

Inconsistency Identification and Resolution in Goal-Driven Requirements Analysis.

Dempster, John Hampton

Ana (Annie) I. Antón, Committee Chair

ABSTRACT

This thesis addresses the need for inconsistency identification and resolution techniques to assist practitioners employing goal-based requirements analysis. Several approaches for inconsistency identification exist, however, the techniques introduced in this thesis are targeted specifically for analyzing goals in conjunction with the Goal-Based Requirements Analysis Method (GBRAM). Reducing inconsistency in a requirements specification greatly impacts its quality, as well as the overall success of a software development effort. Addressing and reducing the number of inconsistencies before development begins, minimizes the need to continually revisit what needs to be developed (the requirements), and consequently reduces development time and costs since inconsistencies are resolved early on (prior to system design and implementation). This thesis introduces a series of techniques and associated heuristics for identifying and resolving inconsistency in requirements specifications. These techniques and heuristics are demonstrated in the analysis of two case studies, the second of which is further described in *Deriving Goals from a Use Case Based Requirements Specification for an Electronic Commerce System* [ADS00]. The main contributions of this thesis include techniques that augment the GBRAM which provide specific guidance to the identification of inconsistency, and a structured goal syntax to aid analysts in forming goals for future analysis, and heuristics to aids in inconsistency identification and resolution.

**Inconsistency Identification and Resolution in Goal-Driven
Requirements Analysis.**

by
John Hampton Dempster

A thesis presented to the Academic Faculty of
North Carolina State University
in Partial Fulfillment of the
requirements for the Degree of
Master of Science

COMPUTER SCIENCE

August 2000

APPROVED BY:

Dr. Fay C. Payton

Dr. Mladen A. Vouk

Dr. Ana (Annie) I. Antón, Chair of Advisory

Committee

Personal Biography

John H. Dempster earned a Bachelor of Arts degree in Economics in 1990 from the University of North Carolina – Chapel Hill. Prior to returning to graduate school, he pursued a career in Trust Banking with Wachovia Bank of North Carolina, INVESCO Retirement Plan Services, and First Citizens Bank. During this time he gained experience in areas ranging from system requirements definition to investment portfolio evaluation. John returned to North Carolina State University in 1997 to pursue a Master of Science degree in Computer Science, and plans to graduate from the College of Engineering in the summer of 2000. He has accepted a position as a Project Manager with BridgePoint in Cary, N.C.

Acknowledgements

Nothing purchased can come close to the renewed sense of gratitude for having family and friends.

Courtland Milloy

I would like to thank the members of my committee, Drs. Antón, Payton, and Vouk, for their time and attention during the last few months. I owe special thanks to Dr. Antón for serving as both my editor and committee chairperson.

I would also like to thank the following students for their support during the preparation of this thesis: Devon, Jason, Ryan, and Thomas. Without humor this would never have been possible.

Finally, I would like to thank my family for their continued support throughout this process. Thanks to my parents whose support made this process possible. Thanks to Elizabeth for not only tutoring me through my prerequisite Calculus classes, but also for feeding Lyne and me on a weekly basis. Thanks to Bo for providing a cold beer and an attentive ear when I needed one. Thanks to Edy for constantly lifting my spirits. However, most of all, I give special thanks to my wife Lyne; now it's finally your turn.

Table of Contents

LIST OF TABLES	vi
LIST OF FIGURES	vii
LIST OF SYMBOLS AND ABBREVIATIONS	viii
GLOSSARY	ix
1. INTRODUCTION	1
1.1 Goal Driven Requirements Analysis	2
1.1.1 Overview of the Goal-Based Requirements Analysis Method	3
1.1.2 Inconsistency Identification and Resolution in the GBRAM	5
1.2 Related Work	6
1.2.1 Definitions of Inconsistency and Conflict	6
1.2.2 Methods for Identifying and Resolving Inconsistency	9
1.3 Overview of Remaining Chapters	13
2. EMPLOYEE BENEFIT SYSTEM CASE STUDY	14
2.1 Employee Benefit Information System	15
2.2 Lessons Learned	17
2.3 Summary	27
3. TECHNIQUES AND HEURISTICS FOR IDENTIFYING AND RESOLVING INCONSISTENCY FOR THE GOAL-BASED REQUIREMENTS ANALYSIS METHOD	28

3.1 Goal-Based Inconsistency Identification and Resolution	31
3.1.1 General Syntax for Expressing Goals	32
3.1.2 Syntax for Specific Goal Classes	36
3.1.2.1 User Goals	37
3.1.2.2 System Goals	38
3.1.2.3 Communication Goals	39
3.1.2.4 Knowledge Goals	40
3.1.2.5 Security Goals	41
3.1.2.6 Quality Goals	42
3.2 Inconsistency Identification Techniques	42
3.2.1 Techniques for Inconsistency	43
3.2.1.1 Organize Goals into Subject Groups	43
3.2.1.2 Build the Subject Tree	44
3.2.1.3 Compare Goals to Identify Inconsistency	44
3.2.1.4 Resolve Inconsistencies	45
3.3 Heuristics for Inconsistency Identification and Resolution	46
3.3.1 Inconsistency Identification Heuristics	47
3.3.2 Inconsistency Resolution Heuristics	50
3.4 Summary	53
4. ELECTRONIC COMMERCE SYSTEM CASE STUDY	54
4.1 Electronic Commerce System	55
4.2 Validation of Lessons, Techniques and Heuristics	57
4.2.1 Lessons Learned and Heuristics Revisited	57
4.2.2 New Lessons Learned and Heuristics	62
4.3 Summary	69
5. CONCLUSIONS AND FUTURE WORK	70
5.1 Summary of Demonstrated Experiences and Contributions	71
5.2 Future Work	71
5.3 Summary	74
REFERENCES	75
APPENDIX A	81

List of Tables

2.1	Sample Goal Inconsistencies Found During Pair Wise Comparison	25
2.2	Synonymous and Redundant Goals Identified	26
2.3	Decomposition of Compound Goals	27
3.1	Example of a Modifier that Mitigates Inconsistency	36
3.2	An Inconsistency That Can Not Be Resolved	46
3.3	Example of a Modifier that Mitigates Inconsistency	50
3.4	Two Inconsistent EBS Goals	51
3.5	EBS Goal Restatement to Mitigate Inconsistency	51
3.6	Goal Decomposition to Mitigate Inconsistency	52
3.7	Addition of a Goal to Mitigate Inconsistency	53
4.1	Two Inconsistent ECS Goals	61
4.2	Example of a Modifier That Mitigates Inconsistency	61
4.3	Two Inconsistent ECS Goals	62
4.4	ECS Goal Restatement to Mitigate Inconsistency	62

List of Figures

1.1	GBRAM Activity Diagram	4
2.1	Goal Network Created to Depict Goal Relationships	20
3.1	Flowchart of Inconsistency Identification and Resolution Activities	32
3.2	Goal Components	33
3.3	Keywords, Their Types, and Their Classes	34
3.4	Sample Subject Groups	47
3.5	Partial Subject Tree	49
4.1	Subject Tree for the ECS Case Study	67

List of Symbols and Abbreviations

\wedge	And
\vee	Or
\cap	Intersection
$ x $	Cardinality of x
$?$	0 or 1 occurrence
EBS	Employee Benefit System
ECS	Electronic Commerce System
GBRAM	Goal-Based Requirements Analysis Method
NCSU	North Carolina State University
RRA	Root Requirement Analysis

Glossary

- **Communication Goal:** A communication goal is associated with the delivery of information to the user.
- **Conflict:** A relationship between two goals such that one goal always prevents the achievement of the other in any circumstances.

As an example, consider two goals from a library system: ACHIEVE fines levied against all borrowers for late returns and ENSURE faculty not fined. These goals are at odds in all situations.

- **Construct:** A linked set of goal components.
- **Functional Goal:** Functional goals describe states that are achieved when an action is completed within a system [Ant97].
- **Goal:** A goal defines the actions to be achieved or the states to be maintained during the operation of a proposed system [Ant97].
- **Goal Class:** A goal class is a label that broadly describes a system area for the purpose of subdividing goals into subgroups.
- **Goal Component:** A component is one of the four pieces of a goal expression: keyword, subject, predicate, or modifier.

- **Goal Hierarchy:** An organization of goals classified by system function and ordered by precedence relations.
- **Goal Keyword:** A keyword identifies the type and class of a goal.
- **Goal Modifier:** A modifier describes a goal subject, and is expressed as a word or phrase; modifiers lend context to a goal by adding additional information that is important for future interpretation of the goal.
- **Goal Predicate:** A goal predicate defines the completed action that is taken by some entity (usually the user or the system) with respect to the goal.
- **Goal Subject:** The subject of a goal statement defines the system concept that is described or acted upon in the goal.
- **Goal Statement:** The linking of components to express a goal. A goal statement is made with respect to a defined syntax.
- **Goal Syntax:** A goal syntax affords a systematic way of arranging the different parts of a goal statement. A goal syntax defines all legal constructions of goal statements.
- **Goal Type:** A goal's type is either functional or non-functional.
- **Inconsistency:** A relationship between two goals such that one goal may thwart the achievement of the other given some set of circumstances.

For example, consider two goals from a library system: ENSURE book loaned as long as needed and ENSURE loaned books returned in two weeks. While books are returned within two weeks, there is no negation of either goal. However, if a book is

needed by a student for a semester-long project, and keeps the book as long as it is needed, then the second goal is violated.

- **Knowledge Goal:** A knowledge goal is associated with the data that should or should not be known by the system or by the users.
- **Non-Functional Goal:** Non-functional goals describe states that are maintained as long as their target condition remains true [Ant97].
- **Quality Goal:** A quality goal describes the system, its data, or its processes in terms of standards or constraints.
- **Security Goal:** A security goal is associated with access levels in a given system.
- **System Goal:** A system goal is associated with the processing actions or ongoing provision of services by a system.
- **User:** A *user* is any individual that interacts with the system, including end-users, system administrators, programmers, etc.
- **User Goal:** A user goal is associated with the actions performed by users while interacting with a given system.

Chapter 1

Introduction

A child of five would understand this. Send someone to fetch a child of five.

Groucho Marx

With the increased complexity of software over the past two decades, a need has arisen for more robust software development methodologies. Requirements engineering researchers and practitioners alike have searched for ways to standardize the software development process to reduce the time and cost associated with development, as well as to improve the quality of the software produced. Requirements engineering involves forming a comprehensive description of a given problem so that it may be addressed by a proposed system. Requirements engineers are responsible for communicating this description in an understandable yet unambiguous way to stakeholders, designers, developers, etc. As with any activity requiring human communication, requirements engineering is not a well-defined science.

One of the greatest challenges facing requirements engineering practitioners and researchers is the identification and resolution of inconsistencies within a

system's requirements. An *inconsistency* is defined in this thesis as a relationship between two goals such that one goal may thwart the achievement of the other given some set of circumstances. Requirements documents are often plagued with inconsistencies ranging from an inconsistent use of terms to conflicting requirements. The results of these inconsistencies can range from an increase in development time and cost to loss of life due to system failure. For this reason, it is imperative from a cost, quality, and safety perspective that inconsistencies be identified and resolved (if possible) in the early stages of a development effort. The research in this thesis addresses the need for more efficient inconsistency identification and resolution within the domain of goal-driven requirements engineering by offering techniques and heuristics that practitioners may use to identify and resolve inconsistency during goal-driven requirements analysis.

1.1 Goal-Driven Requirements Analysis

Requirements analysis is the study of a proposed system to determine its purpose and functionality [Ant97, LF91]. During requirements analysis, analysts gather information concerning the system to be implemented and document this information in a requirements specification. The requirements specification serves as a contract to set expectations for stakeholders and designers. Information from the requirements specification can also be used to aid in the development of a comprehensive test plan and individual test cases.

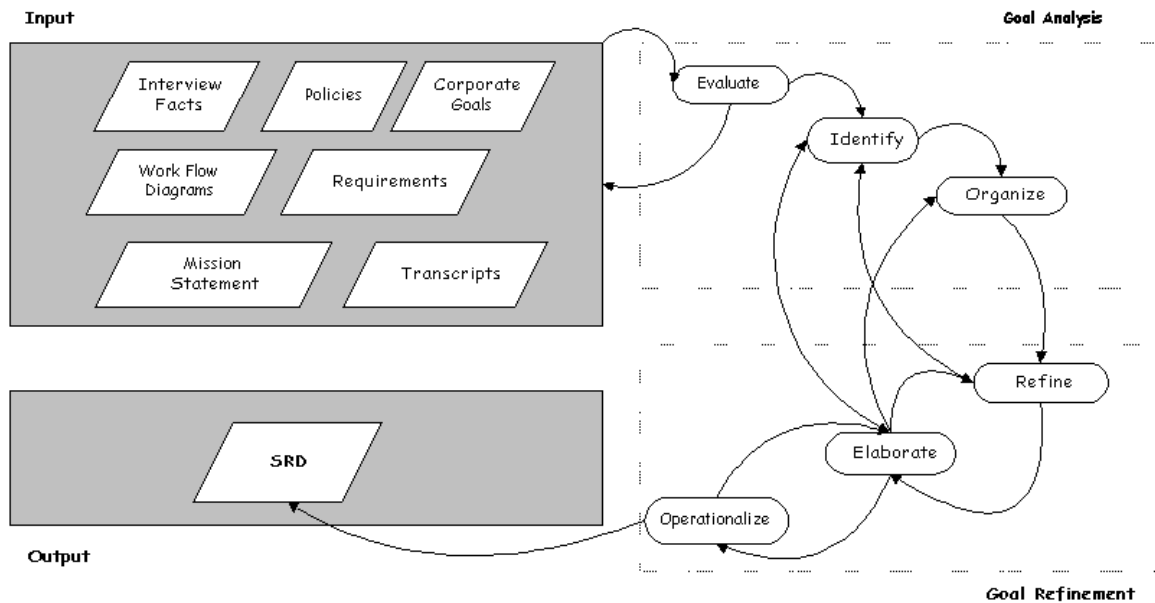
One approach to requirements analysis is goal-driven. Goal-driven requirements analysis advocates the initial identification of stakeholder goals instead

of stakeholder requirements. By identifying goals, an analyst is able to elicit more than a description of the envisioned implementation of a system from a stakeholder. Instead, eliciting goals helps an analyst understand the motivations driving a development effort. This approach avoids implementing a system that simply automates an existing process, and instead assists designers in providing more efficient ways of conducting business using technology. The method for implementing goal-driven requirements analysis, described in the next section, is used as the basis for the inconsistency identification and resolution techniques and heuristics presented in this thesis.

1.1.1 Overview of the Goal-Based Requirements Analysis Method

The Goal-Based Requirements Analysis Method (GBRAM) [Ant97] is a method designed to implement goal-driven requirements analysis. As shown in Figure 1.1, the GBRAM focuses on two types of activities: *goal analysis* and *goal refinement*. Goal analysis concerns the elicitation and documentation of goals, whereas goal refinement concerns the evolution of those goals from identification to operationalization. The following paragraphs detail each activity type.

Figure 1.1 – GBRAM Activity Diagram



Goal analysis consists of exploring goal source material, identifying goals, and organizing goals. Exploring goal source material involves the examination of all inputs to the goal elicitation process. These source materials may include interview transcripts, work flow diagrams, corporate goals and policies, and system requirements. The purpose of this exploration is to identify probable sources of goals, as well as to establish an order for examining each possible goal source. Identifying goals involves extracting goals, stakeholders, and agents from this source material. Each source document is reviewed with the purpose of documenting this information in a format conducive to the GBRAM analysis. Organizing the goals consists of classifying the goals by type and arranging them in a goal hierarchy by dependency relation. Techniques and heuristics for each of these operations are described in [Ant97].

Goal refinement consists of refining goals, elaborating goals, and operationalizing goals into a requirements specification. Refining the goal set consists of eliminating any redundant or synonymous goals identified in the goal hierarchy. Elaborating the goals consists of considering each goal for the purpose of documenting goal obstacles, scenarios, constraints, preconditions, postconditions, questions, and rationales. Finally, during operationalization, goals are restructured into requirements for placement into the requirements specification.

Note that the activities described above need not be performed sequentially. Instead, actions such as identifying a new goal may occur during any stage of the analysis, thereby starting the evolution of a new goal concurrent with the evolutions of the existing goals. Also, an analyst is capable of addressing the GBRAM activities in any order. For example, an analyst may decide to pursue goal elaboration prior to goal organization to facilitate better understanding of each goal.

1.1.2 Inconsistency Identification and Resolution in the GBRAM

Note from the discussion in Section 1.1.1 that inconsistency identification and resolution is not explicitly listed as a GBRAM activity. Although [Ant97] speaks to the use of ViewPoints [Eas91, EN95, FF89, FKG89, LF91, NKF94], it simply addresses these viewpoints according to the specific stakeholders, providing little prescriptive guidance for identifying and resolving inconsistencies. The GBRAM focuses refinement efforts on the organization of goals according to dependency

relationships¹ since goals that share the same dependencies are grouped closely in the GBRAM goal hierarchy. While this grouping does occur, and does facilitate the identification of synonymous and redundant goals, it falls short of providing a mechanism for inconsistency management since inconsistency can be based on relationships other than dependency. Instead, we observe, as discussed throughout this thesis, that the commonality between the meaning or function of the subject of each goal provides an additional basis for identifying inconsistency, since for goals to be inconsistent they must have some common attribute. Once these subjects are identified, they can be organized in a tree such that paths through the tree facilitate the identification of inconsistency by linking all related goals. This technique is discussed in detail in Chapter 3.

1.2 Related Work

This subsection discusses relevant work in inconsistency identification and resolution, and briefly compares this work with the research in this thesis.

1.2.1 Definitions of Inconsistency and Conflict

As previously stated, an *inconsistency* exists when there is a relationship between two goals such that satisfying one goal could, in at least some situations, result in negating the satisfaction of the other. We further define *conflict* as a relationship between two goals such that satisfying one goal results in negating the

¹ A contractual relationship exists between goal₁ and goal₂ if goal₂ must be completed if goal₁ is completed. A precedence relationship exists between goal₁ and goal₂ if goal₁ must be completed before goal₂.

satisfaction of the other in all situations. Therefore, according to our definition, conflict is a subset of inconsistency. Unfortunately, there is no common definition of inconsistency used in research today [vLD98]. This situation creates confusion for those attempting to survey inconsistency literature and compare methods of identification and resolution. Due to this situation, the first step for reviewing inconsistency identification and resolution techniques is to consider the definition of inconsistency as defined by several active researchers.

Robinson and Pawlowski define a *conflict* as a requirement that “depletes a shared resource”, “removes a precondition”, “removes the achieved effect”, or “has interfering actions with respect to another requirement” [RP98]. Notice that this definition is similar to our definition of conflict since it implies that a conflict between two goals represents an opposition in all situations. Further, Robinson and Pawlowski do not discuss the existence of conflict in limited circumstances, i.e. what is termed inconsistency in our research. Ignoring inconsistency is dangerous since these inconsistencies apply only in certain situations, and these situations may not be encountered until the system has been released, possibly causing system failure.

Easterbrook and Nuseibeh define *inconsistency* according to the ViewPoints framework; they define inconsistency as the breaking of a consistency rule within the ViewPoints Framework [EN96]. Consistency rules are defined by an individual responsible for overseeing the use of ViewPoints framework (called the *method designer*) by considering the rationale and operation of the framework, considering examples of use of the framework, and by reflecting on prior experience with the framework [EN95]. These rules dictate consistency relationships that should hold

across ViewPoints. Further, they define a conflict as an interference with the goals of one party caused by the actions of another party, and a mistake as an action that would be acknowledged as an error by its creator [EN96]. Unlike our definitions for inconsistency and conflict, this recognition of inconsistencies and conflicts in the ViewPoints framework is solely based on the effectiveness of the consistency rules. Should these consistency rules be lacking, inconsistency and conflict as we have defined them may exist without being identified or addressed. These inconsistencies and conflicts may persist into latter stages of development, or even into the software release.

Van Lamsweerde, Darimont and Letier broadly define *inconsistency* as a situation in which there is no way to satisfy all goals in a group [vLD98]. However, they address seven inconsistency types to refine their definition: process-level deviation, instance-level deviation, terminology clash, designation clash, structure clash, conflict, and divergence [vLD98]. Process-level and instance-level deviations are concerned with the violation of business rules governing the operation of the system. Terminology clash occurs when multiple real-world concepts are represented by one term. Structure clash occurs when a single real-world concept is given multiple designations. A conflict occurs when two or more goals are grouped such that one goal negates the effect of one or more other goals. A divergence occurs when there can exist a conflict between two or more goals, but only in limited circumstances called *boundary conditions*. The concept of divergence closely resembles our definition of inconsistency; divergences are conflicts that occur only under certain circumstances. Also, our definition of conflict encompasses not only

conflict as these authors define it, but also the other five types of inconsistency as defined in [vLDM95].

Finally, Liu, Tiao and Yen recognize four types of requirements: conflicting, cooperative, mutually exclusive, and irrelevant [YL96, YT97]. They define conflicting requirements as those that are related such that an increase in the degree of satisfaction of one requirement often results in a decrease in the degree of satisfaction of another requirement [YL96, YT97]. Further, they define mutually exclusive requirements as the satisfying of one requirement that negates the satisfying of the other [YL96]. Their definition of mutually exclusive mirrors our definition of conflict since our definition of conflict requires a complete negation of a goal or requirement under all circumstances, i.e. mutual exclusion. Additionally, their definition of conflict closely resembles our definition of inconsistency since under their definition of conflict two goals can conflict but still manage to be somewhat satisfied. One goal does not necessarily negate the other in all situations, thus allowing both goals to be satisfied in some situations.

1.2.2 Methods for Identifying and Resolving Inconsistency

There are several approaches for inconsistency identification and resolution. Robinson and Pawlowski propose Root Requirement Analysis (RRA) [RP98, RP99]. This method attempts to understand the relationships between requirements, and uses these relationships to identify key concepts of the system. Root requirements are found by iterating the following process: group requirements with similar concepts, and prune the set of requirements leaving only the most general requirement. This

process is repeated until groupings are no longer meaningful, leaving general requirements for the system, or *root requirements*. A pair wise comparison of the root requirements is then conducted to identify any conflicts. The result of the comparisons is a matrix listing the degree of potential conflict between each pair (very conflicting, conflicting, neutral, supportive, very supportive). This matrix provides information concerning the percentage of requirements with which a given requirement conflicts. Percentages are ranked, and the goals with the highest percentages are resolved first.

The observation that system concepts can guide detection of inconsistency in a number of comparisons less than pair wise is common to both RRA and the techniques presented in this thesis. Our techniques also resemble RRA by structuring goals for comparison without using formal methods. However unlike RRA, our techniques and heuristics provides greater coverage of information given that all goals are included in the comparisons, not just the general goals of the system. Additionally, we present heuristics for resolving the inconsistencies identified; RRA concentrates on identification, not resolution.

Easterbrook and Nuseibeh advocate the use of ViewPoints to identify and resolve inconsistency [Eas92 Eas93a, EFKN94, EN95, EN96, FKN92, FKG89, NE99, NFK94]. The ViewPoints framework documents multiple perspectives of a system that are then compared using consistency rules. Therefore, inconsistency identification is only as good as the rules on which it is based. The richer the set of consistency rules, the better the inconsistency detection. The ViewPoints are stored in a tree structure. When inconsistency is found, a ViewPoint is split into multiple

ViewPoints that differ only in their resolution of the inconsistency. Each ViewPoint is developed independently, thereby delaying the resolution of the inconsistency until all resolution options have been explored. The existence of these fully developed alternatives provides possible resolution strategies and consequences to an analyst.

The ViewPoints framework can be implemented in a distributed environment, and provides not only inconsistency identification, but also resolution alternatives. However, the quality of inconsistency detection is only as good as the inconsistency rules. Additionally, concepts that are the same across all ViewPoints are not always identified early in the process, and therefore the ViewPoints structure becomes difficult to manage. Our techniques are similar to the ViewPoints framework with respect to the structuring of information. ViewPoints uses a tree of perspective information, while we build a tree of subject information. Also, both ViewPoints and our techniques delay resolution of inconsistency until all information has been elicited and elaborated. However, ViewPoints delays resolution until all conflicts identified and all resolutions explored. Our techniques prescribe inconsistency resolution at the time the inconsistencies are found. While the ViewPoints framework contends that they explore all options by elaborating multiple resolution alternatives, the management of the information gathered in the ViewPoints framework is cumbersome. Our techniques address conflicts after goals have been identified, elaborated and organized. This process makes the handling of information easier, and includes a process for ensuring that a change to fix one inconsistency will not produce another.

Van Lamsweerde, Darimont, and Letier concentrate on the identification and resolution of divergence in [vLD98]. Specifically, they identify two methods of identifying divergence: regressing negated assumptions and divergence patterns. Regressing negated assumptions entails documenting the preconditions of a negated goal as obstacles to the original goal. In other words, analysts find inconsistency by asking what is a precondition of the opposite of a goal. These obstacles are potential boundary conditions for a divergence. Once these obstacles are found, they may be addressed. Divergence patterns are patterns of goals that have produced divergence in other studies. While these patterns seem promising, additional research is needed to build a rich set of patterns before they can be employed.

Van Lamsweerde et.al. recognize the difference between inconsistency and conflict, as well as the importance of structuring goals to aid in the identification of inconsistency [vLD98]. However, their use of formal specifications has been intentionally avoided in this thesis. Although regressing negated assumptions seem promising since it is efficient, it examines goals in isolation, thereby ignoring the most logical indicator of inconsistency, the other goals in the goal set. The authors suggest performing inconsistency detection while elaborating the goals identified. This identification is done independent of the other goals that have been identified. However, our techniques advocate the identification of inconsistency after all goals have been identified. We also advocate testing goals against each other instead of looking at them independently.

Liu, Tiao, and Yen concentrate on inconsistency resolution [YL96, YT97]. Specifically, they focus on mitigating inconsistency by partially satisfying one or

more of the inconsistent requirements and the concept of a marginal rate of substitution. The marginal rate of substitution quantifies the tradeoff between two goals, thereby allowing a quantitative assessment of requirement alternatives.

Liu et.al. [YL96, YT97] define the concepts of inconsistency and conflict as they are defined in this thesis. Additionally, they use a structure for requirements to assist in their understanding much as our method uses a syntax. However, their work only addresses inconsistency resolution, specifically inconsistencies that involve tradeoffs between two requirements. Therefore, our techniques are applicable to a greater range of inconsistent goals.

1.3 Overview of remaining Chapters

The research in this thesis addresses the need for more efficient inconsistency identification and resolution during goal-driven requirements engineering by offering techniques and heuristics by which practitioners may identify and resolve inconsistencies. Chapter 2 discusses the Employee Benefit System (EBS) case study. This analysis effort served to formalize our ideas about the identification and resolution of inconsistency, as well as to identify a portion of the heuristics presented in Chapter 3. Chapter 3 presents the inconsistency identification and resolution techniques and heuristics in detail. Chapter 4 discusses the Electronic Commerce System (ECS) case study, a validation of our techniques and heuristics. Finally, Chapter 5 reviews the contributions of this work to the field of requirements engineering, and discusses future research directions.

Chapter 2

Employee Benefit System Case Study

Do not seek to follow in the footsteps of the wise. Seek what they sought.

Basho

This chapter discusses the development of a set of techniques and heuristics for inconsistency identification and resolution during goal-driven requirements engineering. The techniques and heuristics were developed and validated on real problems. Our first case study, the Employee Benefit System (EBS), was formative in that it facilitated the development of a systematic approach to inconsistency identification and resolution, as discussed in Chapter 3. The approach was subsequently applied to a second system, discussed in Chapter 4; this subsequent analysis enabled the evaluation and refinement of the inconsistency identification and resolution strategies. The research methodology adopted for this thesis was, thus, marked by evolution and validation while analyzing real systems as advocated in [Pot93].

The EBS case study involved the analysis and design of an information system to provide health plan information to North Carolina State University (NCSU) employees. The analysis effort was conducted under the auspices of an academic

health care industry project; the objective of this project was to implement the proposed information system. The Goal-Based Requirements Analysis Method (GBRAM) [Ant96, Ant97, AP98a] was applied to the problem in an effort to specify the system requirements. The GBRAM analysis entailed the identification, elaboration and refinement of goals into operational requirements. Since the GBRAM provides only limited guidance for conflict identification and resolution, the EBS goal set was further analyzed in an effort to actively resolve any inconsistencies among the goals. Section 2.1 provides an overview of the NCSU Employee Benefit System analysis effort, and the lessons learned during this case study are discussed in Section 2.2.

2.1 Employee Benefit Information System

The EBS GBRAM analysis was originally commissioned to document the requirements for a health care information system in conjunction with research in the health care industry domain [Pay00, PG00, PL00]. The proposed system was designed to support decision making by providing health care plan information to NCSU employees; for example, the system provides the following types of information: plan provisions, plan costs, automated enrollment, automated physician selection, and physician evaluations and professional background information to aid in decision-making. The system also includes an employee survey questionnaire. The results from this questionnaire are aggregated according to plan and demographic group; employees may thus view the choices of other employees with similar health care preferences to aid their decision making process.

The requirements engineering efforts for this project entailed interviewing stakeholders including NCSU faculty members as well as representatives from various health care providers; the objective was to document not only what the faculty desired in such a system, but also what the health care providers considered important to communicate to both prospective and existing plan subscribers. The gathering of interview data, the subsequent translation of that data into goals, and the refinement and elaboration of the identified goals took place over the course of six months (February 1999 through July 1999). During this analysis six interviews were conducted with NCSU faculty. These interviews were recorded on cassette tape and later summarized according to the template in Appendix A. Goals were extracted from the interview summaries and then summarized in a Microsoft Excel spreadsheet as in [ADS00]. The goals were elaborated with obstacles, constraints, preconditions, postconditions, scenarios, questions, and rationales associated with each goal [Ant97]. This elaboration process resulted in both the refinement of 50 existing goals as well as the identification of 36 new goals. As prescribed in the GBRAM, a goal hierarchy was then constructed. The goals were grouped according to functionality similar to the approach taken in [ADS00, AP98a]; the five major goal classes in the EBS are: *input*, *output*, *system processing*, and *system help goals*.

The captured goals were analyzed to find patterns that may suggest strategies to improve goal-based requirements analysis by supplementing current goal driven analysis techniques with strategies for identifying inconsistencies among the goals. In accordance with other researchers who believe that goal structuring aids in the identification and/or resolution of inconsistency [Eas93, EN95, EFKN94, YL96], we

observed that a standard goal syntax for expressing goals aids analysts in expressing the goals in a manner conducive to inconsistency analysis. A goal *syntax* affords a systematic way of arranging the different parts of a goal statement. To this end, a goal syntax and a set of general heuristics were constructed based on the lessons learned during inconsistency resolution in the EBS analysis effort as discussed in the remainder of this chapter.

2.2 Lessons Learned

This subsection addresses the lessons learned in the EBS case study and discusses how these lessons were integral in the development of the inconsistency identification and resolution techniques.

1. The GBRAM does not adequately support conflict and inconsistency identification.

Goals that are inconsistent are often related with respect to some topic or action. To discover these relationships, we constructed a goal hierarchy based on precedence relations [Ant97]. We observed that this organizational structure does not lend itself to identifying goal relationships since it only shows temporal relationships between groups of goals in the same goal class (*input, output, system processing, etc.*). The dependency relationship is not the only relationship that may be associated with inconsistency.

Analysis of the goal hierarchy constructed for the EBS surfaced inconsistent goals that were associated with different goal classes. These goals were found while attempting to build a new goal hierarchy based on a revised set of goal classes.

Consequently, these goals were placed in different areas of the goal hierarchy. For example, two goals were identified, and arranged in different subtrees of the EBS goal hierarchy. Goal G_1 (MAKE \wedge survey \wedge obtained \wedge from all faculty) is a user input goal, and Goal G_{33} (ENSURE \wedge EBS \wedge "user-friendly") is a processing goal. There exists a potential inconsistency given that an EBS user may not be able to complete a survey should the system fail to be "user-friendly". Since there is no precedence relationship between G_1 and G_{33} , there exists no dependency relation between the goals as defined in the GBRAM [Ant97]; for example, a user may be able to complete a survey even though the EBS is not "user-friendly". Given that there is no dependency relationship between the two goals and that the two goals are classified differently, no relationship is recognized between these goals in the EBS hierarchy. The lack of recognition of such relationships is a shortcoming of the GBRAM goal hierarchy in general.

In an effort to provide a richer representation of goal relationships, we also constructed a goal network. A goal network is similar to a goal hierarchy in that related goals are connected, but unlike a goal hierarchy, there is no clear decomposition of goals into discrete subtrees. In a goal network, goals from one subtree may be connected to goals in another subtree, thereby creating a directed graph as opposed to a hierarchy.

2. Goal relationships aid in the identification of conflicts and inconsistencies.

The EBS goal network was constructed to surface relationships based on systems concepts. These concepts include system data items, system services, and

business rules. The graphical connections in the network (shown in Figure 2.1) enabled us to test for inconsistency between related goal pairs. However, this approach proved problematic since the goal network structure does not lend itself to a textual depiction and therefore a graphical depiction of related goal pairs was required. This graphical depiction proved to be difficult to read, even for the experienced analyst. Additionally, the creation of the network proved to be extremely time intensive, as it required a pair wise comparison of all goals. Consequently, this process was actually found to be less efficient than a traditional pair wise test for inconsistency [Kar96, KR96].

The goal network exercise did prove valuable in that it provided insights into the goals that were related by their respective system concepts. We observed that system concept relationships are represented as subsets of one another, or are directly attributable to one another, and therefore may be used to identify inconsistency. Since structuring requirements information aids in identifying and resolving inconsistency [Eas93, EN95, EFKN94, YL96], we constructed a standard goal structure for expressing goals. The goal structure, called a goal syntax, resembles natural language and enables goals to be parsed so they may be easily tested for the existence of system concept relationships.

3. A goal syntax is helpful in structuring goals for comparison, while allowing a natural expression.

According to Webster's dictionary, a goal syntax consists of the rules governing the construction of machine language or a systematic arrangement of words to form sentences [Web84], and in computing, a goal syntax is a way of arranging symbols in a language [Fre75]. Similarly, other researchers have structured goals and requirements for the purposes of inconsistency identification [Eas93, EN95, EFKN94, YL96]. A goal syntax affords a systematic way of arranging the different parts of a goal statement as goal components and, for purposes of this thesis, there are four such goal components: the keyword, subject, predicate, and modifier. To facilitate inconsistency identification, all goals should thus be expressed according to a goal syntax; our initial goal syntax was:

{keyword \wedge subject \wedge modifier}.

Section 3.1.1 discusses our catalog of possible keywords. It also defines a goal *subject* as a word or phrase that defines the system concept that is described or acted upon in the goal, and a goal *modifier* as a word or phrase that describes a goal subject. For example, for a meeting scheduler, a possible goal expression is (UPDATE \wedge room reservations \wedge immediately). This goal syntax, however, does not address both functional and non-functional goals. Recall that functional goals describe states that are achieved when an action is completed [Ant97], whereas non-functional goals describe states that are maintained as long as their target condition remains true [Ant97]. Due to the difference between functional and non-functional goals, it is apparent that two goal syntaxes are needed to consistently express all goals. Thus two goal formats were provided, one for functional goals and the other for non-functional goals. Functional goals were expressed as:

{keyword \wedge subject \wedge predicate \wedge modifier},

and non-functional goals as:

{keyword \wedge subject \wedge modifier}.

A goal predicate is defined as the completed action that is taken by some entity (usually the user or the system) with respect to the goal. For example, consider two goals for the meeting scheduler [PTA94, vDLM95]: (ACHIEVE \wedge meeting \wedge scheduled \wedge without conflict) and (UPDATE \wedge room calendar \wedge immediately). These two goals are differentiated as functional and non-functional by their keywords: ACHIEVE and UPDATE respectively. All goals are expressed using these keywords as there appeared to be no real need to distinguish between goal classes at this time. A *goal class* is a label that broadly describes a system area for

the purpose of subdividing goals into subgroup. This logic differs from previous research that advocates the association of goal types with various keywords [Ant97, ADS00]. While this goal syntax was successful in reducing goal complexity by reducing the variation across goal keywords, it is more difficult to distinguish between goal classes, making it consequently more difficult to organize goals efficiently.

We realized that it would be beneficial to classify goals by system-independent classes such as Security Goals or Quality Goals to facilitate the customization of the expression of each goal class to resemble natural language. Several revisions of the goal syntax were made to define keywords that could be associated with specific goal classes. Keywords were added to facilitate the recognition of the following goal classes: *User*, *System*, *Communication*, *Knowledge*, *Security*, and *Quality*. The distinction between goal classes was also found to aid in the grouping of goals that leads to defining goal relationships as discussed in Chapter 3.

4. Constructing a goal subject tree helps surface goal inconsistencies.

Creating a goal subject tree gives rise to goal relationships that point to potential inconsistencies. We assume that two goals that are completely unrelated cannot create an inconsistency since there is no relationship among them. It follows that goals must be related to be inconsistent. Therefore, by linking the system concepts that are described by goals, one can significantly increase the ability to find possible inconsistencies. If two subjects are related in that one is a subset of another, then there is a potential inconsistency between the two. For example, consider the

goals G_{19} (ENSURE \wedge demographic information \wedge confidential) and G_{84} (system information \wedge CONVEYED \wedge to help make health plan choices). The subject of G_{19} , demographic information, is a subset of the subject of G_{84} , system information. These goals are inconsistent since one goal indicates that information should be presented to the user, and the other indicates that a subset of this information should be kept confidential. It is this logic that led to the identification of twenty inconsistencies in the case study, whereas only two were identified in the GBRAM analysis.

5. *Pair wise comparison of goals is a thorough method of finding inconsistency so that it may be resolved. However, the use of goal subject relationships reduces the work involved in inconsistency identification.*

A pair wise comparison was made for all 86 goals in the goal set, yielding 25 inconsistencies. Consider the three examples of inconsistent goals in Table 2.1. The inconsistency between goals G_4 and G_{18} is due to the applicability of two goals that dictate different actions under the same circumstances. This inconsistency is mitigated with the addition of the modifier when not entered to G_{18} , which distinguishes the conditions under which each goal is applicable. The inconsistency between goals G_4 and G_{35} is due to the undefined meaning of demographic information. The information such as name, date of birth or sex that is to be considered part of demographic information must be defined before it can be deemed relevant or irrelevant. Adding subgoals to G_4 such as ACHIEVE \wedge date of birth \wedge entered) further defines the information to be entered. Stakeholders can

then aid in determining if an inconsistency actually exists, i.e. if information gathered is irrelevant. A possible inconsistency also exists between goals G_{39} and G_{35} . The system includes a biography for each physician; depending on the biography length, goals G_{39} and G_{35} could be in conflict. We mitigate this inconsistency by adding the modifier `within five lines` to goal G_{35} . The goal subjects for the 25 goals in this list were directly related and by considering the subjects it becomes quite easy to surface inconsistencies within the goal set. Assuming that all subjects are not directly related, the number of comparisons required will be a function of the number of related subjects, a number less than needed in pair wise comparison. By producing a goal subject tree, an analyst can define the goal relationships, and therefore the comparisons needed to find inconsistency. Hence, the strategy defined here is easier to implement than a goal network given the defined goal syntax and the limited number of subjects. This strategy produces a map from which to test for inconsistency with less than pair wise number of comparisons, by accurately associating goals with related subjects.

Table 2.1 – Sample Goal Inconsistencies Found During Pair Wise Comparison

Goal	Goal
G_4 : ACHIEVE \wedge demographic information \wedge entered	G_{18} : MAKE \wedge demographic information \wedge defaulted
G_4 : ACHIEVE \wedge demographic information \wedge entered	G_{35} : IGNORE \wedge irrelevant information
G_{39} : ENSURE \wedge text blocks \wedge short	G_{35} : CONVEY \wedge physician biographies

The number of comparisons needed for a true pair wise comparison is approximately $N^2 - N$. This number reduces substantially given that for every pair of goals found not to be directly related, the number of comparisons drops by N . This drop is significant. Given a group of 100 goals, a pair wise comparison would require

9,900 $((100 * 100) - 100)$ comparisons. If just one pair of goals was found to be unrelated, the number would drop to 9,800, thereby saving 100 comparisons.

6. *The use of a goal syntax aids in the identification of synonymous and compound goals.*

Defining terminology and structuring goals assists in identifying and eliminating synonymous goals. Eliminating synonymous goals reduces the number of goals in the goal set without a loss of stakeholder information, yielding a reduction in the complexity and work required to manipulate the goal set, as well as an improvement in the quality of the goal set. Table 2.2 shows four pairs of EBS goals that were identified as synonymous or redundant. Although unstructured attempts were made to reconcile such goals during the construction of the goal hierarchy, the goal pairs in Table 2.3 were not identified during that process. This suggests that the defined goal syntax is effective in detecting additional synonymous, redundant and/or inconsistent goals.

Table 2.2 – Synonymous and Redundant Goals Identified

Original Goals (Original Numbering)	Merged Goal
G ₇₂ : ENSURE ^ survey ^ completed before allowed to inquire G ₉₃ : ENSURE ^ survey ^ completed before exiting	G ₇₂ : ENSURE ^ survey ^ completed before exiting or inquiring
G ₂₂ : ACHIEVE ^ health plan survey information updates ^ obtained G ₁₄ : MAINTAIN ^ health plan survey information	G ₂₂ : UPDATE ^ health plan survey information
G ₇₀ : ACHIEVE ^ plan benefits ^ updated G ₁₆ : KEEP ^ plan information ^ updated	G ₇₀ : UPDATE ^ plan provisions
G ₉₀ : ACHIEVE ^ information ^ provided ^ to faculty member to make health plan choices G ₆ : ACHIEVE ^ information ^ provided to user	G ₉₀ : CONVEY ^ system information ^ to make health plan choices

A goal syntax also aids analysts in identifying compound goals. A compound goal is a goal that addresses multiple subjects; compound goals may be problematic in that they link subjects that may not behave identically in all situations. Therefore, compound goals should be atomic. Table 2.3 shows six instances where compound goals were identified and split into distinct goals.

Table 2.3 – Decomposition of Compound Goals

Original Goal (Original Numbering)	New Distinct Goals (Revised Numbering)
G ₂ : KNOW ∧ plan costs and provisions ∧ for all plans	G ₂ : KNOW ∧ plan provisions ∧ for all plans G ₃ : KNOW ∧ plan costs ∧ for all plans
G ₇ : KNOW ∧ demographic and health plan survey information ∧ for all plans	G ₆ : KNOW ∧ demographic information ∧ for all plans G ₇ : KNOW ∧ health plan survey information ∧ for all plans
G ₁₀ : CONVEY ∧ demographic and health plan survey information ∧ for all plans	G ₉ : CONVEY ∧ demographic information ∧ by plan G ₁₀ : CONVEY ∧ health plan survey information ∧ by plan
G ₂₆ : ENSURE ∧ plan provisions and costs ∧ formatted	G ₂₁ : ENSURE ∧ plan provisions ∧ formatted G ₂₂ : ENSURE ∧ plan costs ∧ formatted)
G ₆₆ : ENSURE ∧ survey questions ∧ clear and unambiguous	G ₅₉ : ENSURE ∧ survey questions ∧ clear G ₆₀ : ENSURE ∧ survey questions ∧ unambiguous
G ₂₅ : ACHIEVE ∧ plan costs and provisions ∧ provided ∧ to user	G ₈₇ : CONVEY ∧ plan costs ∧ to user G ₈₈ : CONVEY ∧ plan provisions ∧ to user)

2.3. Summary

This chapter describes the Employee Benefit System case study. First, the system and the context in which it was developed is discussed. Subsequently, the lessons learned from the case study are presented. Chapter 3 presents the techniques and heuristics for our approach to identifying and resolving inconsistencies during goal analysis.

Chapter 3

Techniques and Heuristics for Identifying and Resolving Inconsistency for the Goal-Based Requirements Analysis Method

Everything should be as simple as possible -- but not simpler.

Albert Einstein

This thesis addresses the need for inconsistency identification and resolution techniques to assist practitioners employing goal-based requirements analysis. Several approaches for inconsistency identification exist, as discussed in Chapter 1; however, the techniques introduced in this thesis are targeted specifically for analyzing goals. This chapter introduces a series of techniques and associated heuristics for identifying and resolving inconsistency in requirements specifications. Reducing inconsistency in a requirements specification greatly impacts its quality, as well as the overall success of a software development effort. Addressing and reducing the number of inconsistencies before development begins, minimizes the need to continually revisit what needs to be developed (the requirements), and consequently reduces development time and costs since inconsistencies are resolved early on (prior to system design and implementation).

The inconsistency identification and resolution techniques, as well as the associated heuristics, are designed to enhance goal-driven requirements analysis. Goal-driven requirements analysis attempts to capture system requirements by first defining, elaborating, and refining stakeholder goals, and subsequently by operationalizing those goals into requirements. A *goal* represents the actions to be achieved or the states to be maintained during the operation of a proposed system [Ant97]. Goals serve as a precursor to system requirements by defining the objectives of a system instead of the functions that implement it.

Our approach to inconsistency identification and resolution distinguishes between two goal types: functional and non-functional. *Functional* goals describe states that are achieved when an action is completed within a system [Ant97], whereas *non-functional* goals describe states that are maintained as long as their target condition remains true [Ant97]. The distinction between these two goal types requires no additional effort by an analyst since the goal syntax of each closely matches natural language as discussed in Section 3.1.2. In other words, the differentiation of goals as functional or non-functional should naturally emerge from the expression of the goal that the analyst chooses, given that he/she is familiar with and consistently applies the techniques and heuristics discussed in this chapter.

Our approach also distinguishes between goal classes [ADS00, Ant97]. The goal classes include *User*, *System*, *Communication*, *Knowledge*, *Security*, and *Quality*. These classes serve to organize the goals making them more manageable for an analyst, and provide a goal syntax that approximates the natural language for each

class, thus easing an analyst's burden of expressing goals. These goal classes are discussed in detail in Section 3.1.2.

This chapter details techniques and associated heuristics for identifying and resolving inconsistencies during goal-driven requirements analysis. Applying these techniques and heuristics enables inconsistencies to be identified after elicitation and elaboration of goals, but prior to their operationalization into requirements. The goal elicitation, elaboration, and operationalization processes are not detailed in this document, since the techniques and heuristics were developed as an extension to a specific goal-driven analysis method. An analyst wishing to use our techniques and heuristics may choose any goal-driven method, so long as it either permits the use of the specified goal syntax (see Section 3.1.2) or it separates the goal statements into similarly compatible components as detailed later in this chapter. For purposes of this thesis, the method was employed in conjunction with the Goal-Based Requirements Analysis Method (GBRAM) [Ant96, Ant97, AP98a]. The main objective of the GBRAM is to aid analysts in identifying and refining the goals for a desired system using specific elaboration and refinement techniques (e.g. scenarios and obstacle analysis). Using the information gathered in GBRAM, we construct a subject tree to represent goal relationships. These relationships are not dependency relationships (contractual and precedence) as depicted in GBRAM, but instead relationships based on similarity of meaning or function. We employ this subject tree to identify inconsistencies, resulting in the need for fewer goal comparisons than with other techniques such as pair wise comparison [Kar96, KR96]. Our techniques and

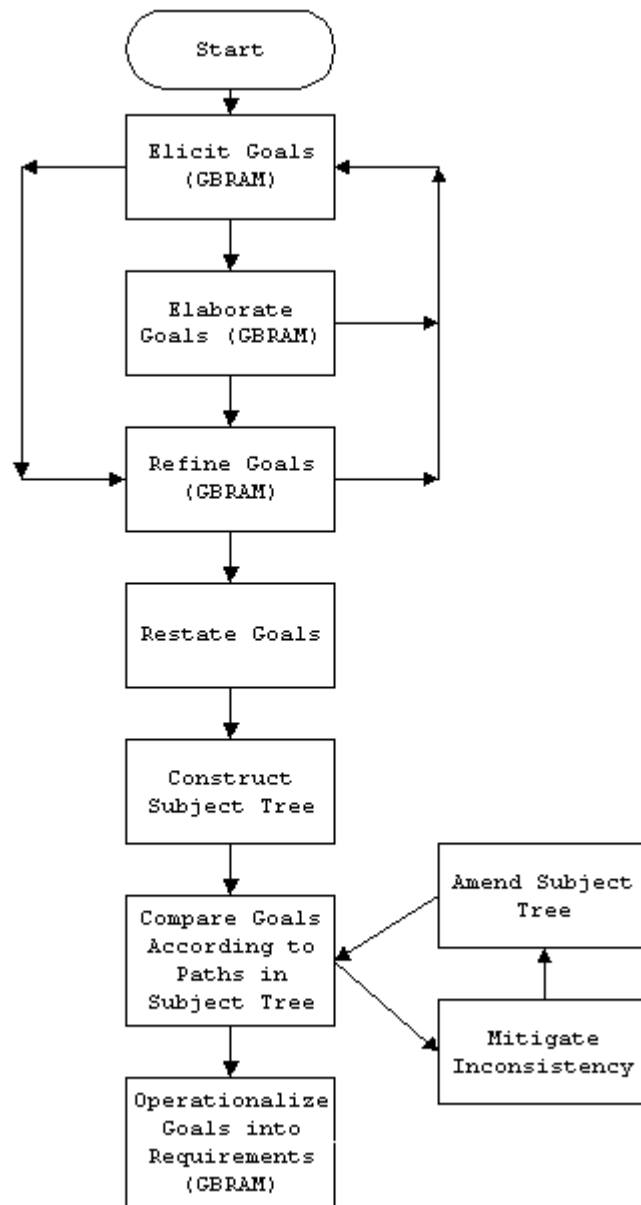
heuristics suggest strategies for resolving identified inconsistencies as discussed in Sections 3.2 and 3.3.

The discussion is presented in three parts: Section 3.1 details the goal syntax and goal classes associated with the inconsistency identification and resolution techniques, Section 3.2 details the inconsistency identification and resolution techniques, and Section 3.3 presents inconsistency identification and resolution heuristics. In order to elucidate many of the concepts discussed in the chapter, we provide examples taken from the case study detailed in Chapter 2: the Employee Benefit Information System (EBS).

3.1 Goal-Based Inconsistency Identification and Resolution

Figure 3.1 depicts the activities related to inconsistency identification and resolution. As shown in the figure, all goals must first be elicited, refined and, most importantly, expressed in a goal syntax that separates a goal into distinct components; these components are defined and discussed in the following section. A goal subject tree, similar to the goal hierarchy in [Ant97, DvL93], is then created to express the relationships between the goals. Once the goal relationships have been defined, goal comparisons are made as prescribed in the heuristics in Section 3.3. These techniques assume that there exists some relationship between goals that can be exploited to allow an analyst to identify inconsistencies without having to perform a pair wise comparison of all goals in the goal set. The following sections discuss the inconsistency identification and resolution techniques in greater detail.

Figure 3.1 Flowchart of Inconsistency Identification and Resolution Activities

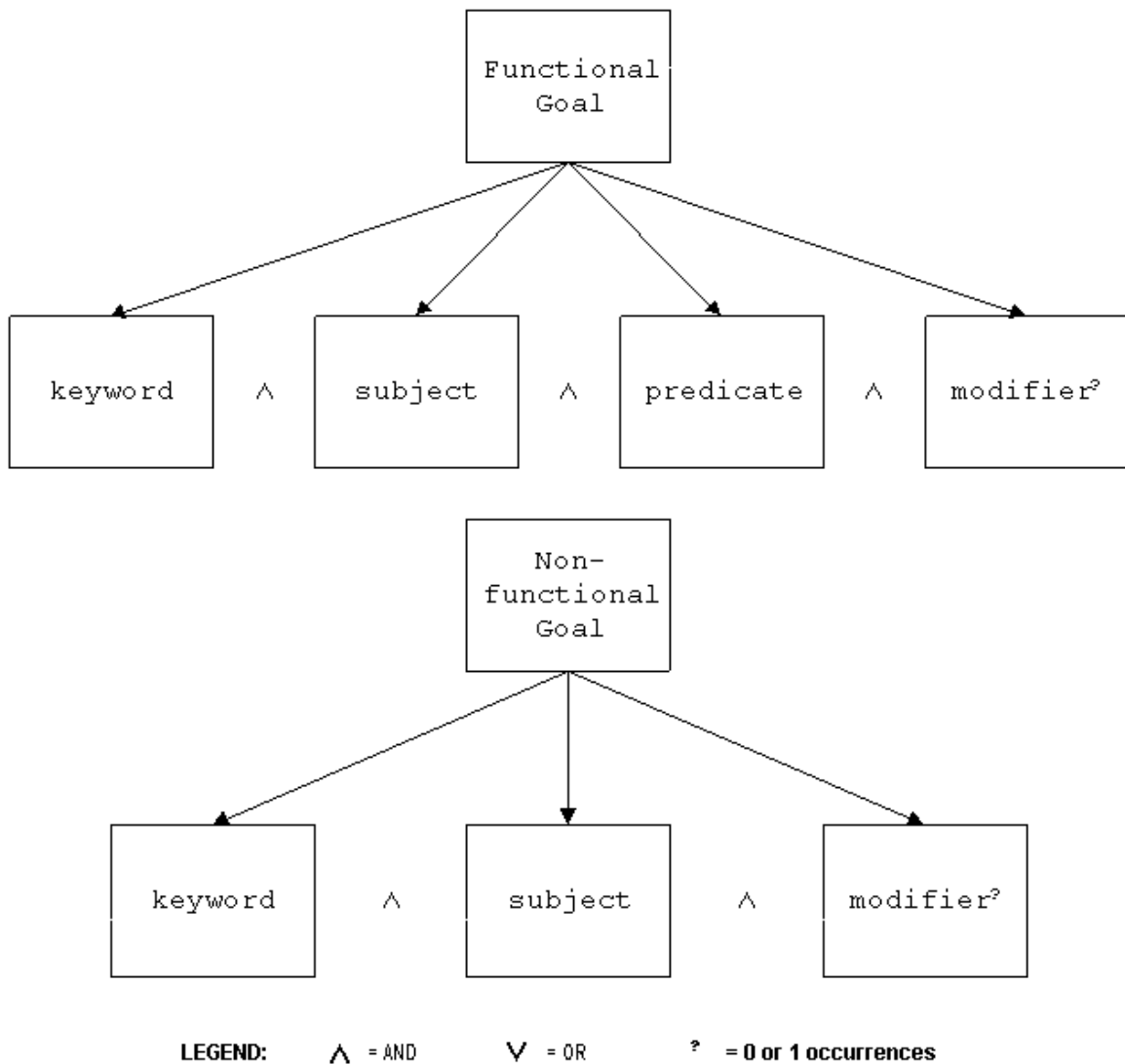


3.1.1 General Syntax for Expressing Goals

The defined goal syntax for the goal-based inconsistency identification and resolution techniques allows goals to be expressed as a combination of the goal components listed in Figure 3.2: *keyword*, *subject*, *predicate*, and *modifier*. The goal

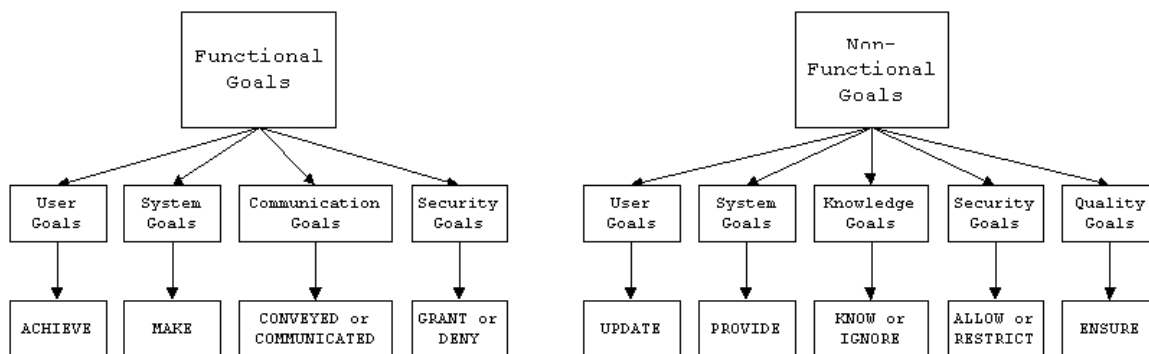
syntax is designed to ensure that all goals are expressed in a syntactically consistent form; this goal syntax facilitates the efficient parsing of goals into separate components that may be used for goal comparison. Both the keyword and subject are minimally required components for all goals. The predicate is required for all functional goals, but is not used with non-functional goals. Finally, the modifier is an optional component for all goals. We now discuss each of these elements in more detail.

Figure 3.2 – Goal Components



The *keyword*, required for all goals, identifies a goal by type and class. While the current set of keywords is sufficient to handle the systems that we have analyzed to date, this list is not exhaustive. Instead, the list is extensible so that analysts may add keywords while applying the method in different domains. Consider four goals from the Employee Benefit System from Chapter 3: G_4 (ACHIEVE \wedge demographic information \wedge entered), G_{32} (MAKE \wedge health plan survey information \wedge associated \wedge with user), G_{34} (PROVIDE \wedge logon help), and G_{67} (ALLOW \wedge inquiry access to survey questions \wedge to new faculty). The keywords for the listed goals are ACHIEVE, MAKE, PROVIDE, and ALLOW respectively. Keywords are a salient element of a goal statement since they are directly associated with a goal type and class as shown in Figure 3.3.

Figure 3.3 – Keywords, Their Types, and Their Classes



A goal statement’s *subject* defines the system concept that is described or acted upon in the goal. It may be expressed as a noun or noun phrase and always denotes concepts or entities associated with the system. A *system concept* may be a data item (e.g. demographic information in G_4), a system service (e.g. logon help in

G_{34}), or a system business rule (e.g. `inquiry access to survey questions in G67`). A *user* is any individual that interacts with the system, including end-users, system administrators, programmers, etc. Although the user is a catalyst for system action, the action affects the system concept, not the user; therefore, a user should never be the goal subject. For example, goal G_1 was originally expressed as `ACHIEVE faculty surveyed`. This goal was restated as `ACHIEVE survey completed by faculty` to better express the intention of having a survey completed. The survey is what is relevant to the system, and thus the goal statement's subject.

The goal statement's *predicate* is the action that is taken by some entity (usually the user or the system) with respect to the goal. Predicates in software engineering can be defined as "a generalization of facts" [Jac95, p.152]; given that we are approximating natural language, we define a predicate as "a verb or verb phrase that tells or asks something about the subject" [KM92, p.A41] for purposes of this thesis. Predicates should be expressed as past tense verbs, denoting that the action prescribed by the predicate must be completed before the goal may be satisfied. Note that not every goal has a predicate; for example, of the goals G_4 (`ACHIEVE \wedge demographic information \wedge entered`), G_{32} (`MAKE \wedge health plan survey information \wedge associated \wedge with user`), G_{34} (`PROVIDE \wedge logon help`), and G_{67} (`ALLOW \wedge inquiry access to survey questions \wedge to new faculty`), only G_4 and G_{32} have predicates. As previously stated, these predicates (`entered` in G_4 and `associated` in G_{32}) must be completed before the goals are satisfied. Generally, non-functional goals do not have a predicate since they describe the maintenance of a state, not its achievement through some action.

Consider G_{34} and G_{67} ; both goals are non-functional goals, and neither has an associated predicate.

The goal statement's *modifier* describes a goal subject, and is expressed as a word or phrase; modifiers lend context to a goal by adding additional information that is important for future goal interpretation. Consider the following EBS goals: G_4 (ACHIEVE \wedge demographic information \wedge entered), G_{32} (MAKE \wedge health plan survey information \wedge associated \wedge with user), G_{34} (PROVIDE \wedge logon help), and G_{67} (ALLOW \wedge inquiry access to survey questions \wedge to new faculty). A modifier is present in only two goals of these goals: with user in G_{32} and to new faculty in G_{67} . Modifiers are valuable when identifying inconsistency since they contain information that may differentiate two goals. For example, goals G_4 and G_{18} in Table 3.1 would clearly be inconsistent were it not for the modifier attached to G_{18} .

Table 3.1 – Example of a Modifier that Mitigates Inconsistency

Goal Number	Keyword	Subject	Predicate	Modifier
G_4	ACHIEVE	Demographic information	Entered	
G_{18}	MAKE	Demographic information	Defaulted	when not entered

3.1.2 Syntax for Specific Goal Classes

As previously mentioned, there are six broad goal classes: *user*, *system*, *communication*, *knowledge*, *security* and *quality goals*. Each of these classes is defined and detailed in Sections 3.1.2.1 – 3.1.2.6. Each goal class is associated with one or more constructs; a construct is a defined way to construct goals for specific class. The collection of all constructs comprises the goal syntax. Each construct consists of a keyword and a subject. Additionally, each construct may contain a

predicate and/or modifier. The following sections define and explain each goal type and their respective constructs, using examples from the EBS case study discussed in Chapter 2.

3.1.2.1 User Goals

User goals are associated with the actions performed by users while interacting with a given system. There are two constructs for user goals: one for functional goals and the other for non-functional goals. Recall that functional goals describe states that are achieved when an action is completed within a system [Ant97], whereas non-functional goals describe states that are maintained as long as their target condition remains true [Ant97]. Keywords are employed to distinguish between functional and non-functional goals. Specifically, goals beginning with the keyword `ACHIEVE` represent functional goals, while those beginning with the keyword `UPDATE` represent non-functional goals. The standard construct for functional user goals is:

$$\{\text{ACHIEVE} \wedge \text{subject} \wedge \text{predicate} \wedge \text{modifier}^?\}.$$

The keyword `ACHIEVE` is associated with functional goals since it implies goal completion. It is important to note that the predicate is always in the past tense since a goal is achieved when the action described by the predicate is completed. Consider the functional goal G_{90} : (`ACHIEVE` \wedge demographic information \wedge updated) from the EBS. This goal expresses the physical process of the user changing his/her demographic data in the online questionnaire.

The standard construct used for non-functional user goals is:

$$\{\text{UPDATE} \wedge \text{subject} \wedge \text{modifier}^?\}.$$

The keyword `UPDATE` is associated with non-functional goals since it may, and in this case is intended to, imply the continual change of information over the life of a system. For example, consider the following EBS non-functional goal G_{12} : `(UPDATE ^ demographic information)`. This goal corresponds to a policy that ensures users continually update their personal demographic information during the system life, or during their tenure as a university system employee.

3.1.2.2 System Goals

System goals are associated with the processing actions or ongoing provision of services by the system. As with user goals, there are two constructs for system goals: one for functional goals and the other for non-functional goals. Keywords are employed to distinguish between functional and non-functional goals. Specifically, goals beginning with the keyword `MAKE` represent functional goals, while those beginning with the keyword `PROVIDE` represent non-functional goals. The standard construct for functional system goals is:

$$\{\text{MAKE } \wedge \text{ subject } \wedge \text{ predicate } \wedge \text{ modifier}^?\}.$$

The keyword `MAKE` is associated with functional goals since it implies the goal. It is important to note that the predicate is always in the past tense since a goal is achieved when the action described by the predicate is completed. Consider the functional goal G_{75} `(MAKE ^ status code ^ changed ^ upon change from new faculty to regular faculty)` from the EBS. This goal expresses the physical process of the system changing an employee's status code within the system.

The standard construct used for non-functional system goals is:

{PROVIDE \wedge subject \wedge modifier[?]}.

The keyword PROVIDE is associated with non-functional goals since it implies the continual provision of a system service. For example, consider the following EBS non-functional goal G_{77} (PROVIDE \wedge status codes). This goal corresponds to the continuous provision of status data for each employee.

3.1.2.3 Communication Goals

Communication goals are associated with the delivery of information to the user. All communication goals are functional since they involve goal achievement. Therefore, there is only one construct for communication goals. Keywords are employed to distinguish between information display goals and messaging goals. Specifically, goals beginning with the keyword CONVEYED represent information display goals, while those beginning with the keyword COMMUNICATED represent messaging goals. That both keywords are stated in the past tense is important. Unlike other goal classes, the keywords serve as both keywords and predicates in communication goals. The standard construct for information display goals is:

{CONVEYED \wedge subject \wedge modifier[?]}.

The keyword CONVEYED is associated with information display goals since it implies the completion of the presentation of information. Consider the information display goal G_9 (CONVEYED \wedge demographic information \wedge by plan) from the EBS. This goal expresses the physical process of presenting data to the user.

The standard construct used for messaging goals is:

{COMMUNICATED \wedge subject \wedge modifier[?]}.

The keyword `COMMUNICATED` is associated with messaging goals since it implies communication between the system and another entity. For example, consider the following EBS non-functional goal G_{56} (`COMMUNICATED` \wedge `stale web link` \wedge `to user`). This goal expresses the physical process of communicating information to the user.

3.1.2.4 Knowledge Goals

Knowledge goals are associated with the data that should or should not be known by the system or by the users. All knowledge goals are non-functional since they involve the maintenance of a system state. Therefore, there is only one construct for knowledge goals. Once again, keywords are employed to distinguish between information that should be known and information that should not be known. Specifically, goals beginning with the keyword `KNOW` represent goals that express information that must be known to the user or system (perhaps as precondition for completion of another goal), while those beginning with the keyword `IGNORE` represent those that express information that is unimportant. The standard construct for knowledge goals is:

$$\{(\text{KNOW} \vee \text{IGNORE}) \wedge \text{subject} \wedge \text{modifier}^?\}.$$

Consider the knowledge goal G_3 (`KNOW` \wedge `plan costs` \wedge `for all plans`) from the EBS. This goal corresponds to the continuous state for the system to know the plan costs for all plans. Alternatively, consider the following knowledge goal G_{35} (`IGNORE` \wedge `irrelevant information`). This goal corresponds to the continuous state of ignoring information that is not needed or is not important.

3.1.2.5 Security Goals

Security goals are associated with user access in a given system. There are two constructs for security goals: one for functional goals and the other for non-functional goals. Again, keywords are employed to distinguish between functional and non-functional goals. Goals beginning with the keywords GRANT and DENY represent functional goals, while those beginning with the keywords ALLOW and RESTRICT represent non-functional goals. The standard construct for functional user goals is:

$$\{(\text{GRANT} \vee \text{DENY}) \wedge \text{subject} \wedge \text{predicate} \wedge \text{modifier}^?\}.$$

The keywords GRANT and DENY are associated with functional goals since they imply a provision of certain access levels, presumably to system users. The predicate is always in the past tense since a goal is achieved when the action described by the predicate is completed. Consider the functional goal G_{61} (DENY \wedge survey change access \wedge to user after survey completion) from the EBS. This goal expresses the physical process of changing a user's access level.

The standard construct used for non-functional user goals is:

$$\{(\text{ALLOW} \vee \text{RESTRICT}) \wedge \text{subject} \wedge \text{modifier}^?\}.$$

The keywords ALLOW and RESTRICT are associated with non-functional goals since they imply a change in the system's security policy. For example, consider the following EBS non-functional goal G_{12} (ALLOW \wedge survey inquiry access \wedge to new faculty); this goal corresponds to a policy which ensures that new faculty receive a given access level by default.

3.1.2.6 Quality Goals

Quality goals describe the system, its data, or its processes in terms of standards or constraints. All quality goals are non-functional since they involve maintenance of system state. Therefore, there is only one construct for quality goals. A keyword is employed to distinguish between quality goals and goals of other classes. Specifically, quality goals begin with the keyword ENSURE. The standard construct for knowledge goals is:

$$\{\text{ENSURE } \wedge \text{ subject } \wedge \text{ modifier}^?\}.$$

Consider the knowledge goal G_{19} (ENSURE \wedge demographic information \wedge confidential) from the EBS. This goal expresses a quality of the information that is to be maintained.

With the completion of goal elicitation, refinement, and statement in the goal syntax above, an analyst may begin the process of identifying and resolving inconsistencies. The next section details these techniques.

3.2 Inconsistency Identification and Resolution Techniques

This thesis defines an inconsistency as a situation in which a relationship between two goals may result in a conflict. Inconsistencies must be identified before they can be resolved. Thus, this research is mainly focused on finding inconsistency within sets of goals. Section 3.2.1 presents techniques by which inconsistencies can be found. These techniques consist of organizing the goal set, creating a subject tree, performing goal comparisons based on paths within the subject tree, and applying the heuristics in Section 3.3 to resolve the identified inconsistencies. A subject tree is a

tree in which there exists a node for each functional subject in the goal set, and each node contains all goals that share a common subject. Additionally, the goals with non-functional subjects are appended to the node that contains the functional subject that they most closely relate in meaning or function. Section 3.3 presents the heuristics to be applied during the application of the techniques.

3.2.1 Techniques for Inconsistency Identification and Resolution

The techniques for identifying and resolving inconsistency consist of re-organizing the goal set, creating the subject tree, performing goal comparisons based on paths within the subject tree, and mitigating inconsistencies according to the heuristics in Section 3.3.

3.2.1.1 Organize Goals into Subject Groups

Goals are first organized into *subject groups*. Subject groups are groups of goals that share the same subject. The subject groups are reviewed individually to determine if they contain any functional goals. Should a functional goal be detected, the group is termed a *functional subject group*. If no functional goal is detected, the group is termed a *non-functional subject group*. The differentiation between these two subject group types becomes important when the subject tree is constructed. The tree construction process is discussed in detail in the following paragraphs. Note that when multiple analysts are involved in goal elicitation, they should all be involved in creating the subject tree, since the intense domain knowledge gained during goal elicitation is crucial during the construction process. The domain knowledge gained

during elicitation gives the analyst(s) insight into the relationships that are leveraged to identify inconsistency.

3.2.1.2 Build the Subject Tree

The construction of the subject tree begins by populating the root node of the subject tree with any goals that have the system itself as the subject. The construction then continues by allocating nodes for each functional subject group. The answers for the following questions: “*Which subjects cover the broadest system concepts?*”, “*Which subjects are subsets of other subjects?*”, and “*Which subjects are similar in subject matter to other subjects?*” guide the placement of tree subject nodes to produce a tree in which goal subjects are decomposed with the most general at the root, to the most specific at the leaves. The non-functional subject groups are then added to the existing nodes by asking the following question: “*To which functional subject group is this non-functional subject group most similar in subject matter?*”. This subject tree is central to the method given that it documents the links between the nodes that are followed to find inconsistency.

3.2.1.3 Compare Goals to Identify Inconsistency

After completing the construction of the subject tree, an analyst proceeds by comparing goals. Beginning at the leaves of the tree and moving towards the root, pair wise comparisons are made between every goal in the current node, between every goal in the current node and every goal in any sibling node, and between every goal in the current node and every goal in every ancestor node. No node may become

the current node until all of its descendants have previously been tested thoroughly for inconsistency. Each comparison entails choosing two goals and asking the following questions: *“Is there any way, including special circumstances, that a failure of the first goal could cause a failure of the second goal?”* and *“Is there any way, including special circumstances, that a failure of the second goal could cause a failure of the first goal?”*. Should the answer to either of these questions be “yes”, an inconsistency exists.

3.2.1.4 Resolve Inconsistencies

Should an inconsistency be found, an attempt should be made to mitigate it immediately using the heuristics in Section 3.3. These heuristics guide the analyst in seeking the cause of and taking action to resolve the inconsistency. If a solution cannot be found, the inconsistency should be noted and revisited at a later time. Note that it is not always possible to immediately mitigate an identified inconsistency. There exist some instances where two valid goals compete with each other. In these situations, discussions with the stakeholders must occur to find a balance between the two goals. Table 3.2 shows an example of two goals that are valid yet inconsistent. A balance must be found between the two goals such that they can coexist. In the example, the stakeholders want privacy, but also want to track a user’s survey so that it may be updated. These two goals are both valid, but inconsistent with the goals of each other.

Table 3.2 – An Inconsistency That Can Not Be Resolved

Goal Number	Keyword	Subject	Predicate	Modifier
G ₂₇	ENSURE	health plan survey information		confidential
G ₃₂	MAKE	Health plan survey information	Associated	with user

Any goals modified during the identification and resolution process are revisited, repeating any comparisons made with goals in descendant nodes, the same node, sibling nodes, and ancestor nodes. Note that the analyst does not usually make comparisons between a goal and goals in descendant nodes. However, in the case of modifying a goal, comparisons previously made must be revisited to prevent the creation of a new inconsistency from the mitigation of an existing inconsistency.

3.3 Heuristics for Inconsistency Identification and Resolution

A number of observations were made concerning inconsistency identification and resolution. This section lists the heuristics developed from these observations. These heuristics are not designed to be exhaustive; analysts are expected to add heuristics as they address issues not encountered during the research for this thesis. Section 3.3.1 lists the heuristics for identifying inconsistencies (identified by HII) and Section 3.3.2 lists the heuristics for resolving inconsistencies (identified by HIR)

3.3.1 Inconsistency Identification Heuristics

There are six heuristics for identifying inconsistency among goals:

[HII1] Organize subject groups by grouping all goals that have the same subject. Subject groups containing at least one functional goal are termed functional subject groups. Subject groups containing no functional goals are termed non-functional subject groups. If a goal's keyword is listed under Non-functional Goals in Figure 3.3, then the goal is non-functional. If a goal's keyword is listed under Functional Goals in Figure 3.3, then the goal is functional.

Example 3.1: Figure 3.4 shows two subject groups from the EBS case study.

The functional subject group contains functional and non-functional goals.

The non-functional subject group contains only non-functional goals.

Figure 3.4 – Sample Subject Groups

<u>Functional Subject Group</u>	<u>Non-Functional Subject Group</u>
<u>Survey Questions</u> G ₆₆ ACHIEVE survey questions completed G ₅₉ ENSURE survey questions clear G ₆₉ ENSURE survey questions quantitative	<u>EBS</u> G ₅₄ ENSURE EBS readable G ₃₃ ENSURE EBS user-friendly G ₄₀ ENSURE EBS not boring G ₅₂ ENSURE EBS locatable

[HII2] When building the subject tree, begin construction by populating the root node of the tree with the subject group for the system regardless if it is a functional or non-functional subject group.

Example 3.2: In Figure 3.5, the root node subject for the Employee Benefit System is EBS. Further population of the tree begins from this point.

[HII3] Construct the subject tree from the root by populating nodes with the functional subject groups corresponding to the most general subjects. Nodes addressing the most specific system concepts should be placed closer to the leaves of the tree. Functional subject groups are placed in the tree by asking the following questions:

a) *Which subjects cover the broadest system concepts?*

These subjects should be closest to the root node.

b) *Which subjects are subsets of other subjects?*

If subject x is a subset of subject y, then subject x should be placed in a leaf node of the node containing subject y.

c) *Which subjects are similar in subject matter to other subjects?*

Subjects that are similar in subject matter to other subjects should be listed near each other in the subject tree.

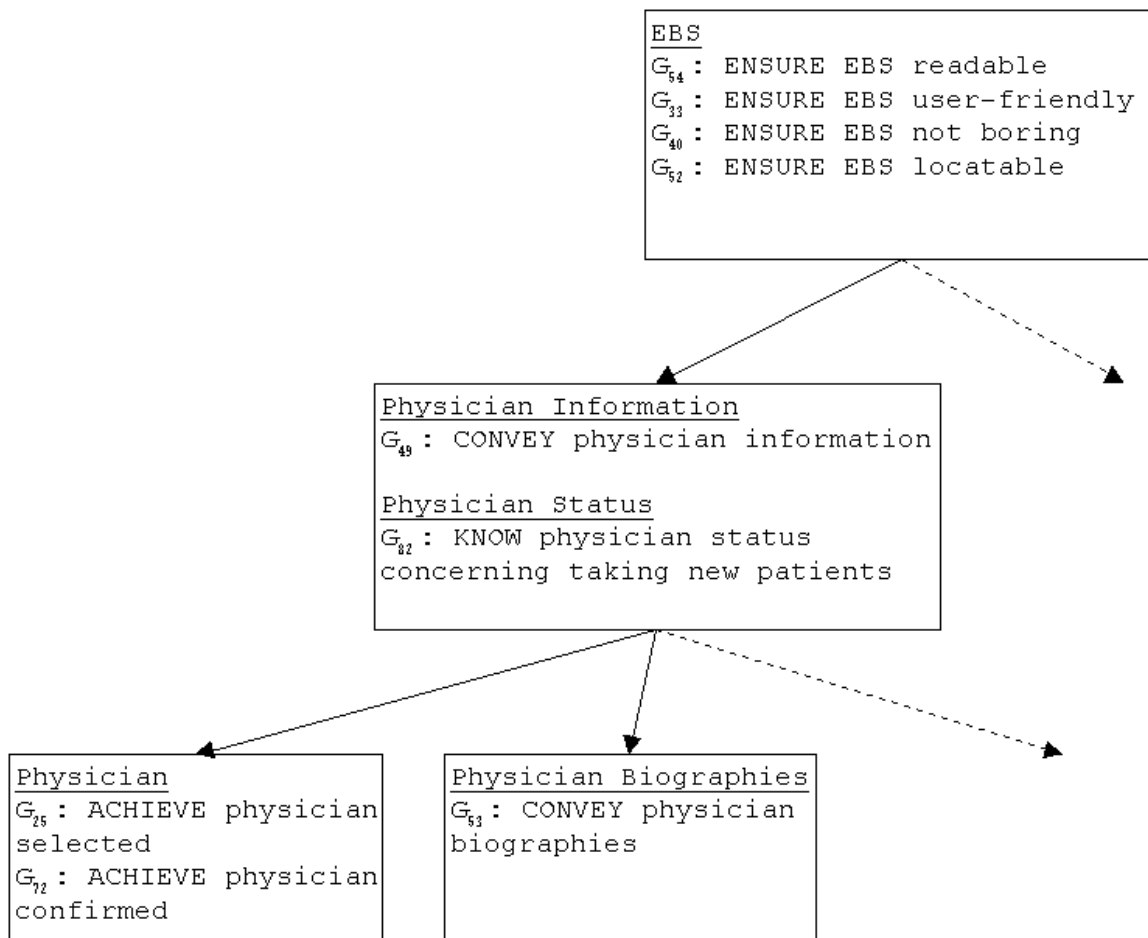
[HII4] *Non-functional subject groups* are placed in the subject tree after the placement of *functional subject groups*. Non-functional subject groups share the node of the functional group with which they are most similar. The non-functional subject groups are placed in the subject tree by answering the following question.

a) *To which functional subject group is this non-functional subject group most similar in subject matter?*

The answer to this should be used in conjunction with the answers to the questions in HII3 to place subject groups.

Example 3.3: Figure 3.5 shows a partial subject tree. All nodes contain a functional subject group; additionally, one node contains both a functional and non-functional subject group. Note that the root node is the only node that may contain only non-functional subject groups.

Figure 3.5 – Partial Subject Tree



[HI15] To test a given goal for inconsistency, first compare it against other goals in the same node, then against all goals in all sibling nodes, and finally against all goals in all ancestor nodes. Ask the following questions to test each goal pair for inconsistency.

- a) *Is there any way, including special circumstances, that a failure of the first goal could cause a failure of the second goal?*

If “yes”, then an inconsistency exists. Apply the heuristics in Section 3.3.2. Otherwise, continue with question b).

b) *Is there any way, including special circumstances, that a failure of the second goal could cause a failure of the first goal?*

If “yes”, then an inconsistency exists. Apply the heuristics in Section 3.3.2. Otherwise continue testing.

[HII6] Should a goal be modified during inconsistency identification or resolution, the analyst must revisit the subject tree, comparing the goal to goals in the same node, goals in all sibling nodes, goals in all ancestor nodes, and goals in all descendant nodes to ensure that the change has not created an inconsistency.

3.3.2 Inconsistency Resolution Heuristics

There are four heuristics for resolving inconsistency among goals:

[HIR1] If two goals are inconsistent, attempt to add a modifier for either goal that expresses the situations in which the goal is appropriate.

Example 3.4: Consider the goals in Table 3.3. Without the modifier in G_{18} , it appears that demographic information should be handled in two ways: it should be entered, and it should be defaulted. Upon examination, the goal-driven analysis makes it apparent that information is defaulted only when it is not entered. Therefore, the addition of the modifier mitigates the inconsistency.

Table 3.3 – Example of a Modifier that Mitigates Inconsistency

Goal Number	Keyword	Subject	Predicate	Modifier
G_4	ACHIEVE	Demographic information	Entered	
G_{18}	MAKE	Demographic information	Defaulted	when not entered

[HIR2] If two goals appear inconsistent, review each goal component against the originally elicited information to ensure that one of the goals has not been misstated, or ambiguously stated. If misstated or ambiguously stated, restate the goal as in Example 3.4.

Example 3.5: Consider the goals in Tables 3.4 and 3.5. G_{61} and G_{67} were originally stated such that they were ambiguous. Once the goals were restated, the inconsistency was mitigated.

Table 3.4 – Two Inconsistent EBS Goals

Goal Number	Keyword	Subject	Modifier
G_{61}	RESTRICT	access to survey questions	after completion
G_{67}	ALLOW	survey access	to new faculty

Table 3.5 – EBS Goal Restatement to Mitigate Inconsistency

Goal Number	Keyword	Subject	Modifier
G_{61}	RESTRICT	completion access to survey questions	after completion
G_{67}	ALLOW	inquiry access to survey questions	to new faculty

[HIR3] Given a broad goal that is inconsistent with another goal, attempt to decompose the goal into subgoals. The decomposition differentiates the issues addressed in the general goal, and allows the inconsistency to be directly addressed.

Example 3.6: Table 3.6 shows a situation in which one goal is inconsistent only in part with another goal. G_4 and G_{35} are deemed inconsistent since the types of information gathered for demographic information are not specified. Age, for example, may be relevant data since people’s health care choices vary with their age. However, information such as a person’s religion would not be relevant to a health care information system. By decomposing G_4 into

subgoals such as (ACHIEVE \wedge age \wedge entered), the information to be gathered as demographic information is defined. G_{35} may now be compared with each subgoal of G_4 . If no inconsistency exists with the subgoals to G_4 , then no inconsistency exists with G_4 , and the inconsistency is mitigated.

Table 3.6 – Goal Decomposition to Mitigate Inconsistency

Goal Number	Keyword	Subject	Predicate
G_4	ACHIEVE	demographic information	Entered
G_{35}	IGNORE	Irrelevant information	

[HIR4] Given two inconsistent goals that cannot be decomposed, the introduction of a new goal may be necessary to mitigate the inconsistency.

Example 3.7: Inconsistency can be mitigated by introducing a new goal, as shown in Tables 3.7. While G_7 and G_{27} are not always inconsistent, there is a situation where the plan population is small enough that respondent’s opinions can be identified. For example, if only one person in a certain age group had responded to the survey, a new faculty checking the survey knew that person had responded, and the survey indicated the number of people that responded for the group, then the new faculty member would know the respondents survey opinions, thereby violating G_{27} . By creating the new goal CONVEYED \wedge no number of responses \wedge to user, the inconsistency is mitigated. The new faculty will now only see an average of all responses, and will not know how many respondents are included in the average. The inconsistency is therefore mitigated.

Table 3.7 – Addition of a Goal to Mitigate Inconsistency

Goal Number	Keyword	Subject	Modifier
G ₇	KNOW	health plan survey information	in relation to each plan
G ₂₇	ENSURE	health plan survey information	confidential
NEW	CONVEYED	no number of responses	to user

3.4 Summary

This chapter presented techniques and heuristics for inconsistency identification and resolution. Chapter 4 will address how these techniques and heuristics were validated during the analysis of a commercial e-commerce system.

Chapter 4

Electronic Commerce System Case Study

Knowledge of the world is only to be acquired in the world, and not in a closet.

Lord Chesterfield

This chapter discusses the validation of the set of techniques and heuristics for inconsistency identification and resolution during goal-driven requirements engineering presented in Chapter 3. These techniques and heuristics were developed and validated on real problems. The first case study, the Employee Benefit System (EBS) discussed in Chapter 2, was formative in that it facilitated the development of a systematic approach to inconsistency identification and resolution. This approach was subsequently applied to a second system, the Electronic Commerce System (ECS), as described in this chapter; this analysis enabled the evaluation and refinement of the inconsistency identification and resolution strategies.

The ECS case study involved revisiting the analysis and design of a proprietary system designed to facilitate the quotation and submission of production orders for a large multi-national corporation. Data from the actual development effort was analyzed in an attempt to produce better strategies than could have been gathered from analyzing contrived examples and generalizing their results. The Goal-Based

Requirements Analysis Method (GBRAM) [Ant96, Ant97, AP98a] was applied to the problem in an effort to specify the system requirements. This GBRAM-based analysis entailed the identification, elaboration and refinement of goals into operational requirements. Since the GBRAM provides only limited guidance for conflict identification and resolution, the ECS goal set was further analyzed, using the heuristics presented in Chapter 3 to actively identify and resolve any inconsistencies among the goals. Section 4.1 provides an overview of the Electronic Commerce System analysis effort, and the evolution of our techniques and heuristics is discussed in Section 4.2.

4.1 Electronic Commerce System

The ECS GBRAM-based analysis enabled our team of analysts to specify the requirements for an electronic commerce, company-wide Intranet system to manage the quotation and ordering process for product sales. The company with which we collaborated on this effort has numerous plants throughout Europe that produce a variety of electrical products. The company maintains its own sales force, whose members provide quotations for product pricing and place orders for customers. Existing processes for providing quotations or ordering products were numerous and ad hoc, with each sales person and/or each plant having a different process, using various computer programs, printed catalogs or direct sales persons as plant contacts. A new, more tightly integrated system was needed to facilitate the provision of a consistent lowest price for each quote or order, as well as to streamline the bidding process to aid the company's distributed sales force. Specifically, the new system

must produce consistent quotations and order prices while tracking statistical information so that important market trends may be revealed.

The requirements engineering efforts for this project differed from the EBS case study. In the EBS case study, our process for defining requirements began with interviewing stakeholders, whereas the ECS case study began with analyzing the company's requirements specification (a collection of use cases). The use case information was supplemented by follow-up questions to the system stakeholders. Our analysis began as an effort to derive goals from the existing set of use cases. Our analysis progressed by annotating each goal with relevant auxiliary notes including agents, constraints, pre- and post-conditions, scenarios and questions as well as answers provided by various stakeholders during follow-up interviews. Since we lacked tool support to record analysis artifacts (e.g., goals, agents, constraints, etc.), we once again employed Microsoft Excel to produce a spreadsheet similar to those produced in previous case studies [Ant97, AP98a]. The analysis effort lasted approximately two months. During this period, a team of three analysts met once a week for sessions ranging from one to three hours in duration. Prior to each weekly meeting, individual analysts performed a goal analysis of assigned use cases that we then discussed and revised while collaboratively recording all goals and auxiliary notes. Our meeting preparations allowed us to concentrate primarily on revising and extending the original analyses during our meeting sessions. These sessions resulted in the identification of 292 goals. Of these goals, 96 were found to be redundant, and 8 to be synonymous, and 70 to be implementation-specific, GUI goals (a result of the quality of the use cases [ADS00]). Additionally, 12 new goals were identified. The

resulting goal set totaled 130 goals.

Following the heuristics presented in Chapter 3, the captured goals were translated using the syntax, a subject tree was constructed, and sample goal comparisons were performed to validate our techniques and heuristics for identifying and resolving inconsistencies. In addition to our validation efforts, observations were made during this case study that were not initially observed in the EBS case study. These observations are detailed in Section 4.2.

4.2 Validation of Lessons, Techniques and Heuristics

This section revisits the lessons and heuristics presented in previous chapters by discussing their applicability to the Electronic Commerce System case study. Our motivation was both to validate the heuristics as well as to examine the scalability of the techniques and heuristics on a larger system analysis effort. Interestingly, while this case study did serve to validate many of the heuristics (All heuristics were validated with the exception of HIR3 and HIR4 since these specific types of inconsistencies were not observed in the ECS case study.) in Chapter 3, scalability still remains to be more thoroughly investigated as addressed in the future work discussion in Chapter 5. Section 4.2.1 discusses the observations and examples from the ECS case study that correspond to our observations in the EBS case study.

4.2.1 Lessons Learned and Heuristics Revisited

The following lessons learned from the EBS case study were also observed in the ECS case study. The lessons and an explanation of applicability to this work are

provided below.

A syntax is helpful in structuring goals for comparison, while allowing a natural expression.

The ECS case study supported the belief that the structuring of goals according to a syntax both clarifies the meaning of each goal, and simplifies goal comparisons. Additionally, the use of the syntax defined in Chapter 3 was neither cumbersome nor cryptic. Consider goal G_{212} originally expressed as `MAKE customer identified`. This goal seemed unusual given that according to our limited domain knowledge, it is not the system's responsibility to identify a customer, but instead that of the user. Upon reviewing the supporting goal data, it became apparent that the purpose of the goal was to associate a customer with each quote. Therefore, the goal was rephrased to `MAKE customer associated with quote`, a more accurate statement of the goal. This is just one example of many in which the goal syntax presented in Chapter 3 yielded a more meaningful set of goals, in which each goal was further elaborated and refined.

Constructing a goal subject tree helps surface goal inconsistencies.

In the ECS, two inconsistencies were identified using the techniques described in Chapter 3, whereas the GBRAM analysis identified no inconsistency. Consider goals G_{159} (`ACHIEVE duplication notice sent`) and G_{191} (`MAKE duplication notice mailed`). Since these goals assign the same task (the

sending of a duplication notice) to two agents, the user and the system, there is an inconsistent assignment of responsibility. As these goals shared the same node in the subject tree, the inconsistency was identified during goal comparison, as prescribed in heuristic HIII5. These inconsistencies were overlooked during the construction of a goal hierarchy in the original goal analysis; these goals would not have been identified had we not employed a goal subject tree.

Pair wise comparison of goals is a thorough method of finding inconsistency so that it may be resolved. However, the use of goal subject relationships reduces the work involved in inconsistency identification.

The reduction in the number of comparisons required to identify inconsistency, using the techniques and heuristics of Chapter 3, is evident when calculating the number of comparisons necessary for one goal using a pairwise comparison technique versus the number of comparisons necessary using the goal subject tree technique. Consider goal G_{164} (ACHIEVE country of end use entered), one of 130 goals in the ECS case study. Using a pairwise comparison technique, this goal must be compared against 129 other goals; in contrast, only 62 comparisons is required using the approach presented in Chapter 3 -- less than half the number previously required. Thus, we observe that the use of subject relationships significantly reduces the work involved in inconsistency identification.

The use of a goal syntax aids in the identification of synonymous goals.

Six synonymous goal pairs were identified during the ECS case study. Given that these six synonymous goals were not initially identified during our prior GBRAM-based analysis, this finding served to validate the usefulness of our goal syntax. Thus, the identification of these synonymous goals supports our contention that the new goal syntax provides enhanced support for the identification of synonymous and redundant goals for the GBRAM.

The inconsistency identification heuristics discussed in Chapter 3 provide analysts with prescriptive guidance for constructing a goal subject tree and were successfully applied in the ECS case study. Since applying the inconsistency identification heuristics was very straightforward, we instead provide a more interesting discussion of our experiences with the resolution heuristics. Examples are provided to demonstrate their applicability in the ECS case study.

[HIR1] If two goals are inconsistent, attempt to add a modifier for either goal that expresses the situations in which the goal is appropriate.

Example 4.1: The two goals G_{48} and G_{52} , listed in Table 4.1, are inconsistent since they prescribe that the line item status must be marked in two inconsistent ways. Adding the constraints under which each goal is applicable to the goal modifier mitigates this inconsistency. For example, adding the phrase `if all line item information has been entered` to the modifier in G_{48} , and the phrase `if all line item information has`

not been entered to the modifier G_{52} clarifies the situations in which each goal is applicable as shown in Table 4.2 (note that the ' symbol indicates a modified goal), thereby resolving the inconsistency.

Table 4.1 – Two Inconsistent ECS Goals

Goal Number	Keyword	Subject	Predicate	Modifier
G_{48}	MAKE	line item status	marked	complete
G_{52}	MAKE	line item status	marked	incomplete

Table 4.2 – Example of a Modifier That Mitigates Inconsistency

Goal Number	Keyword	Subject	Predicate	Modifier
G_{48}'	MAKE	line item status	Marked	Complete if all line item information has been entered
G_{52}'	MAKE	line item status	Marked	Incomplete if all line item information has not been entered

[HIR2] If two goals appear inconsistent, review each goal component against the originally elicited information to ensure that one of the goals has not been misstated, or ambiguously stated. If misstated or ambiguously stated, restate the goal as in Example 3.4.

Example 4.2: Consider goals G_{159} and G_{191} in Table 4.3. The two goals are inconsistent given that a responsibility has been allocated to two agents simultaneously. We determined from our source material that the duplication notice was to be initiated by the user, but sent by the system. Therefore, goal G_{159} was stated ambiguously. Table 4.3 shows a restatement of goal G_{159} that mitigates the inconsistency by refining G_{159} to accurately express the actions of the responsible agent.

Table 4.3 – Two Inconsistent ECS Goals

Goal Number	Keyword	Subject	Predicate
G ₁₅₉	ACHIEVE	duplication notice	sent
G ₁₉₁	MAKE	duplication notice	mailed

Table 4.4 – ECS Goal Restatement to Mitigate Inconsistency

Goal Number	Keyword	Subject	Predicate
G ₁₅₉	ACHIEVE	duplication notification	initiated
G ₁₆₁	MAKE	duplication notice	mailed

4.2.2 New Lessons Learned and Heuristics

In addition to validating our heuristics, the ECS case study also yielded additional lessons learned which led to the creation of new heuristics. The three lessons learned are listed and discussed below.

Lack of contextual information leads to goal misinterpretation.

Lack of contextual information increases the risk that goals may be misinterpreted. Context makes goals more meaningful and when adequate context is not provided, valuable information about the goal may be lost. Consider the goals G₁₅₉ (ACHIEVE duplication notice sent) and G₁₉₁ (MAKE duplication notice mailed). These goals appear to be inconsistent. However, upon examination of the source material, G₁₅₉ was modified to (ACHIEVE duplication notification initiated) to more accurately reflect the role of the user in the notification process. Here the importance of a syntax is demonstrated. By stating goals according to the

syntax, and by defining each term used, the context of the goal is preserved, thus leading to a correct interpretation of the goal.

It is important to consider multiple viewpoints when analyzing a system.

In the ECS case study, each use case analyzed was written using the viewpoint of only one stakeholder, that of the use case author. Since the scenarios were developed by the authors and were not validated by system users, we suspect the scenarios may be plagued due to assumptions made about typical and/or expected user and system interactions. This lack of validation is problematic during product design and development [Dav93]. The fact that the authors only investigated scenarios from one actor's viewpoint also introduces risk, since exploring multiple viewpoints yields more comprehensive coverage of the requirements.

Reusable goal classes help uncover missing goals.

By considering a goal with respect to different goal classes, an analyst may uncover hidden goals. In the ECS case study, each goal was considered from the viewpoint of both the user and the system, resulting in the identification of new goals. The notion of reusable goal classes that occur in various types of software systems is discussed in [Ant97]. Of particular relevance to this case study are the goals for an electronic commerce application, classified according to subject matter, as reported in [AP98a].

These electronic commerce goal classes are:

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

Process support goals describe goals that pertain to system processes enacted by the user or the system. Electronic commerce goals deal with the base functionality of the system. Information display and organization goals describe the organization and presentation of information by the system. Finally, security and access control goals describe those goals involved in limiting access to authorized users [AP98a]. It should be noted that these categories are not mutually exclusive. The same goal can be, and often is, classified according to more than of these goal classes.

As reported in [ADS00], one should expect to have all four of these goal classes represented in the goal set for any electronic commerce application. In this study, 120 process support goals were identified. Since the authors focused heavily on the system's GUI during requirements specification, the large number of process goals is not surprising. The process goals are natural products of this type of problem decomposition. Each goal from a user action translated into a process goal. Additionally, each system response from a user action translated into a different process goal. We also identified 34 information and organization goals that were, typically, buried in the pre- and post-conditions of certain scenarios. As expected, we identified a

fair number of electronic commerce goals; the 73 electronic commerce goals emphasized such items as quotes, shopping carts, product ordering, notifications and confirmations. The non-electronic commerce goals described generic system functions that would be expected in any system.

During this analysis effort, consideration of reusable goal classes aided us in determining which types of users have what kinds of access to the system. There were only 8 security and access goals identified and these solely considered the mechanism for user login. This observation suggests that the availability of goal classes can indeed be very beneficial when developing the requirements for systems since the goal classes can help ensure that all expected behaviors have been considered for the given system. Upon realizing that we had not derived a sufficient number of security and access control goals, we were able to further analyze the various kinds of system users so that the access levels could be defined.

These lessons learned contributed to the specification of three additional heuristics during the ECS case study. One of these heuristics concerns inconsistency identification (HII), while the other two concern requirements coverage (HRC). Requirements coverage aids analysts in estimating the current level of requirements coverage for the system. These heuristics are listed and discussed below.

[HII7] The greater the number of stakeholders involved in a project, the greater the number of viewpoints represented, and therefore the greater the potential for inconsistency.

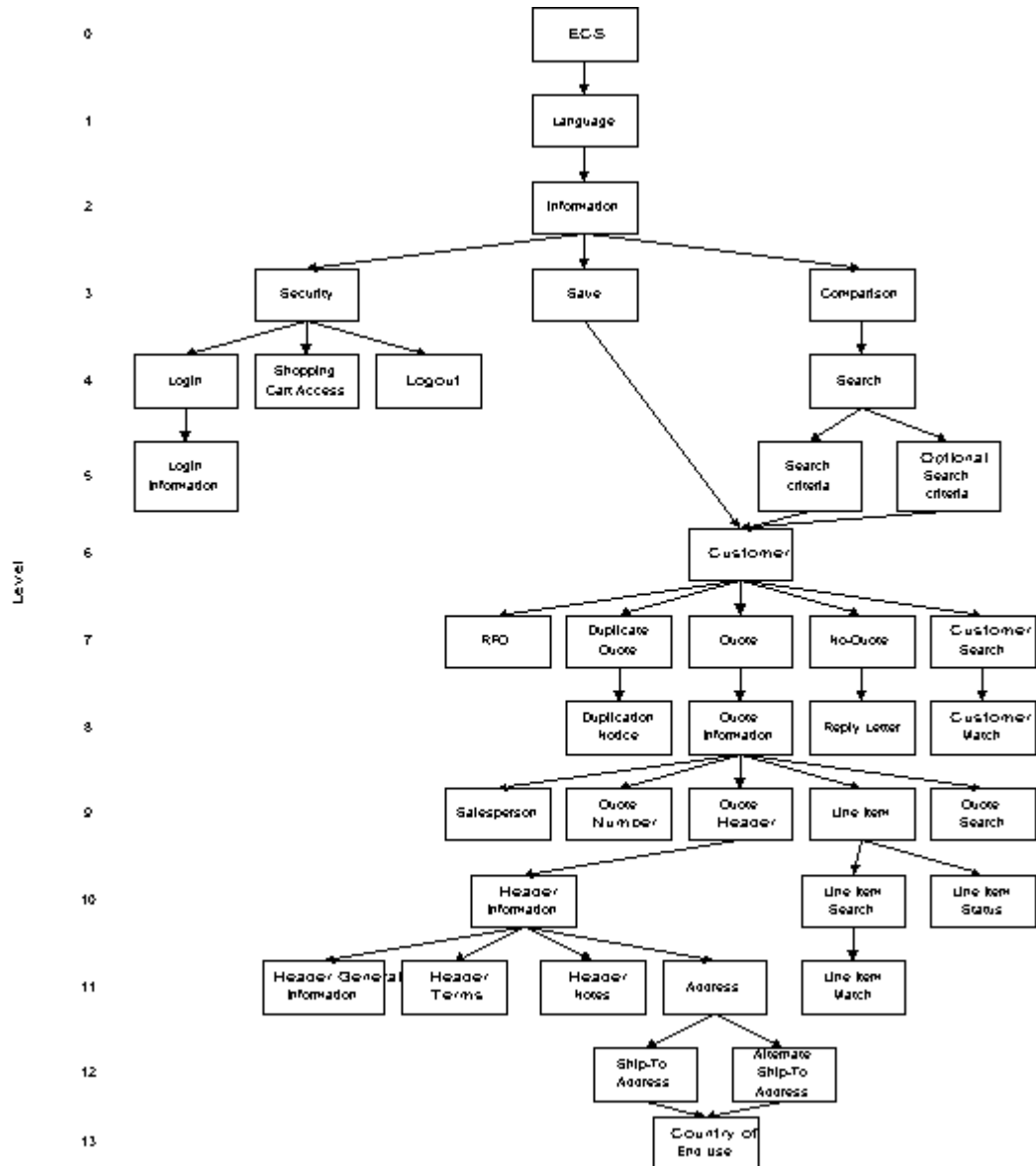
Fewer inconsistencies were observed in the ECS case study than in the EBS case study; this is perhaps due to several factors. Since the use case authors did not consult stakeholders during the creation of the use cases, only a small number of stakeholders were represented in the ECS. Effectively only one stakeholder was represented in connection with each use case. In contrast, we were able to gather the viewpoints of multiple stakeholders for each issue addressed in the EBS, yielding a proportionally larger number of inconsistencies. Analysts should thus proactively seek to gather multiple viewpoints to ensure broader viewpoint coverage during the requirements process. While this may seem to contradict our efforts to reduce inconsistency, it is clearly more dangerous to be lazy and ignore the viewpoints of all stakeholders simply to avoid the potential for inconsistency, since inconsistency surfaces possible implementation alternatives.

[HRC1] A severe imbalance in the branches of a subject tree may indicate poor requirements coverage. Branches that appear underdeveloped should be reviewed to ensure that the subjects contained in those branches have been fully elaborated.

Example 4.3: Consider the subject tree in Figure 4.1. It is clear that the “quote operations” (levels 7-13 of the tree) have been thoroughly analyzed as evidenced by the number of subjects listed under the quote subject. However, the few subjects listed under the “security” subject (see leftmost subtree), clearly a critical aspect of any system, signals that little attention has been given the system’s security

concerns. The imbalance in the subject tree is indicative of the need for further elaboration of security issues. Analysts should thus look for such indications of poor requirements coverage and further elaborate the corresponding goal set.

Figure 4.1 – Subject Tree for the ECS Case Study



[HRC2] If an analyst encounters difficulty placing a subject in the subject tree, it may be an indicator that a higher-level concept is missing from the tree. Add this missing subject to the tree. The following two questions may be asked to assist an analyst in identifying the higher-level concept:

b) *Is there a goal class that should be represented as a subject in the subject tree?*

If so, insert the goal class into the subject tree.

c) *Is there another system concept that should be represented as a subject in the subject tree?*

If so, insert the new subject into the subject tree.

Example 4.4: The leftmost subtree in Figure 4.1 (*security*) shows only three subject “children”. During construction of this tree, we were troubled by the placement of the following three “orphaned” subjects: `login`, `logout`, and `shopping cart access`. These subjects were all related to security, and security as a high level subject was not represented in the subject tree. By applying HRC2, we added security to the tree. After a subject is added to the tree, the following two steps should be followed:

1. An analyst should revisit the GBRAM goal elaboration heuristics to determine if there are goals relating to this new subject.
2. An analyst should review the subject tree to determine if any of the reusable goal classes not represented in the tree should be added.

4.3 Summary

This chapter describes the Electronic Commerce System (ECS) case study. First the system and the context in which it was developed was discussed. Subsequently, validation of the lessons learned in Chapter 2 and the heuristics in Chapter 3 was discussed. Finally, new heuristics developed from the ECS case study were presented. Chapter 5 discusses the conclusions we have drawn from our research.

Chapter 5

Conclusions and Future Work

Nothing great was ever achieved without enthusiasm.

Ralph Waldo Emerson

This thesis describes a set of techniques and heuristics to identify and resolve inconsistency during goal-driven requirements analysis. Chapter 1 introduces the motivation of our thesis, and discusses related research concerning inconsistency identification and resolution. Chapter 2 discusses the analysis of the Employee Benefit System (EBS) case study; the EBS case study served as the basis for the definition of our techniques and heuristics. Chapter 3 defines a goal syntax, and presents inconsistency identification and resolution techniques and heuristics in detail. Chapter 4 discusses the analysis of the Electronic Commerce System (ECS) case study; this case study serves as validation of our research, and led to the evolution of our set of heuristics.

Section 5.1 lists the demonstrated experiences and primary contributions of the thesis. Section 5.2 discusses future work that may be pursued from our research.

5.1 Summary of Demonstrated Experiences and Contributions

The experiences reported in this thesis demonstrate that:

- it is possible to identify and resolve inconsistencies among goals using the heuristics and techniques presented in this thesis;
- using the subject tree approach, goal inconsistencies may be identified more efficiently than with pairwise comparisons which are both cumbersome and time consuming;
- the heuristics and associated recurring questions can be applied in conjunction with those in the Goal-Based Requirements Analysis Method, offering guidance to analysts as they apply the approach; and
- a structured goal syntax offers an effective structure for imposing an initial level of consistency within a goal set for further goal elaboration and refinement.

The primary contributions of this thesis are:

- a structured goal syntax to aid analysts in forming goals for future analysis;
- three sets of heuristics (inconsistency identification, inconsistency resolution, and requirements coverage) to aid analysts in identifying and resolving goal inconsistencies;
- a predefined, extensible set of goal keywords;
- a comparison strategy for identifying inconsistency; and
- techniques that augment those in the Goal-Based Requirements Analysis Method which provide additional prescriptive guidance to analysts for the initial identification and resolution of conflicts and inconsistencies.

5.2 Future Work

Recall that goals must be identified prior to the application of our techniques and heuristics. Some researchers [EN96] see this precondition as an opportunity to

identify implementation alternatives within the requirements; other researchers see this precondition as a limitation. Additionally, subject matter relationships are subjective, and therefore the process of placing subjects in the subject tree may not result in the same subject tree for every analyst. These two examples demonstrate that there are issues surrounding our research that must be addressed in the future. The following paragraphs discuss research opportunities born from this research.

Future plans include further application of these heuristics to a large industrial-sized analysis effort. Our research to date has been conducted in an academic setting, which provided a stable environment for concepts to be identified and proven. However, the increased complexity of an industrial environment would serve to further validate the usefulness of our techniques and heuristics. Additionally, information concerning the scalability of our techniques and heuristics will be quantified on an actual development effort once the SMaRT (Scenario Management and Requirements Tool) is fully implemented.

Unrecognized relationships exist between inconsistent goals that warrant further exploration. The concept of similarity measures defined in [AAB99] is used to calculate the similarity between scenarios. Similarly, this logic can apply to goals. As we have stated earlier in this thesis, goals that are inconsistent must be related by at least one goal component, the goal subject. Similarly, goals that are redundant have four common goal components. Considering the examples we have shown that deal with inconsistency contained in the modifier portion of a goal (see HIR1), it reasons that there is a threshold in the range $[.25 \leq X < 1]$ that identifies goals that are likely inconsistent.

The following formulas define a similarity measure for goals stated according to our syntax. Consider the list of components for two goal types (functional and non-functional):

$$G_1: \{\text{keyword, subject, predicate, modifier}\};$$
$$G_2: \{\text{keyword, subject, modifier}\}.$$

Further, define G as either a goal of type G_1 or of type G_2 . A similarity measure is calculated for two goals of type G , g_1 and g_2 , as:

$$\frac{2 * (|g_1 \cap g_2|)}{(|g_1| + |g_2|)}$$

Currently this concept of similarity measures for scenarios is being implemented in SMaRT. Analogous measures for analyzing goal inconsistencies are planned for the tool as well.

Finally, the techniques and heuristics in this thesis have been designed such that they lend themselves to automation as is currently planned. The syntax described in Chapter 3 decomposes goals into distinct components (keyword, subject, predicate, and modifier). The implementation of a tool to track the set of instances of each component, and to associate each component instance with a definition is not a difficult task. Additionally, links could be provided for related subjects such that the subject tree is created during goal elicitation and elaboration. With such a tool, GBRAM information could be stored such that it could be reviewed immediately should an inconsistency be found.

5.3 Summary

Our research addresses subject matter relationships between goal subjects. These subjects translate directly to operational requirements as prescribed in the GBRAM. Thus, it follows that the techniques and heuristics presented herein are easily modifiable such that they could be used in any requirements engineering effort that is not goal-driven; the subject matter relationships could be derived from requirements instead of goals, thus becoming generalizable to various requirements engineering methodologies. This research demonstrates a more efficient approach for identifying and resolving inconsistency than previously offered during the early stages of goal analysis for requirements specification.

Bibliography

- [AAB99] T. Alspaugh, A.I. Antón, T. Barnes and B. Mott. An Integrated Scenario Management Strategy, *IEEE Fourth International Symposium on Requirements Engineering (RE'99)*, University of Limerick, Ireland, pp. 142-149, 7-11 June 1999.
- [AD93] J.S. Anderson and B. Durney. Using Scenarios in Deficiency-driven Requirements Engineering. *IEEE International Symposium on Requirements Engineering (RE'93)*, San Diego, CA, pp. 134-41, January 1993.
- [ADS00] A.I. Antón, J.H. Dempster and D.F. Siege. Deriving Goals from a Use Case Based Requirements Specification for an Electronic Commerce System, *Accepted to the Sixth Intl. Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ)*, Stockholm, Sweden, 5-6 June 2000.
- [Ant96] A.I. Antón. Goal-Based Requirements Analysis, *Second IEEE International Conference on Requirements Engineering (ICRE '96)*, Colorado Springs, Colorado, pp. 136-144, 15-18 April 1996.
- [Ant97] A. I. Antón. *Goal Identification and Refinement in the Specification of Software-Based Information Systems*, Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA, 1997.
- [AMP94] A.I. Antón, W.M. McCracken and C. Potts. Goal Decomposition and Scenario Analysis in Business Process Reengineering, *Advanced Information System Engineering: 6th International Conference, CAiSE '94 Proceedings*, Utrecht, The Netherlands, pp. 94-104, 6-10 June 1994.
- [AP98a] A.I. Antón and C. Potts. The Use of Goals to Surface Requirements for Evolving Systems, *IEEE International Conference on Software Engineering (ICSE '98)*, Kyoto, Japan, pp. 157-166, 19-25 April 1998.
- [AP98b] A.I. Antón and Colin Potts. A Representational Framework for Scenarios of Systems Use. *Requirements Engineering Journal*, Springer-Verlag, 3(3-4), pp. 219-241, 1998.
- [BFJ92] K. Benner, M.S. Feather, W.L. Johnson and L. Zorman. Utilizing Scenarios in the Software Development Process. In *IFIP WG 8.1 Working Conference on Information Systems Development Processes*, December 1992.

- [BI96] Boehm, B. and H. In. Identifying Quality Requirement Conflicts, *IEEE Software*, pp. 25-35, March 1996.
- [Boo94] G.Booch. *Object-Oriented Analysis & Design with Applications* (2nd edition). Benjamin/Cummings 1994.
- [Dav93] Davis, A.M. *Software Requirements: Objects, Functions, & States*,. Prentice-Hall, 1993.
- [DvL93] A. Dardenne, A. van Lamsweerde and S. Fickas. Goal-Directed Requirements Acquisition. *Science of Computer Programming*, 201(1-2), pp. 3-150, April 1993.
- [Eas91] S. M. Easterbrook. *Elicitation of Requirements from Multiple Perspectives*. Ph.D. Thesis, University of London, London, UK, 1991.
- [Eas92] S. M. Easterbrook. Resolving Requirements Conflicts with Computer-Supported Negotiation. *Workshop on Requirements Engineering*, Oxford, 1992.
- [Eas93a] S. M. Easterbrook. Domain Modeling with Hierarchies of Alternative Viewpoints. *Proceedings, First IEEE International Symposium on Requirements Engineering (RE'93)*, San Diego, pp. 65-72, January 1993.
- [Eas93b] S. M. Easterbrook. Negotiation and the Role of the Requirements Specification. In P. Quintas (ed) *Social Dimensions of Systems Engineering: People, Processes, Policies and Software Development*, London: Ellis Horwood, pp. 144-164, 1993.
- [EFKN94] S. M. Easterbrook, A. C. W. Finkelstein, J. Kramer, and B. A. Nuseibeh. Co-ordinating Conflicting ViewPoints by Managing Inconsistency. *Workshop on Conflict Management in Design, International Conference on Artificial Intelligence in Design*, Lausanne, Switzerland, 15-18 August 1994.
- [EN95] S. M. Easterbrook and B. A. Nuseibeh. Managing Inconsistencies in an Evolving Specification. *Proceedings, Second IEEE International Symposium on Requirements Engineering (RE'95)*, York, UK, pp. 48-55, April 1995.
- [EN96] S. M. Easterbrook and B. A. Nuseibeh. Using ViewPoints for Inconsistency Management. *Software Engineering Journal*, 11(1), pp. 31-43, January 1996.
- [Eas96b] S. M. Easterbrook. Learning from Inconsistency, *Proceedings, Eighth International Workshop on Software Specification and Design (IWSSD-8)*, Paderborn, Germany, pp.136-140, Mar. 22-23, 1996.
- [FF89] A. Finkelstein and Hugo Fuks. Multi-Party Specification. In *Proceedings of the 5th Intl. Workshop on Software Specification and Design*, pp. 19-20, Pittsburgh, PA, May 1989.
- [FKG89] A. Finkelstein, J. Kramer, and M. Goedicke. Viewpoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering. In *Algebraic Methods II: Theory, Tools & Applications*, Mierlo, Netherlands, pp. 29-54, September 1986.

- [FKN92] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., & Goedicke, M. ViewPoints: A Framework for Integrating Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1), pp. 31-57, 1992.
- [HJL96] Heitmeyer, C., Jeffords, R. & Labaw, B. Automated Consistency Checking of Requirements Specifications. *ACM Transactions on Software Engineering and Methodology*, 5(3), pp. 231-261, July 1996.
- [HLK95] C. Heitmeyer, B. Labaw, and D. Kiskis. Consistency Checking of SCR-Style Requirements Specifications. *2nd IEEE International Symposium on Requirements Engineering (RE'95)*, York UK, pp. 56-63, March 1995.
- [Jac95] Michael Jackson. *Software Requirements and Specifications*. Addison-Wesley, 1995.
- [JBC98] Jarke, M., X.T. Bui and J.M. Carroll. Scenario Management: An Interdisciplinary Approach *Requirements Engineering Journal*, Springer Verlag, 3(3-4), pp. 154-173, 1998.
- [Kar96] Karlson, J. Software Requirements Prioritizing. *Proc. 2nd IEEE International Conference on Requirements Engineering (ICRE '96)*, pp. 110-116, Colorado Springs, Colorado, USA, April 1996.
- [KM92] Kirsznner, Laurie G., and Stephen R. Mandell. *The Holt Handbook*. New York: Holt, Rhinehart and Winston, Inc., 1992.
- [KR96] Karlson, J. and K. Ryan. Prioritizing Software Requirements in an Industrial Setting. *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2, pp. 953-958, 1996.
- [Lev00] N.G. Leveson. Intent Specifications: An Approach to Building Human-Centered Specifications. *IEEE Transactions on Software Engineering*, 26(1), pp. 15-35, January 2000.
- [LF91] J. C. S. P. Leite and P. A. Freeman. Requirements Validation Through Viewpoint Resolution. *IEEE Transactions on Software Engineering*, 17(12), pp. 1253-69, December 1991.
- [LS99] E.C. Lupu, M. Sloman, Conflicts in Policy-Based Distributed Systems Management. *IEEE Transactions on Software Engineering*, 25(6), pp. 852-869, November/December 1999
- [Mai98] N. Maiden, S. Minocha, K. Manning and M. Ryan. CREWS-SAVRE: Systematic Scenario Generation and Use. *International Conference on Requirements Engineering*, pp. 148-155, April 1998.
- [MMM98] Maiden, N., S. Minocha, K. Manning and M. Ryan. CREWS-SAVRE: Systematic Scenario Generation and Use, *IEEE Int'l. Conf. on Requirements Engineering (ICRE'98)*, pp. 148-155, April 1998.

- [MS96] N. Maiden and A. Sutcliffe. Analogical Retrieval in Reuse-Oriented Requirements Engineering. *Software Engineering Journal*, 11(5):281-292, September 1996.
- [MvL97] P. Massonet and A. van Lamsweerde. Analogical Reuse of Requirements Frameworks. *3rd IEEE International Symposium on Requirements Engineering*, pp. 26-37, January 1997.
- [MNJ97] R. Motschnig, H. W. Nissen, and M. Jarke. View-Directed Requirements Engineering: A Framework and Metamodel. *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering (SEKE '97)*, Madrid, Spain, 17-20 June 1997.
- [NE99] B. A. Nuseibeh and S. M. Easterbrook, The Process of Inconsistency Management: a Framework for Understanding, *Proceedings, First International Workshop on the Requirements Engineering Process (REP'99)*, Florence, Italy, September 2-3, 1999.
- [NKF94] 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.
- [Pay00] Principal Investigator, Dr. Fay Cobb Payton, 1999-2000, North Carolina State University Faculty Development and Research Grant, "Using Inquiry-Based Requirements Analysis to Develop A Health Care Prototype for New/Existing NCSU Employees"
- [PG00] Payton, F. C. and M. J. Ginzberg, 2000, Interorganizational Health Care Systems Implementations: An Exploratory Study of Early Electronic Commerce Initiatives, forthcoming Health Care Management Review.
- [PL00] Payton, F.C. and H. Lucas, 2000, The State of E-Health and Patient Information, Working Paper, North Carolina State University, College of Management
- [Pot93] C. Potts. Software Engineering Research Revisited. *IEEE Software*, pp. 19-28, September 1993.
- [Pot99] C. Potts. ScenIC: A Strategy for Inquiry-Driven Requirements Determination, *IEEE Fourth International Symposium on Requirements Engineering (RE'99)*, University of Limerick, Ireland, pp. 58-65, 7-11 June 1999.
- [PTA94] C. Potts, K. Takahashi and A.I. Antón. Inquiry-Based Requirements Analysis, *IEEE Software*, 11(2), pp. 21-32, March 1994.
- [RF94] W.N. Robinson, Fickas, S. Supporting Multi-Perspective Requirements Engineering. *IEEE First International Conference on Requirements Engineering*, pp. 206-215, 18-22 April 1994.
- [Rob96] W.N. Robinson, Automated Assistance for Conflict Resolution in Multiple Perspective Systems Analysis and Operation, *ACM Workshop on Viewpoints in Software Development, In Association with the ACM Symposium on Foundations of Software Engineering*, San Francisco, USA, 14-15 October 1996.

- [RP98] W.N. Robinson, S. Pawlowski, Surfacing Root Requirements Interactions from Inquiry Cycle Requirements, *IEEE International Conference on Requirements Engineering*, Colorado Springs, CO, pp. 82-89, April 6-10, 1998.
- [RP99] W.N. Robinson, S. Pawlowski, Managing Requirements Inconsistency with Development Goal Monitors. *IEEE Transactions on Software Engineering*, 25(6), pp. 816-835, November/December 1999.
- [RGK99] C. Rolland, G. Grosz, R. Kla. Experience With Goal-Scenario Coupling in Requirements Engineering. *IEEE Fourth International Symposium on Requirements Engineering (RE'99)*, University of Limerick, Ireland, pp. 74-81, 7-11 June 1999.
- [RSB98] C. Rolland, C. Souveyet and C.B. Achour. Guiding Goal Modeling Using Scenarios, *IEEE Trans. on Software Eng.*, 24(12), pp. 1055-1071, Dec. 1998.
- [VL98] van Lamsweerde, A. and E. Letier. Integrating Obstacles in Goal-Driven Requirements Engineering. *20th International Conference on Software Engineering*. Kyoto, Japan, IEEE Computer Society Press, 1998.
- [vLD98] van Lamsweerde, A., Darimont, R., Letier, E. Managing Conflicts in Goal-driven Requirements Engineering, *IEEE Transactions on Software Engineering*, 24(11), pp. 908-926, November 1998.
- [vLDM95] van Lamsweerde, A., Darimont, R. and Massonet, P. Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt, In *Proceedings 2nd International Symposium on Requirements Engineering (RE'95)*, York, UK, pp. 194-203, March 1995.
- [Web84] "Syntax." Webster's American Heritage Dictionary. 1984 ed.
- [WPJ98] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer. Scenarios in System Development: Current Practice. *IEEE Software*: 15(2), pp. 34-35, March 1998.
- [YL96] J. Yen and X.F. Liu. An Analytic Framework for Specifying and Analyzing Imprecise Requirements. *Proceedings of the Fifth IEEE International Conference*, Volume: 1, pp. 349-354 vol.1, 1996.
- [YT97] J. Yen and W.A. Tiao. A Systematic Tradeoff Analysis for Conflicting Imprecise Requirements. *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pp. 87 -96, 1997.

APPENDIX

Appendix A

Interview Summary Template

INTERVIEW

<Name of Interviewee>

<Date and Time of Interview>

Interviewer: <Name>

<Document File Name>

I interviewed <interviewee> in <his/her> office (<office location>). The <interviewee>'s comments are summarized below. Observations concerning similar subjects have been grouped together.

- <Comment>
- <Comment>
- <Comment>
- <Comment>
- ...

I drew the following conclusions from <interviewee>'s statements, although they were not specifically expressed in the interview.

- <Observation>
- <Observation>
- <Observation>
- <Observation>
-

