirements document allows analysts to list unresolved questions. The document is interspersed with unresolved issues and questions which serve as reminders for the analyst. Items requiring an answer or decision are explicitly flagged to expedite resolutions.

Discussion items assist in the development of requirements

An overriding objective of this thesis is to make the GBRAM and the inquiry process useful for many people by discussing the kinds of goals identified, the kinds of discussion items used, and how the inquiry cycle facilitated the process of formulating a system. This thesis does not attempt to impose unusable descriptive methods on analysts. Throughout the CommerceNet study, memos pointed to some content and subsequent actions (i.e., construct a scenario, answer a question, or explore a constraint). Each of these may appear in a scenario; by capturing and tracking memos, the analyst is essentially constructing a "memory" for the project. Another kind of memo used in the CommerceNet study is constraints*. During the early stages of analysis, constraints assume the form facts about how the system (CommerceNet) works and its possible relationship with other systems. Writing memos during the analysis process may be likened to incorporating a procedure into a system which reminds analysts to go back, check, and review.

Scenarios point to implementation alternatives

In the CommerceNet study, scenarios surfaced alternative implementation options which require a decision on the part of either the stakeholders or the analyst. For example, one scenario prompted the analyst to ask the question *"How are queries to be submitted?"*.

*Constraints are considered memos at the early stages of analysis because they have not yet been refined.

Stakeholders responded, "Via FORMS, but email is desirable as well; however, FORMS is a priority." This led to two possible scenarios: a query submitted via FORMS and a query submitted via email. Given these specifications, the system must eventually be able to handle these different negotiation protocols. If the system is driven by email protocols, the recipient is required to become a data entry user; this need for the recipient to enter data upon receipt of email would be by-passed by a FORMS implementation. Scenarios enable analysts to identify alternatives and consider the corresponding behaviors which the system must exhibit. In addition to surfacing implementation alternatives, scenarios point to policies which affect other goals; this was especially apparent when identifying policy-oriented scenarios affecting other goals for the goal `MAKE member registered`. For example, the scenario "Only sponsor members can vote" affects the goal `ENSURE voting supported`, which is a process support goal. Thus, the voting goal was elaborated with this scenario.

Scenarios facilitate the identification of new goals

Scenarios were used extensively in the CommerceNet case study. There are numerous examples of goals which were identified via scenario analysis. This form of identifying goals has been thoroughly discussed in this dissertation. Given the initial set of CommerceNet functions, the principle analyst identified several which were a bit vague, requiring clarification. The stakeholders were thus asked to elaborate the following five scenarios:

- Processing membership fees;
- Purchasing seminar video;
- Approving proposal;
- Finding information; and

- Making proposal.

The stakeholders elaborated each of the scenarios by listing the different activities for which agents are responsible. The first scenario "Processing membership fees," was comprised of the following 12 actions:

```
 1. User:              finds the membership application form page
 2. User:              fills out the membership application form
 3. User:              selects e-check as payment method
 4. User:              types in public key
 5. User:              submits the membership application form
 6. Certification      approves user payment
    Authority
 7. CN Server          responds to user with receipt
 8. CN Server          increases budget balance
 9. CN Server          creates user's entry in member database
10. CN Server          adds user to member mailing list
11. CN Server          adds user to member web page
12. CN Server          sends user membership kit
```

These actions were then stated using the naming conventions suggested by HIG 1. (page 144) and HIG 2. (page 144). Further analysis using the inquiry process suggested in the GBRAM's goal identification heuristics resulted in the identification of 12 goals. For example, consider the goal `MAKE payment method selected` in Table 6.4 which shows the obstacles and scenarios which correspond to this goal.

Table 6.4. CommerceNet Goal: MAKE payment method selected

Goal	Obstacles	Scenarios
MAKE payment method selected	1. Payment method not selected 2. Payment methods not clear	1. User selects e-check as payment method 2. George isn't sure if Burdell & Assoc. has an account set up yet & needs to know how to get one

Obstacles #1 and 2 indicate the users' need to select from various payment options, such as check, money order, or credit card. Additional goals were identified through the consideration of possible scenarios. For example, consider Scenario #2. George is an employee at Burdell & Associates. Before he selects a payment method, he must access his firm's CommerceNet Membership Web page to obtain the information he needs to select his firm's preferred payment method. This "walk through" approach was employed throughout the CommerceNet study and was helpful in the identification of goals.

When actions appear as instantiations (e.g. e-check vs. payment method), the goal is named in general terms (e.g. payment method) and the action is listed as a scenario. Having concrete scenarios aids analysts and stakeholders in considering other possible concrete scenarios. For example, given e-check as a possible payment method, other stakeholders are prompted to express their preferred payment method (e.g. credit card).

Constraints are often preconditions which may be expressed as goals

During the CommerceNet case study, conditions which could prevent a goal from being achieved were initially expressed as constraints. However, during goal elaboration, it became apparent that some of these constraints are actually preconditions. For example, consider goals G_{54} and G_{55} in Table 6.5. Both of their respective constraints are actually preconditions; they represent or illustrate states that must be achieved before the goal can be completed. Therefore, these constraints served as pointers to new goals for the system. In contrast, consider the constraint for goal G_{56}. This constraint restricts achievement of the goal by limiting the ability to purchase a videotape to a selected population of authorized persons. Thus, the constraint associated with G_{56} remains a constraint in the CommerceNet

Server goal set. Heuristic HIC 2. (page 159) suggests searching for temporal connectives to identify constraints. In CommerceNet, it was observed that `Register for seminar before deadline` indicates a constraints on the goal `MAKE member registered`.

Table 6.5. Constraints, Preconditions, and Goals

Goals	Constraints
G_{54} : MAKE on-line statement available for each organization	Member must be authorized to access on-line statement
G_{55} : AVOID duplicate purchases	Member must be able to ascertain whether his organization previously purchased the desired item
G_{56} : MAKE video tapes purchased	Only CN members & seminar participants may purchase seminar video tape

Goal Refinement

Goal refinement concerns the way goals change from the moment they are first identified to the moment they are operationalized in a system specification. Figure 6.1 shows the evolution of the CommerceNet goals into operational requirements. As shown in the figure, 18 high level goals were initially elicited from the stakeholders during the meetings conducted over the MBone. Subsequently, 54 goals were derived from the initial set of 18 by applying the GBRAM. The ovals in the figure represent the elaboration phase in which the inquiry cycle [61,64] was employed. The resulting requirements document for the electronic commerce web server contained a total of 79 requirements. The oval on the left side of the figure illustrates the number of questions, scenarios, alternatives, and answers generated which led to the specification of 52 functional requirements. The oval on the right side of

the figure corresponds to the inquiry process which led to the specification of 27 nonfunctional requirements. As illustrated in the figure, the inquiry cycle led to the specification of nearly twice as many functional requirements as nonfunctional requirements. This is due mainly to the techniques employed in GBRAM, their strength lies in forcing analysts to systematically consider the behavioral aspects of systems by focusing on obstacles and scenarios.

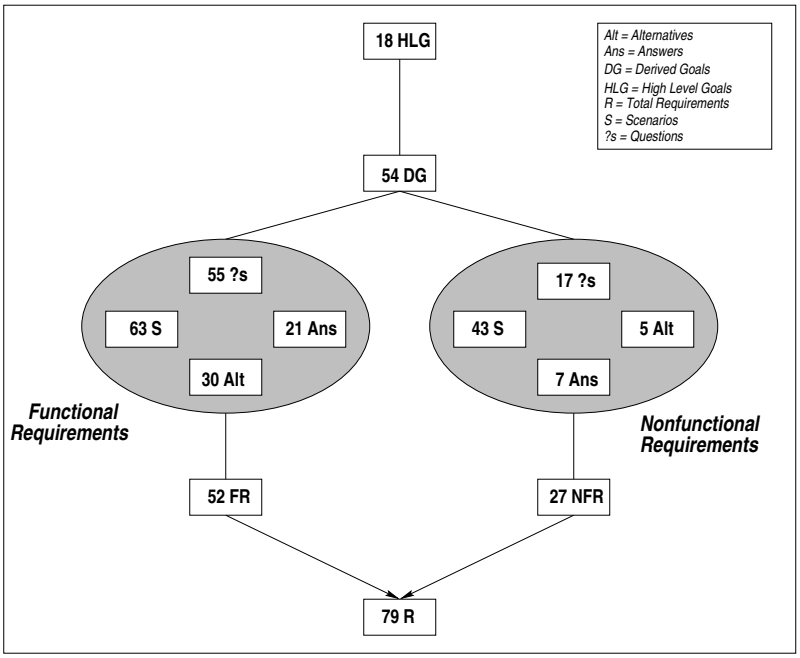

Figure 6.1. Evolution of Electronic Commerce Requirements

The CommerceNet requirements document was not traditional in that it was not composed solely of requirements. Instead, each requirement in the document was annotated with the relevant questions, answers, alternatives, and scenarios that arose during application of

the inquiry-driven approach. Moreover, scenarios pertaining to processes and issues that were nebulous or not well understood were elaborated by specifying the possible sequences of actions for that scenario. This was beneficial in that the requirements document became a "living" document which could be easily annotated by the stakeholders. Any item (i.e., a requirement or a scenario) may be the target of several annotations. While annotation support via HyperMail in this case study was certainly helpful for tracking the discussion elements, paper is adequate if annotation mechanismsare not available. The critical issue is tracking the rationale associated with specific decisions and flagging any unresolved issues so that they may be discussed among analysts and stakeholders and targeted for resolution.

Summary and Discussion

The CommerceNet Web server is used by a global consortium which consists of many international companies. The primary objective of CommerceNet is to change the way in which customers, service providers, and developers participate in business transactions and to facilitate interactions and collaborations between these parties. Users are provided access to all CommerceNet information and applications via the WWW.

In the CommerceNet case study, goal tables supported the actual production of requirements. At the early stages of analysis, scenarios provided glimpses of current processes and future iterations. The goal tables initially identified scenarios primarily as notes to the analyst indicating possible scenarios. For example, the goal `Different entrance to server supported for each user level` has two subgoals: `Public entrance to server supported` and `Member entrance to server supported`. The corresponding scenario pro-

vided by one of the stakeholders was: "As a Working Group chair, Kenji has access to more content," which implied that there were issues involving access to information. By constructing this scenario, it was possible to reason about what constrains this behavior and, as analysis ensued, the scenarios which were elaborated were those for which the analyst had not yet developed a thorough understanding. The goals, obstacles, scenarios, and associated annotations were captured in tables for the analysis by utilizing both Microsoft Excel spreadsheets and GBRAT. The initial brainstorming results were analyzed using GBRAT, while subsequent analyses employed Excel due to the ability to expedite ordering goals, formatting and printing the goal tables, and scrolling the tables.

The CommerceNet project involved reengineering an existing Web server. The stakeholders were experienced with the existing system and were also experienced analysts. This introduced new challenges in applying the GBRAM. Experienced analysts and stakeholders were eager to jump to discussions surrounding functions; thus it was important to ensure that everyone objectively analyzed the relevant goals. This was used as a form of early validation (e.g. "Are we building the right product"). Further studies with experienced analysts may show that some people are sufficiently comfortable with the system and its functionality to warrant by-passing goals. However, for this analysis, it was imperative that the analyst always keep an eye on the overall objectives and goals to allow for the cohesive integration of functions. Additionally, the stakeholders worded the goals using computerese which often referred to actual functions within the system. The analyst is charged with managing these facets of the projects.

In constructing the CommerceNet goal hierarchy, the goals were organized according to the four goal classes shown in Table 6.6.

Table 6.6. Electronic Commerce Web Server Goal Classes

Goal Class	Description
Process Support	Process support goals involve the underlying functionality of the system as needed by the users. This class of goal was observed in the Electronic Commerce System.
Security and Access Goals	Security and access goals restrict access to certain parts of the system to authorized users. This goal class was observed in both the Vacation/Sick Leave System and the Electronic Commerce System
Electronic Commerce Goals	Electronic commerce goals involve transactions that take place within an Intranet and over the Internet.
Information Display and Organization Goals	These goals pertain to how information is organized throughout an information system and how the information is displayed for the users.

`Proposal voted on` is a *process support goal*; `Read/write access controlled` is a *security and access goal*; `Member billed for membership fees` is an *electronic commerce goal*; and `Search results Web page displayed` is an *information display and organization goal*. All of the identified CommerceNet goals fit into one of the four classes detailed in Table 6.6. The organization provided by the goals facilitated the organization of goals into naturally different functional requirements.

The CommerceNet case study further validates the GBRAM by demonstrating its successful application to a large commercial internet application. The method facilitated the identification and refinement of goals into operational requirements which guided a team of analysts throughout the process. The following section discusses further validation of the GBRAM; the validation process discussed sought to determine whether GBRAM allows for better identification of system requirements as compared to a commonly used method.

6.3 An Empirical Evaluation of the GBRAM

In this investigation, the Financial Services Office (FSO) is a real organization and the requirements are for a vacation/sick leave hour tracking system. The hypothesis for this empirical evaluation is that GBRAM allows analysts to better identify requirements than does another commonly used method or the lack of a method. Additionally, interest is also expressed in the difference in the nature of the requirements. This section provides an overview of the experimental method employed and discusses the results of the investigation.

Experimental Method

This experiment contrasts the use of GBRAM for the specification of information system requirements with one other analysis method, OMT [75], and a control condition in which no method was stipulated. This section explains the experimental method employed for this validation effort.

Design

The subjects were subdivided into three groups, each of which was asked to use one of the following methods:

- the Goal-Based Requirements Analysis Method (GBRAM);
- the Object Modeling Technique (OMT) [75]; or
- an analysis method of their choosing.

The independent measures involve two groups using different analysis methods and a third control group. The assignment materials supplied were analogous, but the subject groups differed in the assigned methodology.

Subjects

The subject population was a group of Georgia Tech undergraduate students enrolled in a senior level Management Information Systems course who were given the opportunity to voluntarily participate in an experiment for which they would be compensated with extra course credit. The subjects performed a problem analysis followed by a requirements analysis for the vacation/sick leave hour tracking system. Subjects were informed of the purpose of the study and assured that their identities would be concealed in the final report.

Measures were taken to ensure that the experts in a particular analysis method were not over-represented in any one condition. Questionnaires were administered to the students regarding their expertise level with different analysis methods; students were then assigned to the three conditions in a balanced fashion. Allocation was random within the expertise level. Although the students were initially subdivided into three groups with 8 students each, different participants dropped out of the study, resulting in condition groups with varying numbers of subjects (7 GBRAM subjects, 6 OMT subjects, and 4 CONTROL subjects). A total of seventeen subjects completed the analysis for this research.

Materials

Since this experiment entailed a two-part study, the materials* used for the experiment include two sets of detailed instructions. The first set consists of instructions for the problem analysis assignment for each of the three conditions; the second set consists of instructions for the requirements analysis assignment for each of the three conditions.

The instructions for Part One explain the objective of the study, provide a description of the problem, and present a step-by-step overview of the analysis method, providing a detailed example of how to apply the respective methods. The instructions for part two explain how to write a requirements document and how to translate the results of the problem analysis into requirements. The instructions for both parts explain several condition-dependent† ways to construct and represent software requirements.

The vacation/sick leave hour tracking system involves the process of monthly submission and tracking of employee vacation, sick leave, and consulting hours at an academic institution. Currently, all absences are either submitted in writing or via email to the Financial Services Office (FSO), which in turn generates various monthly and yearly reports using the submitted data. A new system is desired which supports employee hour submission and report generation for the FSO office.

This problem was well suited for our empirical analysis because the system involves various types of stakeholders with conflicting needs (e.g. faculty, full-time and part-time staff, and FSO employees responsible for processing the data and records responsibly) and it displays properties that are characteristic of information systems such as security, messaging,

*A copy of the materials is available from the author upon request.
†Condition: GBRAM, OMT, or CONTROL.

and report generation. Additionally, the scope of the system was sufficiently self-contained for a controlled experiment, not requiring a major time commitment of the participants in the study.

The GBRAM subjects were provided with blank spreadsheets and worksheets to facilitate capturing and tracking the number of identified goals. The OMT subjects were provided with blank worksheets to create a data dictionary, object models, and dynamic models. The control subjects were provided with blank worksheets for a table of definitions, flow charts, and a two-page handwritten description of the properties of the problem.

Procedure

The experimenter introduced the assignment in class and explained how to use the provided worksheets. The assignment was self-paced; subjects were free to work as much as they wished, completing the study as a homework assignment without supervision during a one-week period. For reasons beyond the control of the experimenter, some subjects were not able to start the assignment immediately; thus, the experiment continued for several weeks. Subjects were instructed to avoid discussing the assignment with other participants in this study.

The assigned task was a week-long effort. The assignment was presented in two parts: a problem analysis assignment and a requirements analysis assignment. Students were asked to sign a consent form and were provided with a five-page summary of the problem and the method to be employed. The assignment included two interview transcripts, one with a professor and the other with an FSO staff member. Thus, subjects had three sources of information available to two them: two interview transcripts and the problem description

in the assignment. Table 6.7 summarizes what subjects were required to produce for each of their respective conditions. The problem analysis assignment required the subjects to produce different method-dependent artifacts. The requirements analysis assignment resulted in the production of a requirements document, not to exceed 3 pages in length.

Table 6.7. Artifacts Produced by Subjects in Each Condition

Condition	Artifacts (Part 1)	Artifacts (Part 2)
GBRAM	Goal tables (goals, scenarios, obstacles) Elaborated scenarios (3) Goal hierarchy	Requirements document
OMT	Data dictionary Object model Dynamic models (3)	Requirements Document
CONTROL	Table of definitions Two-page description of the problem properties	Requirements Document

Subjects in all three conditions first read an introduction to and explanation of the vacation/sick leave system. They then read instructions for how to perform the problem analysis for the system. Results of their analysis were recorded on the provided spreadsheets and/or worksheets (e.g. GBRAM: goal tables, goal hierarchy; OMT: data dictionary, object model, dynamic model; and control: table of definitions, flow charts) and the assignment turned in at the beginning of class. By the end of that class period, the submitted problem analysis worksheets were copied and the original worksheets returned to the subjects with part two of the study, the requirements analysis assignment. Subjects were allotted a half-week to complete the requirements analysis assignment. For this second part, subjects were

asked to use a word-processor to compose their requirements documents.

A pilot study was initially run in a graduate level software engineering course. The requirements produced during this initial test were given to an independent judge* to rank the requirements on the basis of critical importance. Critical requirements are those requirements that cause a system to fail if not met; they established a threshold for a minimal set of requirements for the system. Of the 185 requirements identified by the three condition groups in the pilot study, 81 were ranked as critical.

Measurements

Some aspects of the experiment that may be compared across the three groups of subjects are:

- effort;
- total number of requirements and number of critical requirements expressed; and
- total number of requirements of different classes and number of critical requirements of those classes identified (detailed in Table 6.8 as the goal classes of interest in this study).

The effort expended cannot be analyzed for several reasons. First, the time required to use a method with which one is unfamiliar will vary greatly in comparison with a subject using a method with which they are an expert. Second, although the subjects were asked to keep track of the amount of time they spent on each phase of the analysis, informal interviews with subjects and an analysis of the submitted time data indicates that many

*The judge is a major stakeholder, not one of the previously interviewed stakeholders, as well as an expert in the application domain.

subjects merely jotted down a time for each phase as an after-thought, not taking the need for this information seriously.

Table 6.8. Vacation/Sick Leave System Goal Classes (See Chapter 4.2 for rationale)

Goal Class	Description
Calculation Goals	Calculation goals involve mathematical calculations. These may be in the form of accounting, accrual rates, balance, etc., as seen in the Vacation/Sick Leave Hour Tracking System.
Messaging Goals	Messaging goals pertain to notifications and reminders sent by and/or within the system. This goal class was observed in the Vacation/Sick Leave System.
Report Generation Goals	Report generation goals pertain to the generation of reports for an organization or enterprise. In the Vacation/Sick Leave System, the report generation goals involved the generation of internal departmental reports as well as external institute summary reports.
Security and Access Goals	Security and access goals restrict access to certain parts of the system to authorized users.

Results

This section discusses the results of the empirical evaluation of the Goal-Based Requirements Analysis Method. We employed the Mann Whitney U non-parametric test [68, 77] because of the small sample sizes and the large and possibly non-normal inter-subject variances. The resulting computation of the significance tests are shown in Table 6.9.

Figure 6.2. Total Number of Requirements Identified (Ordered by subjects in the three conditions). $GBRAM : OMT$ ($U = 11$, $p > .05$) and $GBRAM > CONTROL$ ($U = 4$, $p < .05$).

Figure 6.3. Number of Critical Requirements Identified (Ordered by subjects in the three conditions). $GBRAM > OMT$ (Mann Whitney $U = 6$, $p < .02$).

Of particular note are the results concerning the messaging requirements, shown in Figure 6.4. GBRAM subjects identified significantly more messaging requirements than the subjects in either of the two other conditions (GBRAM:OMT Mann Whitney $U = 6$, $p < .02$ and GBRAM:CONTROL $U = .5$, $p < .01$). An example of what is meant by "messaging requirements" is illustrated in Figure 6.5. Two obstacles are shown for the goal `Vacation/Sick leave hours collected`: `Hours not submitted` and `Employee never`

Figure 6.4. Number of Total Messaging Requirements Identified (Ordered by subjects in the three conditions). $GBRAM > OMT\ U = 6,\ p < .02$ and $GBRAM > CONTROL\ U = .5,\ p < .01$.

In order to better evaluate the performance of the GBRAM subjects who were using the method for the first time, an expert (the author) also performed the problem and requirements analysis using the exact same materials utilized during the controlled study. Figure 6.6 illustrates a comparison between the GBRAM experts' performance compared to the average performance of the subjects in the experiment. As may be expected, there is a great performance difference between the expermental sets; the expert out-performed the experiment subjects in all points of comparison.

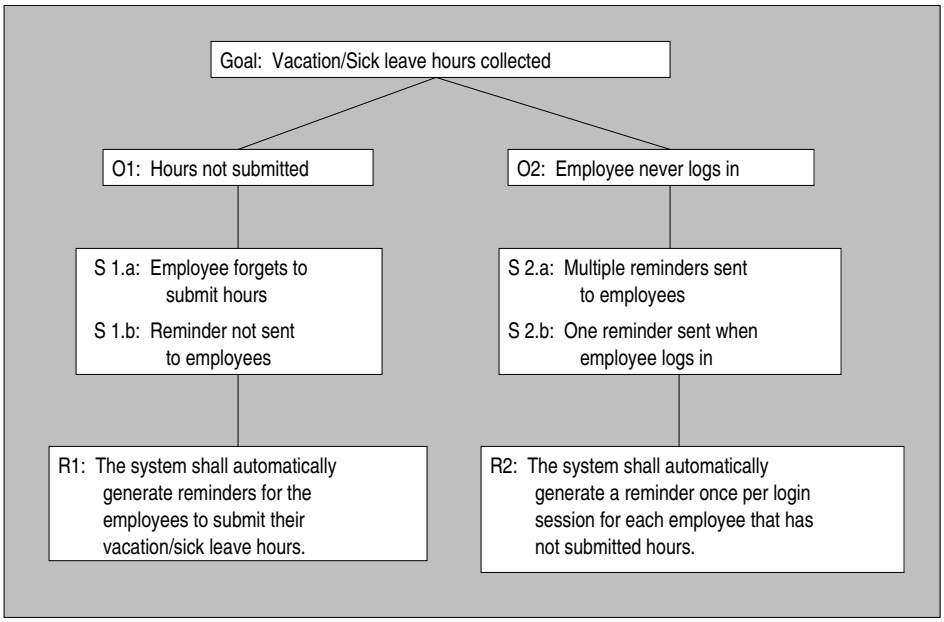

Figure 6.5. Messaging Requirements Example for Vacation/Sick Leave Hours Tracking System ($O = obstacle$, $S = scenario$, and $R = requirement$)

Figure 6.6. GBRAM Expert vs. GBRAM Subjects (Average)

Since the difference in performance observed between the GBRAM expert and the average of the subjects was so vast, an additional comparison was made between the GBRAM

*The concept of derived and synthesized goals was introduced by W. Michael McCracken during a discussion with the author of this thesis at Georgia Tech.

Figure 6.7. GBRAM Expert vs. Best Performing GBRAM Subject

The interest in tracking whether a goal is derived or synthesized is two-fold. This differentiation enables those goals which were simply extracted from available information sources to be distinguished from those goals which were uncovered as a result of an in-depth inquiry-driven analysis. Further investigation of the possibility of forcing traceability

from synthesized goals to derived goals thereby establishing reasonable stopping criteria for analysts is also an objective. Although collection of data for such an analysis has begun, analysis and further collection of data is left for future work, as discussed in Chapter 7.

Summary and Discussion

This study demonstrated the feasibility of employing the GBRAM as a reasonable analysis method, and is supported by similar identification of requirements for the proposed system across the GBRAM and OMT conditions.

Since the critical requirements are considered 'important', we can conclude that from this study that GBRAM performed as well as or better than OMT and the CONTROL conditions in identifying the important requirements. The most meaningful results in these studies are observed for the messaging requirements. GBRAM subjects out-performed both OMT and CONTROL subjects in the total number of identified messaging requirements, and out-performed OMT in the number of identified critical messaging requirements.

GBRAM relies on obstacle and scenario analysis to uncover exceptional behaviors in systems. Since scenarios are descriptions of sequences of behaviors, it naturally follows that they would yield behavioral requirements such as the messaging requirements in the vacation/sick leave system. GBRAM performed at least as well as the other two conditions in identifying goals and requirements for the other three goal classes.

A significant limitation of this evaluation of the method is the size of the subject population. Additionally, further comparisons are needed with other analysis methods which employ the use of scenarios, such as Jacobson's use case approach [48].

6.4 Summary

This chapter has presented three efforts (two case studies and an empirical evaluation) which served as the summative validation for the Goal-Based Requirements Analysis Method. Each case study detailed in this chapter involves a particular system:

- the Goal-Based Requirements Analysis Tool (GBRAT);
- the CommerceNet Web Server; and
- the Vacation/Sick Leave Hour Tracking System.

The Goal-Based Requirements Analysis Tool provides analysts with the procedural support needed for the analysis and refinement of goals for software-based information systems, supporting and guiding analysts as they identify, capture and structure information in the form of goals. This case study validated the efficacy of the GBRAM to acquire specific requirements for a tool to support the method itself, but limited in size and scope of the problem to which the GBRAM was applied. The CommerceNet Web Server case study validated the GBRAM's ability to handle enterprise and functional goals and the method's scalability for systems supporting a large organization comprised of multiple users holding conflicting goals. The empirical evaluation involving the Vacation/Sick Leave Hour Tracking System demonstrated the feasibility of employing the GBRAM as a reasonable analysis method, and was supported by similar identification of requirements for the proposed system across the GBRAM and Object Modeling Technique conditions. Table 6.10 summarizes the findings of each of the validation efforts discussed in this chapter.

Table 6.10. Summary of GBRAM Validation Efforts

GBRAT	CommerceNet	Vacation/Sick Leave
+ Viewpoints expressed using operational definitions + Implementation issues & alternatives raised using inquiry + Follow-up questions for stakeholder identified via constraints + Straightforward design & implementation by the developer as a result of requirements documentation + Redundant goals easily identified + GBRAM facilitated GBRAT's development	+ Goals easily named using standard subset of natural language terms + Auxiliary notes track unresolved issues and requirements discussions + Discussion items assist in development of requirements + Scenarios surface implementation alternatives + Obstacles and scenario analysis facilitate identification of new goals + Constraints may be refined into goals	+ GBRAM subjects identified more requirements than OMT & control subjects + GBRAM provides better coverage of requirements than does the control condition + GBRAM subjects identified more critical requirements than subjects in other two conditions + GBRAM subjects identified more messaging requirements than subjects in other two conditions + Obstacle & scenario analysis surfaced exceptional behaviors
• Operational definitions which supported the requirements leaned towards design • Requirements focused on system operations		• GBRAM performed at least at same level as other conditions in identifying calculation, report generation, & security/access goals
− An experienced analyst who was also the primary stakeholder & user of the proposed system tested GBRAM − GBRAM does not support information modeling − High level/organizational level goals were not part of this analysis − Only one stakeholder/user was actively involved in the analysis process, despite GBRAT's multi-user capabilities	− Stakeholders may be hesitant to focus on agents	− Size of subject population − Need comparisons with other scenario-intensive methods

A synopsis of each case study in this chapter was followed by a discussion of the methodology and the lessons learned. Discussion of the empirical evaluation encompassed the experimental method, the results, and a discussion of the findings. These case studies served as the source of summative validation, confirming the lessons learned about the GBRAM from the earlier case studies. The next chapter presents the conclusions of this dissertation and discusses future work.

CHAPTER 7

Conclusions

Let me tell you the secret that has led me to my goal.
My strength lies solely in my tenacity.

Louis Pasteur

The research in this thesis follows the "Industry as Laboratory" approach [62]. The Goal-Based Requirements Analysis Method (GBRAM) and the Goal-Based Requirements Analysis Tool (GBRAT) were developed by applying them to their own development as well as to other projects. The GBRAM evolved as a result of the refinements assembled from applying the method to the experiences detailed in Chapters 3 and 6 of this dissertation. In the first case study, goal decomposition and scenario analysis were applied in the context of Business Process Reengineering [7], demonstrating that both scenario and goal analysis are valuable for an effective analysis effort. The GBRAM was then evaluated by applying it to the requirements analysis of a continuing education system [4] and subsequently to the meeting scheduler system. Further validation of the method was sought by applying it to construct the requirements for a tool to support the method; this effort led to the design and implementation of the GBRAT prototype [6]. Through these various case studies, the method was evaluated and evolved based on the experiences and lessons learned. However, it was also necessary to reason about the scalability of GBRAM to larger com-

mercial applications. This was accomplished by applying the GBRAM to the redesign and reorganization of the CommerceNet Web server, which involved the active participation of several analysts and stakeholders. Finally, in an effort to compare the method to other requirements analysis approaches, an empirical evaluation was conducted as an additional validation of the method.

This chapter provides a synopsis of each chapter in this dissertation, presents a summary of the contributions of this work, and provides an overview of future work.

7.1 Chapter Synopsis

Chapter 1 introduced and articulated the problem addressed in this work. One of the principal problems in software requirements engineering is the difficulty in transforming a set of incomplete, inconsistent, and nebulous ideas into a complete, consistent elaboration of the technical requirements for a software system which are comprehensible to the intended customers and stakeholders. This chapter presented the importance of a good software requirements process in the context of a discussion of the consequences of poorly interpreted software requirements and detailed the importance of the issue in terms of increased software costs and potential software failures. The notion of goals as a means of early requirements validation was also outlined, giving rise to the research that has been presented in this dissertation.

Chapter 2 presented a survey of the related work in the field of software engineering and discussed the influence of goals in other fields.

Chapter 3 discussed three case studies: the Financial Services Office business process reengineering project, the Career Track Training System, and the meeting scheduler system. The methodology employed for each case study was presented, followed by discussions of the lessons learned. The case studies presented in the chapter are significant in that they were formative in nature. The GBRAM was simultaneously developed and informally validated while being applied to real projects; thus, the GBRAM evolved as a result of its application to the case studies that were discussed in this chapter.

Chapter 4 introduced the Goal-Based Requirements Analysis Method (GBRAM). The chapter provided an overview of the method, discussing the activities an analyst is involved with while employing the method and differentiating between the goal analysis and goal refinement phases. The method was illustrated in detail via the presentation of a reasonable sequence of activities, progressing from the initial identification of goals to translations of those goals into operational requirements.

Chapter 5 presented four general types of heuristics employed by analysts using the GBRAM: identification heuristics, classification heuristics, refinement heuristics, and elaboration heuristics. Several techniques were investigated as background for this research: scenario analysis, identification of goal obstacles and constraints, and goal operationalization. The heuristics and guidelines provided in this chapter were shown to be useful for identifying and analyzing the specified goals and were beneficial for their elaboration and refinement. The heuristics and supporting inquiry included references to appropriate construction of scenarios and the process in which they should be discussed and analyzed. A set of recurring question types was presented to assist analysts in applying an inquiry-driven approach to goal-based analysis which discussed the inquiry process, examples, potential

results, and final outcomes. Given a particular answer, analysts can determine the change which results from asking stimulative questions and the kind of refinement which may be made.

Chapter 6 presented the summative validation of the research discussed in this dissertation. Three specific validation efforts were conducted: two case studies and an empirical evaluation. In the first case study, GBRAM was employed to specify the requirements for tool support and was followed by construction of a prototype based on the resulting requirements. The second summative case study involved the reorganization of the CommerceNet Web Server. An empirical evaluation was also conducted in which the performance of one group of subjects using the GBRAM was compared with the performance of other groups of subjects using alternative analysis methods. Completion of these efforts conceptually validated the premise of this research.

7.2 Summary of Contributions

The principal contribution and focus of this work is the introduction of the Goal-Based Requirements Analysis Method for the identification and elaboration of goals and the refinement of those goals into operational requirements. Research for the method included the evaluation of existing goal-based approaches for the determination of the level of support and guidance they provide for the initial discovery of goals and requirements information. This, coupled with preliminary case studies, furnished the formative basis for the early development of the method. The GBRAM was developed to assist analysts in gathering software and enterprise goals from many sources and to support the process of elaborating

and refining goals into operational requirements. The method's chief contribution is the provision of heuristics and procedural guidance for identifying and constructing goals. A catalog of heuristics and questions for guiding the inquiry process was constructed and later validated and updated during the summative case studies.

The following three subsections outline the experiences demonstrated in this body of work, the primary contributions of this dissertation, and the limitations of the Goal-Based Requirements Analysis Method.

Demonstrated Experiences

The experiences reported in this dissertation demonstrate that:

- It is possible to specify effectively the functional requirements for a software-based information system using the heuristics and guidelines presented in this dissertation.
- Information pertaining to the goals and system objectives which are often overlooked when using other analysis methods are readily uncovered by employing GBRAM's elaboration and refinement heuristics throughout the analysis process.
- The inquiry cycle has been effectively applied/instantiated in conjunction with the set of recurring question types GBRAM offers to guide analysts in applying the approach.
- Goals offer a rich structure for organizing requirements information (i.e., obstacles, scenarios, constraints, and auxiliary notes) in the form of goal topographies which aid analysts in finding information and sorting goals into naturally different functional requirements.

Primary Contributions

The primary contributions of this dissertation are:

- the Goal-Based Requirements Analysis Method which provides prescriptive advice to analysts for the initial discovery and identification of goals;

- four sets of heuristics to assist analysts in identifying, classifying, refining, and elaborating goals (the 51 heuristics are summarized in Appendix C);
- a set of recurring question types, following the inquiry cycle approach, with suggested applications and resolutions; and
- a method for converting goals into operational requirements.

In addition, as a result of the GBRAT case study, proof of concept tool support has been developed in the form of the Goal-Based Requirements Analysis Tool (GBRAT) which is a multi-user Web-based tool designed to support the GBRAM.

Limitations of the GBRAM

The main limitations of the GBRAM are:

- the method provides informal, as opposed to formal semantics, for goals and thus it does not support formal reasoning;
- while well suited for identifying functional requirements which represent specific behaviors the proposed system should exhibit, GBRAM has not been adequately proven and tested for nonfunctional requirements other than general maintenance requirements; and
- while this dissertation presents some useful key words for the identification of various elements (e.g. goals, constraints, etc.), the method does not offer a complete lexicon of key words which would greatly assist analysts.

7.3 Future Work

The work presented in this dissertation addresses some of the fundamental problems with requirements identification; however, work remains to be done in these areas. This section provides an overview of areas of future interest.

The rationale for Michael Jackson's problem frames is similar to the rationale for goals classes in the GBRAM. Jackson asserts that before analyzing a software problem, an appropriate method for solving the problem must be chosen [46, 47]. His problem frames* provide a way to characterize, classify, and analyze software problems before beginning to solve the problem. Given a software problem, it is helpful to determine whether there are other problems which fit the same problem frame so that the methods and techniques which work for other problems may be applied to the problem currently being solved. Problem frames address broad problem characteristics and their strategies for solution.

Three of Jackson's problem frames are the JSP frame, the workpieces frame, and the environmental-effect frame [46,47]. In the JSP frame, the machine to be built is the program. The frame is tightly constrained with input and output streams (sequential structures of elements). Since the real word domain in the JSP frame is assumed to be autonomous, it is inadequate for an embedded system. The workpieces frame regards the machine as a production tool for textual or graphic documents. While suitable for an application involving text processing, i.e., a word processor, the workpieces frame is not suitable for large or embedded systems. The environmental-effect frame is suitable for an embedded system which controls an external domain. Jackson points out the need for more problem frames since all systems do not adequately fit into his proposed problem frames. For example, he discusses the Simple IS frame in [**?**], exposing its limitations as being too general.

The systems addressed in this dissertation do not adequately fit into the Simple IS frame, as it is too general a frame for most aspects of IS problems; however, some components of the problems addressed in this dissertation do fit into his workpieces frame (e.g.

*A problem frame is a generalization of a class of problem [46, 47].

the GBRAT tool). This dissertation introduced the notion of goal classes as generalizable to different software systems. A generalizable class or subclass of goals which may be viable for more than one system is envisioned (i.e., security and access goals). As illustrated in the CommerceNet and Vacation/Sick Leave case studies, goal classes are a way of carving up the problem domain. The problem domain may be decomposed into goals about process, security, etc., so that the classes dictate both what the system goals are and what they mean.

There appears to be a substantial overlap between the concept of goal classes and Jackson's problem frames. Future work will examine the commonality which underlies both of these approaches for characterizing software problems. Further investigation will determine whether there are clusters of specific goal classes in certain systems, such as those characterized by Jackson's problem frames. This determination will impact the heuristics presented in this dissertation by indicating refinements for the heuristics which call for the addition of specific questions for analysts to ask regarding goals which fall into particular goal classes.

Table 7.1 provides an overview of three generalizable goal classes discussed in this dissertation. If analysts can characterize the software problem according to goal classes before attempting to solve the problem and thus tailor the GBRAM to their needs; the method would be both easier to apply and more effective in its results. Just as the JSP frame is not well suited to embedded systems, it may be that certain goal classes or subclasses may not be well suited for analysis using certain GBRAM heuristics. Consider that the GBRAM's performance in comparison to other analysis methods was superior for the identification of messaging requirements, but comparable with respect to the other three goal classes in

the vacation/sick leave system. A goal class framework to characterize the types of problems which may be effectively analyzed using different heuristics and inquiry points in the GBRAM would enrich this work and increase usefulness for analysts who wish to employ the method.

Goal Class	Description
Messaging Goals	Messaging goals pertain to notifications and reminders sent by and/or within the system. This goal class was observed in the Vacation/Sick Leave System.
Process Support	Process support goals involve the underlying functionality of the system as needed by the users. This class of goal was observed in the Electronic Commerce System.
Security and Access Goals	Security and access goals restrict access to certain parts of the system to authorized users. This goal class was observed in both the Vacation/Sick Leave System and the Electronic Commerce System

Table 7.1. Generalizable Goal Classes

Future work will also involve extensions to the GBRAM. Two possible ways of structuring inquiry questions are under consideration: a process-based structure and a goal class structure. A process-based structure would offer analysts a set of questions structured according to when in the process the questions should be asked. Alternatively, questions may be organized and structured according to generalizable goal classes, as discussed earlier in this section.

This dissertation discussed the concept of employing a subset of natural language to comprise useful key words for the identification of the different elements which analysts work with when applying the GBRAM. Two extensions are under consideration to further

this notion of a standard set of key words: a lexicon and a goal language. A lexicon which analysts can employ for the identification of goals, obstacles, constraints, etc., would afford a more systematic and standard stock of terms than are currently offered by the method. Similarly, a simple goal language composed of the key words presented in this dissertation would provide a standard goal naming convention which would provide a way of representing goals as clauses with associated rules for manipulation (e.g. `MAKE [state] true` and `KNOW [state] acquired`).

The concept of derived and synthesized goals was introduced in Chapter 6 of this dissertation. Derived goals are those goals directly identified or explicitly extracted from descriptions of the problem. Synthesized goals are those goals identified and constructed through inquiry or due to the realization of the need for a new goal while analyzing the problem. As discussed in Chapter 6, data has been collected for the investigation of this differentiation between derived and synthesized goals; analysis of this will facilitate consideration of the implications of goal origins. Analysis of the traceability between derived and synthesized goals may lead to the development of stopping criteria for analysts during inquiry-driven analysis.

The use of goals and scenarios to augment natural language descriptions has been explored throughout this dissertation. However, strategies are needed to improve requirements documents. Future work will focus on reducing the level of ambiguity of these documents without the need for formal models. Additionally, requirements documents are typically difficult to index and offer no clear organizational structure. This dissertation presented the concept of goal topographies for structuring requirements information in software requirements documents (SRD). If goal topographies provide a reasonable way of structuring

SRDs, it naturally follows that topographies may provide a direct way to structure hypertext documents using goal topographies for the automatic generation of Web pages.

7.4 Conclusions

Goal analysis is a useful approach for the early validation of requirements. Multiple stakeholders have multiple goals; this multiplicity often augments the difficulty of developing a clear understanding of those goals. In the Goal-Based Requirements Analysis Method, the stakeholders' goals are clarified by tracking associated rationale. Stakeholders frequently discard systems because they do not understand how the system assists in the achievement of their goals, or because they use only a portion of the system. Analysts are charged with addressing both the root of this dissatisfaction and the cause of this inefficiency and misuse of effort.

It is commonly held that the problem with requirements engineering stems from the need to develop an understanding of the requirements expressed by stakeholders who do not themselves understand the requirements. At the same time there is a tendency to take short cuts during the requirements analysis task leading to an unstable design and often analysts fail to consider alternatives before the software is specified. The GBRAM may be modeled as an instance of the Inquiry Cycle approach [61, 64]. Goal analysis improves the understandability of the requirements by enabling analysts to produce a set of requirements which may be easily understood, validated, and readily accepted by the stakeholders involved in the process. In sum, each of the stages of the method correspond to a set of related expression activities; the heuristics of the method correspond to guided

discussion activities (i.e., standard questions and templates for reasonable answers and reasons); and detailed steps correspond to refinements.

The issues inherent in determining and specifying the requirements for software-based information systems are addressed by the Goal-Based Requirements Analysis Method. The strength of the GBRAM lies in its focus on goals and objectives and the derivation of operational requirements from those goals. The method offers a straightforward, methodical approach to identifying system and enterprise goals and requirements; it also suggests goal identification and refinement strategies and techniques through the inclusion of a set of heuristics, guidelines, and recurring question types. Use of this method produces a software specification of the functional requirements in the form of goal schemas depicting behaviors in terms of goals, relationships, constraints, obstacles, and agent responsibilities.

APPENDIX A

Vacation/Sick Leave Problem Description

An academic institution requires its employees to keep track of vacation hours, sick leave hours, and consulting hours. A more efficient system is needed to reduce the time spent tracking these hours each month. Currently, all absences must be recorded by each employee and accurate records of accrued/taken sick leave and vacation hours must be individuall maintained. Employees submit their "hours" via electronic mail to the director of financial services, who in turn generates a monthly time roster for the Payroll office. Payroll then produces the "accrued hours" report for each department. On December 31st, individual vacation records are adjusted to show no more than 45 accrued days of vacation time. There is no limit as to the total amount of sick leave which may be accrued.

APPENDIX B

Bugs Problem Description

---

*. We wish to simulate the evolution of "bugs" in a simple two-dimensional world. It is important to maintain a balanced and stable population in this eco-world. The world contains bacteria and bugs that eat the bacteria. The bacteria appear at random and persist at fixed locations until they are eaten. The bacteria in this world do not spread, age, or reproduce. Each bug has a variable position and orientation within the world; the bug population moves around the world randomly under the control of motion genes. Time is divided into uniform time steps in this two-dimensional world; during each step, each bug rotates randomly to a new orientation, then moves one unit forward in its new direction. Rotation is controlled by the motion gene. The world is divided into uniform cells with a finite number of angles such as a hexagonal grid with six possible angles. A bug eats any bacteria it finds within its cell, gaining a fixed amount of weight for each meal; however, at each time step the bug loses a fixed amount of weight to maintain its metabolism. If its weight becomes zero, the bug starves. If it weight exceeds a certain value, then the bug reproduces by splitting itself into two identical bugs each with half the original weight.

---

*This problem description is available in [74]

APPENDIX C

Summary of GBRAM Heuristics

Table C.1. Heuristics for Identifying Goals #1-7

Code	Heuristic	Page
HIG 1.	Goals are named in a standardized subset of natural language in which the first word is a verb that describes the kind of goal being named. For example, `AVOID` denotes one kind of goal. Goals of this kind are satisfied as long as their target conditions remain false.	144
HIG 2.	Abstraction mechanisms may be employed to extract goals from available documentation by asking: What goal(s) does this statement exemplify? and What goal(s) does this statement block or obstruct? If the answer to either of these questions is yes, then express the statement as a goals which represent a state that is desired or achieved within the system. The inquiry points are shown in Figure 5.3.	144
HIG 3.	Action words that point to some state that is or can be achieved once the action is completed are candidates for goals in the system. They are identified by considering each statement in the available documentation by asking: Does this behavior or action denote a state that has been achieved, or a desired state to be achieved? If the answer is yes, then express the answer to these questions as goals which represent a state that is desired or achieved within the system.	146
HIG 4.	An effective way to uncover hidden goals is to consider each action word and every description of behavior and persistently ask "Why?" until all the goal have been 'treated' and the analyst is confident that the rationale for each action is understood and expressed as a goal. The action words should be restated so that they denote a state that has been achieved or a desired state.	147
HIG 5.	Key action words such as: track, monitor, provide, supply, find out, know, avoid, ensure, keep, satisfy, complete, allocate, increase, speedup, improve, make, and achieve are useful for pointing to candidate goals.	147
HIG 6.	If a statement seems to guide design decisions at various levels within the system or organization, express it as a goal.	147
HIG 7.	Goals may be uncovered by examining the information available to identify avoidance goals. Avoidance goals are found by identifying bad states or states that should be avoided within the system.	148

Table C.2. Heuristics for Identifying Goals #8-14

Code	Heuristic	Page
HIG 8.	Goals can be uncovered or discovered by considering the goal dependencies for the previously specified goals by asking: What are the pre-conditions of this goal? and What are the post-conditions of this goal? Since preconditions and postconditions are expressed as goals in GBRAM, it is possible to identify new goals that had not been previously considered or identified by considering each goal's dependencies.	148
HIG 9.	Stakeholders tend to express their requirements in terms of operations and actions rather than goals [7, 25]. Thus, when given an interview transcript, it is beneficial to apply the action word strategy to extract goals from stakeholders' descriptions.	149
HIG 10.	Analysts should first seek to understand the stakeholder's application domain and goals before concentrating on the actual or current system so that the system requirements may be adequately specified. Previous research indicates that customers tend to express their goals within the context of their application domain, not in terms of an existing or desired system [7].	149
HIG 11.	Goals are also identified by considering the possible goal obstacles for previously specified goals.	149
HIG 12.	Goals may be identified by considering possible scenarios. Given each goal obstacle, the analyst should determine whether or not the occurence of the goal obstacle would initiate system failures, these obstacles are key candidates for scenario construction and analysis since the analyst must be sure to specify the goals and requirements to enable the system to handle exceptional cases. Goals may also be identified by considering the normal non-exceptional scenarios.	150
HIG 13.	Goals may be identified by considering constraints.	150
HIG 14.	Goals may be extracted from process diagrams by searching for actions and behaviors, as well as by consistently applying the Inquiry Cycle to clarify the goals and requirements.	151

Table C.3. Heuristics for Identifying Stakeholders and Agents

Code	Heuristic	Page
HIS 1.	Systems or subsystems which do not involve multiple stakeholders may not require stakeholder identification; analysts may choose to skip stakeholder identification entirely in these systems.	154
HIS 2.	Multiple stakeholders may be associated with one goal. If different stakeholders are associated with a goal, but their associations occur at different times within the life of the system, the analyst should document these variances to ensure that the role of stakeholders throughout the lifetime of a goal or the system is well understood.	156
HIS 3.	Any representative affected by the completion or prevention of a goal is a stakeholder. A customer or person representing the enterprise requesting the system or an analysis effort is a stakeholder. Users of the proposed system are stakeholders. Stakeholders are thus identified by asking: Who or what claims a stake in this goal? Who or what stands to gain or lose by the completion or prevention of this goal? Who will use the system?	156
HIA 1.	At least one agent must be responsible for the completion of each goal. If the analyst is unable to allocate responsibility for a goal to any agent, then the analyst can assume that the goal lies outside the scope of the proposed system. If the analyst believes there is a responsible agent, but doesn't know who or what, then the inquiry cycle should be applied using the Who-is question.	157
HIA 2.	Responsible agents may be identified by considering each goal and asking: Who or what agent *is*, *could be*, or *should be* responsible for this goal? The answer to this question will be the name of the responsible agent. The agent name should be 'attached' to the goal for which it is responsible. Table 4.4 on page 81 illustrates how agents can be attached to goals using a tabular notation.	157
HIA 3.	Different agents can be responsible for the completion of the same goal at different times.	158
HIA 4.	Agents may be either the system, organization, or a human agent.	158

Table C.4. Heuristics for Identifying Constraints

Code	Heuristic	Page
HIC 1.	Constraints can be identified by considering each statement and asking: Does this fragment impose some constraint on the goal(s)? Does this statement specify some requirement that must be met? Given an answer of 'yes' to either of these two questions, restate as a constraint every statement that exemplifies or states a requirement which must be met to achieve some goal.	159
HIC 2.	Constraints can be identified by searching for temporal connectives (i.e., *during*, *before*, *after*, etc.). Restate statements that describe *when* some condition is true or *when* a goal can be completed as a constraint.	159
HIC 3.	Constraints can be identified by searching statements which place limits on the completion of a goal.	160
HIC 4.	Since constraints may place a condition on the achievement of a goal, they should be restated as goal obstacles to allow for subsequent elaboration of the obstacle using scenario. This enables the consideration of exception cases which the system is required to handle.	160

Table C.5. Heuristics for Classifying Goals

Code	Heuristic	Page
HCA 1.	Goals are classified as achievement goals by considering each goal and asking: Is completion of this goal self-contained? Does this goal denote a state that has been achieved or a desired state? Does the completion of this goal depend on the completion of another goal? Is the ability of another goal to complete depend upon the completion of this goal? Given an answer of 'yes' to any of the questions above, classify the goal as an achievement goal.	161
HCA 2.	Achievement goals can be identified by searching for key words representing desired behaviors within the system (i.e., *make*, *improved*, *speed up*, *increase*, *satisfied*, *completed*, *allocated*, etc.)	162
HCA 3.	Achievement goals are relatively self-contained. While other goals may depend on the completion of the given goal, achievement goals rarely impose constraints upon an entire class of goals (e.g. a group of security and access goals). In contrast, a Maintenance goal is likely to impose a constraint upon an entire class of achievement goals.	163
HCM 1.	Goals are classified as maintenance goals by considering each identified 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? Is continuous achievement of this goal required?	164
HCM 2.	Maintenance goals can be identified by searching for key words that suggest a continuous state within the system (i.e., *keep*, *ensure*, *avoid*, *know*, *monitor*, *track*, *provide*, *supply*, etc.).	164
HCM 3.	Maintenance goals tend to be operationalized as actions that prevent certain states from being reached within the system. Since maintenance goals are those goals which are satisfied while their target condition remains true, they are named using the verbs `MAINTAIN`, `KEEP`, `AVOID` and `ENSURE`.	164

Table C.6. Heuristics for Refining Goals

Code	Heuristic	Page
HRR 1.	If the same goal appears more than once AND the same agent is responsible for the goal on each occurrence, then all but one of the goals may be eliminated.	167
HRR 2.	If the same goal appears more than once BUT two or more different agents are responsible for the same goal at different times, then all but one occurrence of the goal should be eliminated. However, to prevent the loss of information the analyst must keep track of all current and future agents who assume responsibility for the goal.	167
HRS 1.	If two goals are synonymous, reconcile the duplication by eliminating the goal which can be semantically subsumed by the other.	168
HRS 2.	If two goals are heterogeneous (e.g. one is an achievement goal and the other is a maintenance goal), then it is likely that the maintenance goal was classified incorrectly. If a maintenance goal is synonymous with an achievement goal, then the maintenance goal should be decomposed into an achievement goal. If the maintenance goal is decomposed into more than one achievement goal, then at least one of the goals should be synonymous with the originial achievement goals, and thus synonymous with the maintenance goal.	168
HRS 3.	Consolidate and refine goals by merging synonymous goals.	168
HRS 4.	Since synonymous goals tend to share precedence relations, they appear clustered together when ordered. Ordering goals according to their precedence relations thus facilitates the identification of synonymous goals.	168
HRSS 1.	Individual information dissemination goals may be refined by asking: What is 'information' and why is it significant or important? Do any goals depend on the availability of this information for goal achievement? Restate goals based on system-specific entities to capture the essence of the goal without including any system-specific information. When an implementation bias exists (i.e., if the customer has requested a certain implementation platform) then it may not be possible to ignore system-specific information. When an implementation bias exists (i.e., if the customer has requested a certain implementation platform) then it may not be possible to ignore system-specific information.	169
HRSS 2.	Restate routing goals to avoid emphasizing the receiving party and so that the underlying process and activity is represented.	170

Table C.7. Heuristics for Goal Elaboration via Dependencies and Obstacles

Code	Heuristic	Page
HED 1.	An effective way to discover precedence dependencies between goals is to consider each goal and ask: What goals are prerequisites for this goal? What goals must follow this goal? The answer to these questions indicate the given goal's precedence relations. They should be documented by the analyst so that the goal may be subsequently ordered according to these relationships. Chapter 4 on page 88 which analysts may employ for these annotations, or a tabular notation as shown in Table 5.3.	173
HED 2.	An effective way to identify contract dependencies is to consider each goal and ask: What goal(s) *must* be completed if this goal is achieved?	175
HED 3.	An effective way to identify agent dependencies between goals is to consider each goal and ask: What agent must complete the goal(s) they are responsible for before the agent responsible for this goal can achieve this goal?	176
HEO 1.	There is at least one goal obstacle for every goal. This is informally referred to as the trivial obstacle and formally referred to as the normal first case goal obstacles. These obstacles are worded by negating the verb in the goal name.	177
HEO 2.	A statement that illustrates a condition which prevents the completion of a goal or which illustrates an example of a goal being blocked by another goal is indicative of an obstacle and should be expressed as an obstacle.	179
HEO 3.	An effective way to identify goal obstacles is to consider each goal and ask: What other goal(s) or condition(s) does this goal depend on? What other goal(s) must be completed or achieved in order for this goal to be achieved? What goal(s) depend on this goal? What goal(s) must follow from this goal? Can the failure of another goal to complete cause this goal to be blocked? If this goal is blocked, what are the consequences? The answer to the questions above should be worded to emphasize the state that is true, thereby denoting a goal obstacle.	179
HEO 4.	A prerequisite failure obstacle occurs when a goal having a precedence relation is obstructed because the precedence goal fails. Prerequisite failures are identified by considering each goal and asking: What other goal(s) does this goal depend on?	180
HEO 5.	An agent failure obstacle occurs when a goal fails because the responsible agent fails to achieve the goal. Agent failures are identified by considering each goal and asking: Can the failure of an agent to fulfill their responsibilities cause this goal to fail?	180
HEO 6.	An contract failure obstacle occurs when a goal which holds a contract with another goal fails. Contract failure obstacles are identified by considering each goal and asking: Does this goal share a contractual relation with another goal?	181

Table C.8. Heuristics for Goal Elaboration via Scenarios

Code	Heuristic	Page
HES 1.	An effective way to identify candidate scenarios for construction is to consider each goal and ask: What happens if this goal is not achieved? What are the circumstances under which this obstacle can occur? The identified scenarios are elaborated by listing the activities that must occur should the scenario actually take place.	181
HES 2.	Another effective way to identify candidate scenarios is to consider each obstacle and ask: Why did this obstacle occur? Why was this goal not achieved? Under what circumstances would this obstacle occur?	183
HES 3.	The scenarios which analysts should provide particular or special attention to are those which violate goals or obstacles. Scenarios should be analyzed by considering the possible ways in which goal obstacles could be prevented. This process leads to the identification of new goals and requirements for the system.	183

Bibliography

[1] *CASE, The Potentials and Pitfalls.* QED Information Sciences, Inc., Wellesley, Mass., 1989.

[2] Ruessell J. Abbot. Program Design by Informal English Descriptions. *Communications of the ACM*, 26(11):882–894, November 1983.

[3] John S. Anderson and Brian Durney. Using Scenarios in Deficiency-driven Requirements Engineering. In *IEEE Intl. Symposium On Requirements Engineering*, pages 134–41, San Diego, CA, January 1993.

[4] A.I. Antón. Goal-Based Requirements Analysis. In *International Conference on Requirements Engineering (ICRE '96)*, pages 136–144, Colorado Springs, Colorado, USA, April 1996.

[5] A.I. Antón, T.A. Gale, W. Michael McCracken, and J.J. Shilling. Object Based Requirements Modelling for Process Continuity. In *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences*, volume 3, pages 191–202, Wailea, HI, USA, January 1993.

[6] A.I. Antón, E. Liang, and R.A. Rodenstein. A Web-Based Requirements Analysis Tool. In *Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '96)*, pages 238–243, Stanford, California, USA, June 1996.

[7] A.I. Antón, W. Michael McCracken, and Colin Potts. Goal Decomposition and Scenario Analysis in Business Process Reengineering. In *Advanced Information System Engineering: 6th International Conference, CAiSE '94 Proceedings*, pages 94–104, Utrecht, The Netherlands, June 1994.

[8] Kevin Benner, Martin S. Feather, W. Lewis Johnson, and Lorna Zorman. Utilizing Scenarios in the Software Development Process. In *IFIP WG 8.1 Working Conference on Information Systems Development Process*, December 1992.

[9] B.W. Boehm. *Software Engineering Economics.* Prentice Hall, 1981.

[10] Grady Booch. *Object Oriented Design with Applications.* Benjamin Cummings, Redwood City, California, 1991.

[11] George P. Burdell. *Disguise Her in White and Gold.* Buzz University Press, 50 Yard Line, Grant Field, 1908.

[12] John A. Byrne. Strategic Planning. *Business Week*, pages 46–52, August 1996.

[13] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1983.

[14] J.M. Carroll and M. Rosson. Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario. *ACM Transactions on Information Systems*, 10(2):181–212, April 1992.

[15] P.B. Checkland and J. Scholes. *Soft Systems Methodology in Action*. Wiley, 1990.

[16] Minder Chen and Jay F. Nunamaker. The Architecture and Design of a Collaborative Environment for Systems Definition. *DATABASE*, Winter/Spring 1991.

[17] Lawrence Chung, Brian Nixon, and Eric Yu. Using Non-Functional Requirements to Systematically Support Change. In *Proceedings 2nd International Symposium on Requirements Engineering (RE'95)*, pages 132–139, York, UK, March 1995.

[18] P. Coad and E. Yourdon. *Object Oriented Analysis*. Yourdon Press, 1991.

[19] D Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremaes. *Object-Oriented Development: The Fusion Method*. Prentice Hall, 1994.

[20] E.J. Conklin and K.C. Burgess Yakemovic. Report on a Development Project Use of an Issue-Based Information System. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, Los Angeles, California, USA, October 1990.

[21] Bill Curtis, Herb Krasner, and Neil Iscoe. A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*, 31(11), November 1988.

[22] R. Daniels, A. Dennis, G. Hayes, J. Nunamaker Jr., and J. Valacich. Enterprise Analyzer: Electronic Support for Group Requirements Elicitation. Technical report, University of Arizona, 1991.

[23] A. Dardenne, S. Fickas, and A. van Lamsweerde. Goal-directed Concept Acquisition in Requirements Elicitation. In *Proceedings 6th Intl. Workshop on Software Specification and Design*, pages 14–21, Como, Italy, October 1991.

[24] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed Requirements Acquisition. *Science of Computer Programming*, 20(1-2):3–50, April 1993.

[25] Anne Dardenne. On the Use of Scenarios in Requirements Acquisition. Technical Report CIS-TR-93-17, University of Oregon, Eugene, OR, 1993.

[26] Alan M. Davis. *Software Requirements: Analysis and Specification*. Prentice-Hall, 1992.

[27] Alan M. Davis. *Software Requirements: Objects, Functions, & States*. Prentice-Hall, 1993.

[28] S.M. Easterbrook. *Elicitation of Requirements from Multiple Perspectives*. PhD thesis, University of London, London, UK, 1991.

[29] Steve Easterbrook and Bashar Nuseibeh. Managing Inconsistencies in an Evolving Specification. In *Proceedings 2nd International Symposium on Requirements Engineering (RE'95)*, pages 48–55, York, UK, March 1995.

[30] Kurt D. Fenstermacher. Case-Based Teaching of Cardiac Auscultation. *http://www.cs.uchicago.edu/~fensterm/pubs/SpringSymp96/AI-med-sub.html*, February 1996.

[31] A. Finkelstein and Hugo Fuks. Multi-Party Specification. In *Proceedings of the 5th Intl. Workshop on Software Specification and Design*, pages 19–20, Pittsburgh, PA, May 1989.

[32] A. Finkelstein, J. Kramer, and M. Goedicke. Viewpoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering. In *Algebraic Methods II: Theory, Tools & Applications*, pages 29–54, Mierlo, Netherlands, September 1989.

[33] A. Finkelstein and C. Potts. Structured Common Sense: The Elicitation and Formalization of System Requirements. In *Software Engineering 86*, pages 236–50, September 1986.

[34] R.K. Fjelstad and W.T. Hamlen. Application Program Maintenance Study: Report to Our Respondents. In G. Parikh and N. Zvegintozov, editors, *Tutorial on Software Maintenance*. IEEE Computer Society, Philadelphia, PA, 1983.

[35] Joseph A. Goguen and Charlotte Linde. Techniques for Requirements Elicitation. In *IEEE Intl. Symposium On Requirements Engineering*, pages 152–164, San Diego, CA, January 1993.

[36] Orlena Gotel and Anthony Finkelstein. Contribution Structures. In *Proceedings 2nd International Symposium on Requirements Engineering (RE'95)*, pages 100–107, York, UK, March 1995.

[37] Orlena C.Z. Gotel and Anthony C.W. Finkelstein. An Analysis of the Requirements Traceability Problem. In *International Conference on Requirements Engineering (ICRE '94)*, pages 94–101, Colorado Springs, Colorado, USA, April 1994.

[38] S. Green. Goal-Driven Approaches to Requirements Engineering. Technical Report DoC TR-93-42 1994, Imperial College of Science, Technology and Medicine, Department of Computing Technical Report, London, UK, 1994.

[39] J.V. Guttag, J.J. Horning, and J.M Wing. The Larch Family of Specification Languages. *IEEE Software*, 2(5):24–26, September 1985.

[40] Peter Hall. *Great Planning Disasters*. University of California Press, Berkeley, California, 1980.

[41] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution.* HarperCollins Books, 1993.

[42] Michael Hammer. Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, 68:104–112, jul-aug 1990.

[43] Glenda Hayes. *Group Matrix tool: A Collaborative Modelling Tool.* PhD thesis, University of Arizona, Tuzon, AZ, 1991.

[44] Constance Heitmeyer, Bruce Labaw, and Daniel Kiskis. Consistency Checking of SCR-Style Requirements Specifications. In *Proceedings 2nd International Symposium on Requirements Engineering (RE'95)*, pages 56–63, York, UK, March 1995.

[45] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, and C. Chen. Formal Approach to Scenario Analysis. *IEEE Software*, 11(2):33–41, March 1994.

[46] Michael Jackson. Problems, Methods and Specialisation. *Software Engineering Journal*, 9(6):249–55, November 1994.

[47] Michael Jackson. *Software Requirement and Specifications.* Addison-Wesley, 1995.

[48] I. Jacobson. *Object-Oriented Software Engineering: A Use-Case Driven Approach.* Addison-Wesley, 1992.

[49] Capers Jones. Software Estimating Rules of Thumb. *IEEE Computer*, 29(3):116–188, March 1996.

[50] C.B. Jones. *Systematic Software Development Using VDM.* Prentice Hall, 1990.

[51] J. Kramer, K. Ng, C. Potts, and K. Whitehead. Tool Support for Requirements Analysis. *Software Engineering Journal*, 3(3):86–96, May 1988.

[52] Eiji Kuwana and James D. Herbsleb. Representing Knowledge in Requirements Engineering: An Empirical Study of what Software Engineers Need to Know. In *IEEE Intl. Symposium On Requirements Engineering (RE '93)*, pages 273–276, San Diego, CA, January 1993.

[53] J.C.S.P. Leite and P.A. Freeman. Requirements Validation Through Viewpoint Resolution. *IEEE Transactions on Software Engineering*, 17(12):1253–69, December 1991.

[54] Ted Lewis. Software Architectures: Divine Plan or Digital Darwinsim. *IEEE Computer*, 29(8):13–15, August 1996.

[55] Mitch Lubars, Colin Potts, and Charles Richter. Developing Initial OOA Models. In *Proc. 15th Int. Conf. Software Eng.* ACM/IEEE Comp. Soc. Press, 1993.

[56] Mitch Lubars, Colin Potts, and Charles Richter. A Review of the State of the Practice in Requirements Modeling. In *IEEE Intl. Symposium On Requirements Engineering (RE '93)*, pages 2–14, San Diego, CA, January 1993.

[57] Robyn R. Lutz. Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems. In *IEEE Intl. Symposium On Requirements Engineering*, pages 126–33, San Diego, CA, January 1993.

[58] N.A.M. Maiden and A.G. Sutcliffe. A Computational Mechanism For Parallel Problem Decomposition During Requirements Engineering. In *Proceedings of the 8th International Workshop on Software Specification and Design*, pages 159–63, Schloss Velen, Germany, March 1996.

[59] B. Nuseibeh, J. Kramer, and A. Finkelstein. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *IEEE Transactions on Software Engineering*, 20(10):760–73, October 1994.

[60] C. Potts. Using Schematic Scenarios to Understand User Needs. In *Symposium on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, pages 247–256, University of Michigan, Ann Arbor, Michigan, USA, August 1995.

[61] C. Potts and K. Takahashi. An Active Hypertext Model for System Requirements. In *Proceedings of the Seventh International Workshop on Software Specification and Design*, pages 62–68, Redondo Beach, California, December 1993.

[62] Colin Potts. Software Engineering Research Revisited. *IEEE Software*, 10(5):19–28, September 1993.

[63] Colin Potts. Supporing Software Design: Integrating Design Methods and Design Rationale. In T.P. Moran and J.M. Carroll, editors, *Design Rationale: Concepts, Techniques and Use*. Lawrence Erlbaum Associates, 1994.

[64] Colin Potts, Kenji Takahashi, and Annie I. Antón. Inquiry-Based Requirements Analysis. *IEEE Software*, 11(2):21–32, March 1994.

[65] William N. Robinson. Negotiation Behavior During Requirement Specification. In *Proceedings of the 12th International Conference on Software Engineering*, pages 268–76, Nice, France, March 1990.

[66] W.N. Robinson and S. Volkov. Conflict-Oriented Requirements Restructuring. Technical Report Working Paper CIS-96-15, Georgia State University, Atlanta, Georgia, September 1996.

[67] W.N. Robinson and S. Volkov. A Meta-Model for Restructuring Stakeholder Requirements (to appear). In *Proceedings of the 19th International Conference on Software Engineering*, Boston, Massachusetts, USA, May 1997.

[68] Colin Robson. *Experiment Design and Statistics in Psychology*. Penguin Books Ltd., Harmondsworth, Middlesex, England, 1973.

[69] Daniela Rosca, Sol Greenspan, Mark Feblowitz, and Chris Wild. A Decision Making Methodology in Support of the Business Rules Lifecycle. In *Proceedings 3nd International Symposium on Requirements Engineering (RE'97)*, pages 236–246, Annapolis, Maryland, USA, January 1997.

[70] Douglas T. Ross and Kenneth E. Schoman Jr. Structured Analysis for Requirements Definition. *IEEE Transactions on Software Engineering*, SE-3(1), January 1977.

[71] D.T. Ross. Structured Analysis (SA): A Language for Communicating Ideas. *IEEE Transactions on Software Engineering*, SE-3(1), January 1977.

[72] William B. Rouse. *Best Laid Plans*. PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[73] Kenneth S. Rubin and Adele Goldberg. Object Behavior Analysis. *Communications of the ACM*, 35(9):48–62, September 1992.

[74] James Rumbaugh. The Evolution of Bugs and Systems. *JOOP*, Nov-Dec 1991.

[75] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, New York, NY, 1991.

[76] David L. Schnitt. Reengineering the Organization Using Information Technology. *Journal of Systems Management*, 44:14–20, January 1993.

[77] Sidney Siegel. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill Kogakusha Ltd., 1973.

[78] H.A. Smith and J.D. McKeen. Reengineering the Corporation: Where Does I.S. Fit In? In *Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences.*, January 1993.

[79] J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, New York, 2nd Edition, 1992.

[80] J. Stage. The Use of Descriptions in Analysis and Design of Information Systems. In R.K. Stamper, P. Kerola, R. Lee, and K. Lyytinen, editors, *Collaborative Work, Social Communications and Information Systems*, pages 237–260. Elsevier Science Publishers B.V., 1991.

[81] A.G. Sutcliffe and N.A.M. Maiden. Bridging the Requirements Gap: Policies, Goals and Domains. In *Proceedings of the Seventh International Workshop on Software Specification and Design*, Redondo Beach, California, December 1993.

[82] A.G. Sutcliffe and N.A.M. Maiden. Domain Modeling for Reuse. In *Proceedings of the Third International Conference on Software Reuse: Advances in Software Reusability*, pages 169–77, Rio de Janeiro, Brazil, November 1994.

[83] K. Takahashi and Y. Yamamoto. An Analysis of Traceability in Requirements Documents. *IEICE Transactions on Information Systems*, E78-D(4):394–402, 1995.

[84] Kenji Takahashi. *Hypermedia Support for Requirements Analysis*. PhD thesis, Japan, 1997.

[85] Kenji Takahashi and Colin Potts. Tuiqiao: A Hypertool for Requirements Analysis. Technical Report GIT-CC-94107, Georgia Institute of Technology, Atlanta, Georgia, 1994.

[86] Kenji Takahashi, Colin Potts, Vinay Kumar, Kenji Ota, and Jeffrey D. Smith. Hypermedia Support for Collaboration in Requirements Analysis. In *International Conference on Requirements Engineering (ICRE '96)*, pages 31–40, Colorado Springs, Colorado, USA, April 1996.

[87] A. van Lamsweerde, R. Darimont, and P. Massonet. Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt. In *Proceedings 2nd International Symposium on Requirements Engineering (RE'95)*, pages 194–203, York, UK, March 1995.

[88] Eric Yu. Modeling Organizations for Information Systems Requirements Engineering. In *IEEE Intl. Symposium On Requirements Engineering*, pages 34–41, San Diego, CA, January 1993.

[89] Eric S.K. Yu. Towards Modeling and Reasoning Support for Early Phase Requirements Engineering. In *Proceedings 3nd International Symposium on Requirements Engineering (RE'97)*, pages 226–235, Annapolis, Maryland, USA, January 1997.

[90] E.S.K. Yu and J. Mylopoulos. Using Goals, Rules, and Methods to Support Reasoning in Business Process Reengineering. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences. Vol.IV: Information Systems: Collaboration Technology Organizational Systems and Technology*, pages 234–43, Wailea, HI, USA, January 1994.