Exploring Alternatives During Requirements Analysis

Goal-oriented requirements analysis techniques focus on organizational as well as technical objectives, and provide means for refining such objectives so that you can more effectively explore alternatives during requirements definition. Once you select a set of alternatives to achieve these objectives, you can elaborate them during subsequent phases to make them more precise and complete.

John Mylopoulos, *University of Toronto*
Lawrence Chung, *University of Texas, Dallas*
Stephen S.Y. Liao and Huaiqing Wang, *City University of Hong Kong*
Eric Yu, *University of Toronto*

Traditionally, requirements analysis consisted of identifying relevant data and functions that a software system would support. The data to be handled by the system might be described in terms of entity-relationship diagrams, while the functions might be described in terms of data flows. Or you could use object-oriented analysis techniques that offer class, use case, state chart, sequence, and other diagrammatic notations for modeling.

While such techniques[1] form the foundation for many contemporary software engineering practices, requirements analysis has to involve more than understanding and modeling the functions, data, and interfaces for a new system. In addition, the requirements engineer needs to explore alternatives and evaluate their feasibility and desirability with respect to business goals.

For instance, suppose your task is to build a system to schedule meetings. First, you might want to explore whether the system should do most of the scheduling work or only record meetings. Then you might want to evaluate these requirements with respect to technical objectives (such as response time) and business objectives (such as meeting effectiveness, low costs, or system usability). Once you select an alternative to best meet overall objectives, you can further refine the meaning of terms such as "meeting," "participant," or "scheduling conflict." You can also define the basic functions the system will support.

The need to explore alternatives and evaluate them with respect to business objectives has led to research on goal-oriented analysis.[2,3] We argue here that goal-oriented analysis complements and strengthens traditional requirements analysis techniques by offering a means for capturing and evaluating alternative ways of meeting business goals.

The remainder of this article details the five main steps that comprise goal-oriented analysis. These steps include goal analysis, softgoal analysis, softgoal correlation analysis, goal correlation analysis, and evaluation of alter-

0740-7459/00/$10.00 © 2001 IEEE

natives. To illustrate the main elements of the proposed analysis technique, we explore a typical scenario that involves defining requirements for a meeting scheduling system.

Goal Analysis

Let's suppose that the meeting-scheduling task is a generic goal we want to achieve. We can then use AND/OR decompositions to explore alternative solutions. Each alternative reflects a potential plan for satisfying the goal. Figure 1 presents the decomposition of the Schedule Meeting goal. We mark the AND decompositions with an arc, which indicates that satisfying the goal can be accomplished by satisfying all subgoals. We mark the OR decompositions, on the other hand, with a double arc; these decompositions require only one of their subgoals to be satisfied.

In Figure 1, the goal Schedule Meeting is first AND-decomposed to two subgoals: Collect Constraints and Generate Schedule. The Generate Schedule subgoal is OR-decomposed into three subgoals that find a schedule manually, automatically (by the system being designed), or interactively. Other decompositions explore alternative ways of fetching the necessary information, including timetable information, which might or might not be publicly available for each potential participant. Even in this simple example, there are literally dozens of alternative solutions.

Generally, AND/OR diagrams systematize the exploration of alternatives within a space that can be very large. These diagrams make it possible to conduct a simple form of analysis. In particular, suppose we choose three leaf nodes—marked with a checkmark in Figure 1—as requirements for the system. With a simple, well-known algorithm, we can explore whether we've achieved the root goal or not. In particular, the algorithm propagates a check upwards if all the subgoals of an AND-goal are checked or one of the subgoals of an OR-goal is checked.

Softgoal Analysis

Unfortunately, basic goal analysis can only help us with goals that can be defined crisply, such as having or not having a meeting scheduled. Not all goals, however, can be so clearly delineated. Suppose we want to represent and analyze less clear-cut requirements such as "the system must be highly usable" or "the system must improve meeting quality." Such requirements, often called qualities or nonfunctional requirements, can't be defined generically nor can they have clear-cut criteria to determine whether they've been satisfied. For these requirements, we need a looser notion of goal and a richer set of relationships so that we can indicate, for example, that a goal supports or hinders another one without being limited to strict AND/OR relationships.

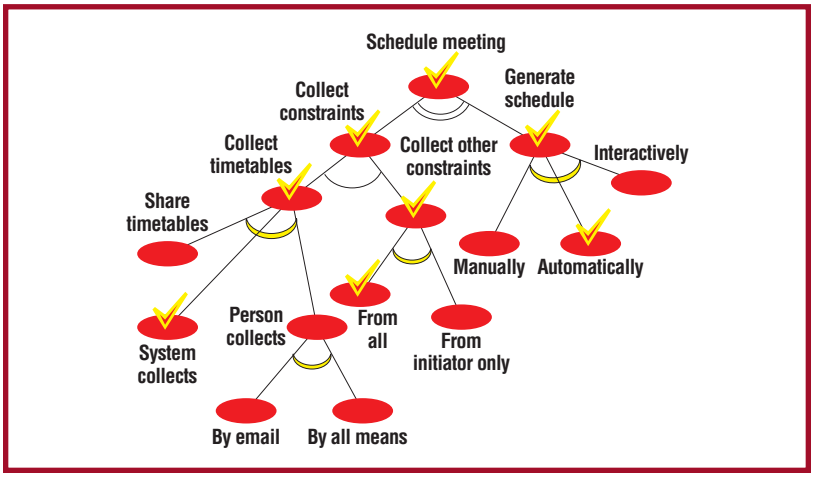

Figure 1. An AND/OR decomposition that illustrates alternatives for achieving the meeting-scheduling goal.

To model this looser notion of goal, we use the notion of *softgoal*, presented in detail in *Nonfunctional Requirements in Software Engineering*.[4] Softgoals represent ill-defined goals and their interdependencies. To distinguish softgoals from their hard goal cousins, we declarea softgoal as being satisfied when there is sufficient positive evidence and little negative evidence against it; a softgoal is unsatisfiable when there is sufficient negative and little positive evidence.

Figure 2 illustrates an example that focuses on the quality "highly usable system," which might be as important an objective as any of the functional goals encountered earlier. The softgoal Usability in Figure 2 represents this requirement. Analyzing this softgoal consists of iterative decompositions that involve AND/OR relationships or other more loosely defined dependency relations. Figure 2 shows a number of such relationships labeled with a + sign, which indicates that one softgoal supports—or "positively influences"—another.

In Figure 2, for instance, User Flexibility is clearly enhanced not only by the system quality Modularity, which allows for module substitutions, but also by the system's ability to allow setting changes. These factors, however, are not necessarily sufficient to satisfy User Flexibility. This is why we use +/- labels to describe relationships instead of AND/OR labels. Figure 2 provides only a partial decomposition of the softgoal. In particular, the

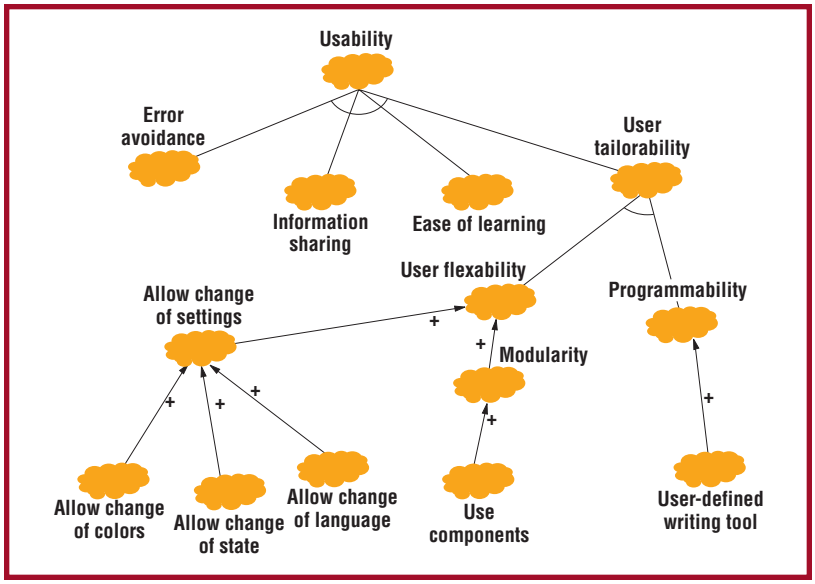

Figure 2. A partial softgoal hierarchy for usability. We adopted this diagram from coursework by Lisa Gibbens and Jennifer Spiess, prepared for a graduate course taught by Eric Yu.

softgoals Error Avoidance, Information Sharing, and Ease of Learning have their own rich space of alternatives, which might be elaborated through further refinements.

What is the relevant knowledge for identifying softgoal decompositions and dependencies? Some of the relevant knowledge might be generic and related to the softgoal being analyzed. For instance, general software system qualities—such as reliability, performance, and usability—all have generic decompositions into a number of finer-grain quality factors. There can also be generic rules for decomposing finer-grain softgoals. Here is one example: "A Speed performance softgoal can be AND-decomposed into three softgoals: Minimize User Interaction, Use Efficient Algorithms, and Ensure Adequate CPU Power." In certain situations, however, you can use project-specific or task-specific decomposition methods after agreement among the project's stakeholders.

For any given software development project, you will always initially set several softgoals as required qualities. Some of these might be technical—such as system performance—because they refer specifically to qualities of the future system. Others will be more organizationally oriented. For instance, it is perfectly reasonable for management to require that introduction of the new system should improve meeting quality (by increasing average participation and/or effectiveness measured in some way) or that it should cut average cost per meeting (where costs include those incurred during the scheduling process). Softgoal analysis calls for each of these qualities to be analyzed in terms of a softgoal hierarchy like the one shown in Figure 2.

Softgoal Correlation Analysis

You can build softgoal hierarchies by repeatedly asking what can be done to satisfy or otherwise support a particular softgoal. Unfortunately, quality goals frequently conflict with each other. Consider, for example, security and user friendliness, performance and flexibility, high quality and low costs. Correlation analysis can help discover positive or negative lateral relationships between these softgoals.

You can begin such analysis by noting top-level lateral relationships, such as a negatively labeled relationship from Performance to Flexibility. This relationship can then be refined to one or more relationships of the same type from subgoals of Performance (such as Capacity or Speed) to subgoals of Flexibility (such as Programmability or Information Sharing). You repeat this process until you reach the point where you can't refine relationships any further down the softgoal hierarchies.

As with the construction of softgoal hierarchies, you can discover correlations by using generic rules that state conditions under which softgoals of one type—say, Performance—can positively or negatively influence softgoals of another type. For example, a correlation rule might state that complete automation can prevent users from customizing or modifying output. This rule negatively correlates certain automation-related performance softgoals to certain flexibility ones.

Figure 3 shows diagrammatically the softgoal hierarchy for Security, with correlation links to other hierarchies developed during the softgoal analysis process.

Goal Correlation Analysis

We next need to correlate the goals shown Figure 1 with all the softgoals identified so far, because we propose to use the latter in order to compare and evaluate the former. For example, alternative subgoals of the goal Schedule Meeting will require different amounts of effort for scheduling. With respect to the softgoal of minimal effort, automation is desirable, while doing things manually is not. On that basis, we can set up positively or negatively labeled relationships from subgoals to choose the schedule Automatically or Manually, as shown in Figure 4. On the contrary, if we determine that meeting quality will be the

criterion, doing things manually is desirable because, presumably, it adds a personal touch to the scheduling process.

Figure 4 shows a possible set of correlation links for a simplified version of the Schedule Meeting goal in terms of the softgoals Minimal Effort and Quality of Schedule. The goal tree structure in the lower right half of the figure shows the refinement of the Schedule Meeting goal, while the two quality softgoal trees at the upper left of the figure represent the softgoals that are intended to serve as evaluation criteria.

Figure 4 illustrates a major advantage of distinguishing between goals and softgoals: It encourages the separation of the analysis of a quality sort (such as Flexibility) from the object to which it is applied (such as System). This distinction lets you bring relevant knowledge to bear on the analysis process—from very generic knowledge ("to achieve quality X for a system, try to achieve X for all its components") to very specific ("to achieve effectiveness of a software review meeting, all stakeholders must be present"). Knowledge structuring mechanisms such as classification, generalization, and aggregation can be used to organize the available know-how for supporting goal-oriented analysis.

Evaluation of Alternatives

The final step of goal-oriented analysis calls for an evaluation of alternative functional goal decompositions in terms of the softgoal hierarchies we've already constructed. You can evaluate alternatives by selecting a set of leaf goals that collectively satisfy all given goals. For our single given goal, Schedule Meeting, we might want to choose the two leaf goals labeled with checkmarks in Figure 4. These leaf goals clearly satisfy the Schedule Meeting goal. In addition, we might want to choose sets of leaf softgoals that collectively either satisfy or at least provide the best overall support for top-level softgoals.

Satisfying softgoals might be impossible because of conflicts. Accordingly, our search for alternatives might involve finding a set of leaf softgoals that maximize their positive support for top softgoals while minimizing negative support. The result of this softgoal evaluation will lead to additional design decisions, such as using passwords or allowing setting changes.

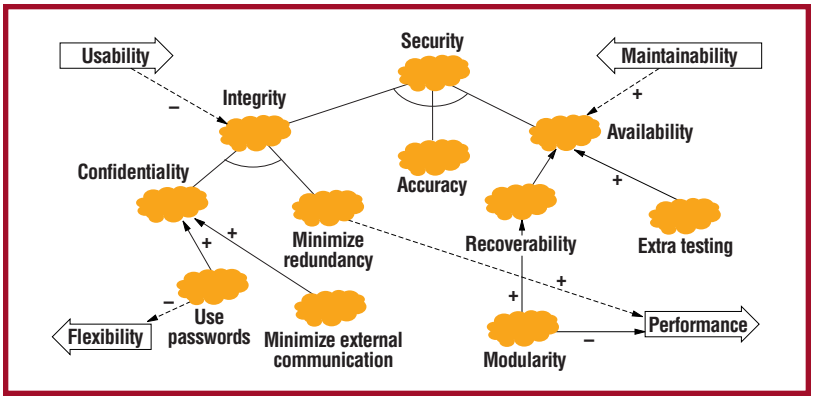

Figure 3. A partial softgoal hierarchy for Security, with correlations to and from other hierarchies.

Of course, generally there will be several possible solutions to the satisfaction of given goals and satisfying or finding the best overall support for softgoals. These alternatives all need to be evaluated and compared to each other.

Putting Together the Pieces

In summary, goal-oriented analysis amounts to an intertwined execution of the different types of analysis we've outlined in this article. The following steps summarize the process:

- Input: A set of functional goals, such as Schedule Meeting, and also a set of qualities, such as Improve Meeting Participation or Reduce Meeting Costs.
- Step 1: Decompose each functional goal into an AND/OR hierarchy.
- Step 2: Decompose each given quality into a softgoal hierarchy.
- Step 3: Identify correlations among softgoals.
- Step 4: Identify correlations between goals and softgoals. Select a set of leaf softgoals that best satisfy all input qualities.
- Step 5: Select a set of goals and softgoals

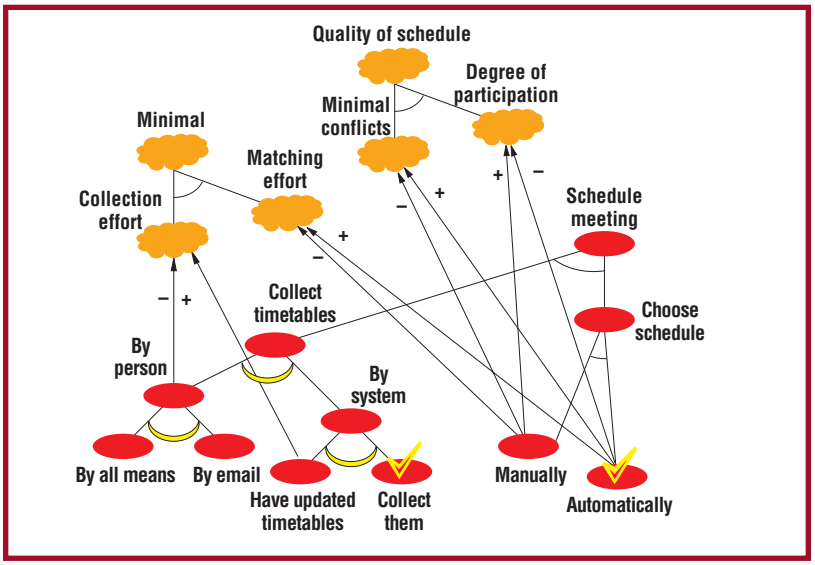

Figure 4. The result of goal correlation analysis for Schedule Meeting.

About the Authors

John Mylopoulos is a professor of computer science at the University of Toronto. His research interests include requirements engineering, databases, knowledge-based systems, and conceptual Modeling. He received a PhD from Princeton University. He was a corecipient of the "most influential paper" award at the International Conference on Software Engineering, is the president of the VLDB Endowment, and is a fellow of the AAAI. He currently serves as coeditor of the Requirements Engineering journal and is a member of the editorial board of the ACM Transactions on Software Engineering and Methodology. Contact him at jm@cs.toronto.edu.

Lawrence Chung is an associate professor of computer science at the University of Texas, Dallas. His research interests include requirements engineering, software architecture, electronic business systems, and conceptual modelling. He received a PhD in computer science from the University of Toronto and serves on the editorial board of the Requirements Engineering journal. He is the principal author of Non-Functional Requirements in Software Engineering. Contact him at chung@utdallas.edu.

Huaiqing Wang is an associate professor at the Department of Information Systems, City University of Hong Kong. His interests include intelligent systems, Web-based intelligent agents, and their e-business applications, such as multiagent supported financial monitoring systems and intelligent agent-based knowledge management systems. He received a PhD in computer science from the University of Manchester. Contact him at jswang@is.cityu.edu.hk.

Stephen Liao is an assistant professor in the Information Systems Department, City University of Hong Kong. His research interests include OO modeling, systems, and technology; user profiling in e-business; and data mining techniques and applications. He received a PhD in Information Systems from Aix Marseille III University. Contact him at ishongko@is.cityu.edu.hk.

Eric Yu is associate professor in the faculty of Information Studies at the University of Toronto. His research interests include conceptual modeling, requirements engineering, knowledge management, and organizational modeling. He has served as cochair of workshops on "Agent-Oriented Information Systems" held at the Conference on Advanced Information Systems Engineering (CAiSE) and the AAAI conference. Contact him at yu@fis.utoronto.ca.

that satisfy all given functional goals and best satisfy all given qualities.

- Output: A set of functions to be performed by the system that collectively meet the functional goals set out.

Output can also reflect a set of design decisions, such as to use back-ups that are intended to address particular quality requirements. In addition, output can be a set of requirements on the system's operational environment, such as the requirement that their owners update timetables at least once a week.

We hope you agree that the steps we've outlined here can precede and augment conventional requirements analysis. Indeed, defining data and functions can begin with the functional requirements produced by a goal-oriented analysis.

For our example, once we select a set of leaf nodes for satisfying the Schedule Meeting goal, we can proceed to define how meeting scheduling will be done and what the role of the new system will be. In other words, conventional requirements analysis assumes that we have already settled on a particular solution that meets predefined organizational and technical objectives through the introduction of the new system.

The framework we presented here is essentially a simplified version of a proposal we've already fleshed out in detail elsewhere.[4] Moreover, this framework is only one sample among many proposals. For example, the KAOS methodology employs a rich and formal notion of goal-identification as the central building block during requirements acquisition.[2] There is also empirical evidence that goal-oriented analysis leads to better requirements definitions.[5]

Different conceptions of meeting a goal have led to different ways of handling conflict and evaluating alternatives.[6] In addition, goals have been used as an important mechanism for connecting requirements to design. The "composite systems" design approach, for instance, used goals to construct and later prune the design space.[7]

While there are indeed several different approaches to goal-oriented requirements analysis, we believe the technique proposed here systematizes the search for a solution that can characterize early phases of software development, rationalizes the choice of a particular solution, and relates design decisions to their origins in organizational and technical objectives.

References

1. A. Davis, *Software Requirements: Objects, Functions, and States*, Prentice Hall, (1993).
2. A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-Directed Requirements Acquisition," *Science of Computer Programming*, vol. 20, 1993, pp. 3-50.
3. J. Mylopoulos, L. Chung, and E. Yu, "From Object-Oriented to Goal-Oriented Requirements Analysis," *Comm. ACM*, vol. 42, no. 1, Jan. 1999, pp. 31-37.
4. L. Chung et al., *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, Dordrecht, the Netherlands, 2000.
5. A.I. Anton, "Goal-based Requirements Analysis," *Proc. 2nd IEEE Int'l Conf. Requirements Engineering*, CS Press, Los Alamitos, Calif., Apr. 1996, pp. **XX-XX**.
6. A. van Lamsweerde, R. Darimont, and P. Massonet, "Goal Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt.," *Proc. 2nd IEEE Int'l Symp. Requirements Engineering*, CS Press, Los Alamitos, Calif., Mar. 1995, pp. 194-203.
7. M. S. Feather, "Language Support for the Specification and Development of Composite Systems," *ACM Trans. on Programming Languages and Systems*, vol. 9, no. 2, Apr. 1987, pp. 198-234.