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 automated 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.

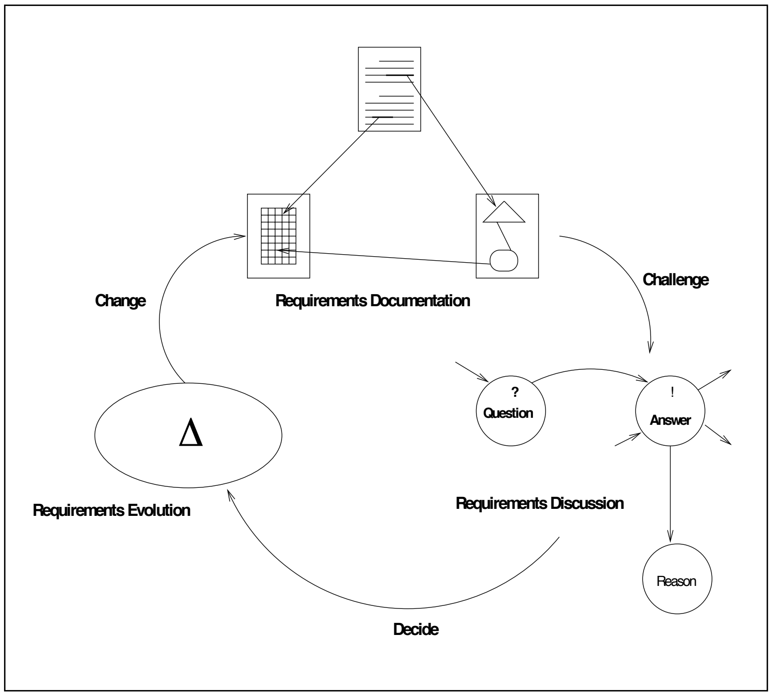

Figure 2.1. The Inquiry Cycle Model

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 errorless 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.