Automated Support for Requirements Negotiation

William N. Robinson
Department of Computer Science
Oregon State University
Corvallis, OR 97331
wnr@cs.orst.edu

Stephen Fickas
Department of Computer Science
University of Oregon
Eugene, OR 97403
fickas@cs.uoregon.edu

Keywords: Conflict resolution, Resolution generation, Design, CSCW, Tool support.

Abstract

Developing requirements from a group of analysts and system users is a difficult task. In addition to the usual problems of individual requirements acquisition, group requirements acquisition entails conflict detection, resolution generation, and resolution choice. In essence, requirements must be negotiated.

In this paper, we summarize our model for requirements negotiation and its automated support. The model calls for the independent representation of user requirements followed by their negotiation. The model centers around three themes: user participation, resolution generation, and negotiation records. To support these themes, we have built a tool, called Oz, which provides: (1) automated methods for conflict detection and resolution generation, (2) an interactive resolution choice procedure, and (3) records of the negotiation process. This paper overviews our negotiation method and tool support.

1 Introduction

Requirements negotiation is a common and difficult problem for software developers. Varied system perspectives must be understood, their conflict and common ground recognized and developed. Systems analysts need support in guiding this process.

Requirements negotiation consists of three basic processes: (1) conflict detection, (2) resolution generation, and (3) resolution choice. Such processes arise in all requirements methodologies, but are addressed with varying sophistication at a variety of points during the development process. Herein, we present our automated support for these processes.

Our tool, called Oz, assists an analyst in deriving conflict free requirements. The tool assists analysts by: (1) detecting conflicts between requirement sets, (2) generating similar but conflict free requirements, called resolution alternatives, and (3) depicting the value of resolution alternatives so as to aid resolution choice. Oz supplies such support within the context of our multi-perspective requirements model.

As an example, suppose that an analyst is specifying a university library system. Using our multi-perspective model, the analyst develops the requirements using alternative requirement sets based on shared themes, or *stakeholders*. In this way, alternative requirements perspectives are formally represented at the outset. Some likely library stakeholders are: administrators, patrons, managers, accountants, and even lawyers; each has its own formal perspective on how the final library system should function. Some stakeholders may be in direct conflict with each other. For instance, *patrons* may like 24 hour service and extensive on-line search. In contrast, *library administrators* worry about holding costs down. Other stakeholders may seem to be in accordance—*faculty* and *students* both want access to library resources—but a more detailed analysis may show them to also be in conflict; each has different uses of resources, and hence different requirements.

Once the library stakeholder perspectives are represented, Oz can detect conflicts and generate resolutions. For example, Oz can detect a conflict between patrons and administrators over library service hours. Next, Oz will generate resolution alternatives. Standard resolutions are: (1) *one of the given conflicting values* (e.g., the administration's service hour requirement), or (2) *a compromise value* between the conflicting values. Oz can also generate resolutions by using domain knowledge. For example, Oz can generate a resolution containing a relaxed patron requirement of 10 hour service, and to compensate for the patron's loss, a 24 hour computer dial-up service. This resolution is derived using the negotiation strategy of compensation, i.e., give a "losing" stakeholder alternative satisfaction. Such strategies can generate relevant resolutions by using Oz's knowledge of relationships among requirements and the functions which achieve them. After their generation, Oz presents such resolutions with measures of satisfaction calculated from each stakeholder perspective. Then, the analyst can choose resolutions knowing how much satisfaction will be lost or gained by each stakeholder. Through such choices, a conflict free requirements set can be derived.

Providing automated negotiation support is the major focus of this research. We are exploring how to represent stakeholder perspectives and how to reason about their interactions. We are exploring how to automatically detect conflicts, generate resolution alternatives, and present them to a human analyst. Our representations and methods are part of an overall negotiation framework which emphasizes human-computer interaction.

1.1 Our Negotiated Requirements Perspective

Our requirements perspective is derived from our attempt to support the way groups actually interact. The philosophy is: understand how teams work and provide needed automation. Currently, we know teams: work from common goals, work independently, diverge from their group goals, recognize conflicts, modify goals, and integrate work. We know this from studies of specification teams[12][46]. These studies support the many behavioral studies which point to conflict resolution as a cornerstone of group behavior[34].

In addition to supporting collaboration, we want to achieve our requirements engineering goals of: (1) accurate assessment and (2) rational negotiation. It is important to accurately reflect what stakeholders actually want so we can derive systems which they can use. Of course it is necessary to provide stakeholders with feedback concerning what is feasible. Part of this process is recognizing the interaction of stakeholder requirements and rationally dealing with conflicts. We believe conflicts which are resolved using informed negotiation methods will produce better systems.

One difficulty that exacerbates negotiated requirements development is stakeholder (non-) rationality. In our model, we assume stakeholders have *bounded rationality*. Stakeholders are not omniscient, nor do they have complete knowledge of a domain. Consequently, their requirements at the outset will typically differ from those they hold after system specification, construction, or maintenance. We address the problem of bounded rationality in requirements development with interactive perspectives.

1.2 Interactive Stakeholder Perspectives

Oz stakeholder perspectives are interactive in that they can be modified during specification derivation and requirements negotiation. During specification, a stakeholder can modify requirements in response to improved understanding of the derived specification; understanding gained by seeing the actual functions employed and analysis of their interactions. In fact, independent specification construction is part of the Oz development process specifically to aid stakeholder requirements understanding. Only after each stakeholder has a derived specification does requirements negotiation begin.[1] Even then, stakeholders can modify their requirements in response to improved understanding of other stakeholder needs.

Oz uses three mechanisms to support accurate acquisition using interactive perspectives:

- *Independent specifications.* For each stakeholder perspective, a complete *idealized* specification is derived which maximally achieves the stakeholder's requirements. This allows stakeholders to understand functional implications of their requirements.
- *Abstract specifications.* Each specification is constructed by composing cataloged components. The catalog consists of domain-specific abstraction hierarchies which support composition and refinement. Specification refinement is only carried down to the level of specificity required by a perspective. In this way, stakeholders are freed from making arbitrary decisions which can lead to unnecessary conflict.
- *Requirements negotiation via specification integration.* Independent *specifications* are integrated into a single specification using negotiation techniques. This allows stakeholders to modify their requirements during negotiation while understanding functional consequences concerning their own, and others', requirements.

These mechanisms address the problem of operational uncertainty inherent in abstract requirements[50]. Operational uncertainty stems from the variety of operators which can be used to satisfy stakeholder requirements and their interactions. Goals may appear to conflict when, in fact, they do not. Do the goals achieve(x) and achieve(~x) conflict?

1. Oz can be applied directly to perspectives without going through the specification process; however, then its conflict *interference* processes cannot be applied[37].

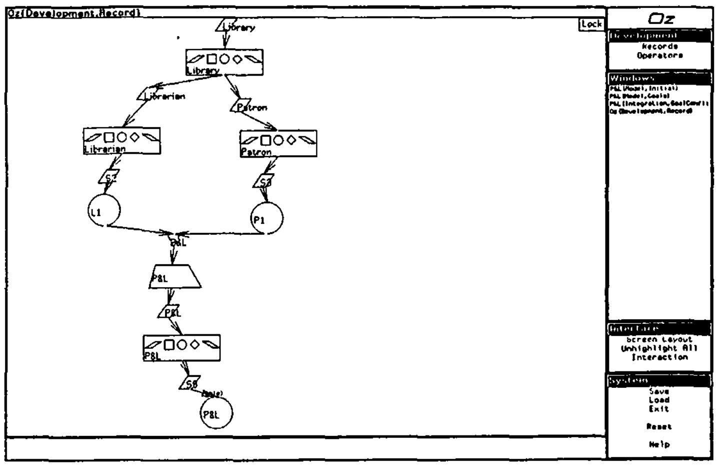

Figure 1. *Oz screen depiction of group requirements and specification development.*

They may not, if their achievement can vary in time or place. Goals "conflict" if, and only if, all known ways to achieve them conflict[55][59]. Hence, there is a trade-off between unnecessarily creating specifications for goals which will be dropped or compromised, versus unnecessarily negotiating nonconflicting goals—and possibly compromising. In Oz, we wait and see if specifications conflict before engaging in negotiation.

2 The Oz Requirements Negotiation Tool

Oz was constructed to address negotiated requirements development. It provides: (1) a requirements language, (2) a specification language, (3) a plan-based specification constructor, and (4) negotiation tools consisting of: (i) a conflict detector, (ii) a conflict characterizer, (iii) a conflict resolver, and (iv) an interactive resolution chooser. Oz is experimental. It has *not* been applied by real-life analysts; nor would that be appropriate yet! However, Oz does provide an environment for negotiated requirements development.

To convey a sense of how Oz works, we present the following integration example drawn from the library domain. The example concerns resource loan periods and overdue notices for the University of Michigan General Library[7].[2] The problem involves a conflict between two representative stakeholders trying to specify a portion of the circulation system: a librarian who wishes to reduce loan periods and minimize the number of overdue notices sent out, and a patron who wishes to maximize loan periods and maximize the number of overdue notices sent out.

2.1 Processes in the Oz Model

Our rederivation will proceed as follows:

(1) *Modeling.* An analyst represents librarian and patron requirements in the Oz requirements language.

(2) *Specification.* From the requirements, the Oz plan-based specification constructor derives individual functional specifications for the librarian and patron.

(3) *Integration.* Using the individual specifications, Oz engages the librarian and patron perspectives in requirements negotiation.

Figure 1 shows an Oz depiction of library development. Rectangles depict the generic library requirements model, and specific patron and librarian requirements for the example. Circles linked below the requirements depict derived specifications. The trapezoid depicts the integration process; it records negotiation information. Below it are the negotiated requirements and specification.

In applying Oz to the example, Oz: (1) detects the loan period and overdue notice conflicts, and (2) generates a variety of resolutions. During generation, the analyst chooses resolutions matching those found in the UM analysis. Finally, an integrated specification is derived.

2. A more complete analysis of this example is found in[36].

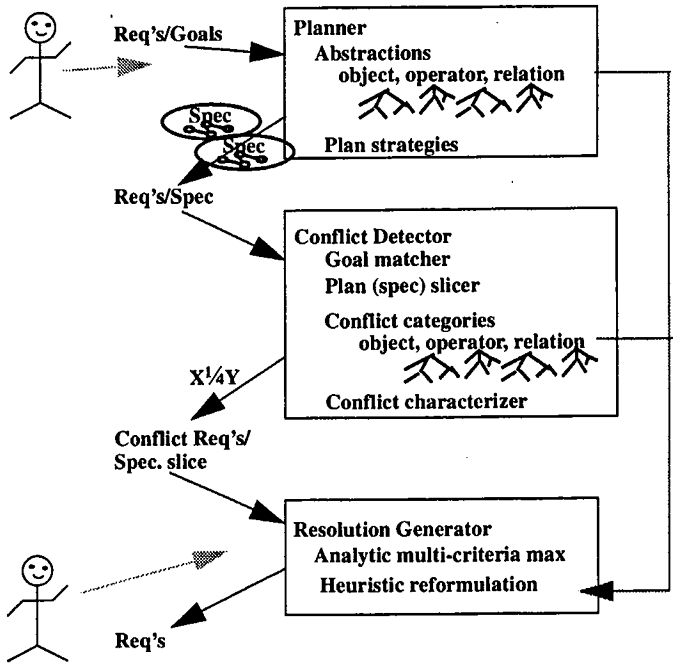

Figure 2. *Oz system components.*

2.2 Oz Architecture

Oz provides support through three basic knowledge-based components. Figure 2 illustrates these components.

- *Planner.* The planner takes system requirements as input and outputs a plan achieving those requirements. During planning, it makes use of a *domain model* consisting of operator, object, and relation abstraction hierarchies. For example, abstract operators are inserted into a plan achieving systems requirements. As planning continues, these operators are refined until a complete plan satisfying all preferences and system goals is obtained. The operators from the complete plan form a functional specification.
- *Conflict detector.* The conflict detector takes specifications and their associated requirements as input. It outputs two types of conflicts: (1) syntactic differences between "similar" requirements and (2) operational interference between specification-level components.
- *Resolution generator.* The resolution generator take conflicts as input and outputs resolutions. Resolution generation is based on two methods: (1) analytic multi-criteria goal maximization (compromise within a constraint space) and (2) heuristic reformulation. Heuristic reformulation makes use of the planner's abstraction hierarchies to reformulate conflicting system requirements into similar non-conflicts requirements, thereby "dissolving" conflicts[37][59].

2.3 Integration

Requirements negotiation is instigated through the process of specification integration. Requirements and specifications of the two stakeholders are compared, conflicts are noted, and then requirements are negotiated within the context of the specifications. The five processes of integration are described below in the context of the UM example.

(1) *Conflict detection.* As in the UM library study, Oz detects the loan period and overdue notice requirements conflicts.

(2) *Conflict characterization.* The conflicts are characterized as: (i) object attribute conflicts, a syntactic conflict category of the requirements language; and (ii) interfering, a category of specification level plan interactions.

(3) *Conflict resolution.* Using the conflicts, Oz is able to apply three basic methods of resolution generation:

(a) Replanning. Oz conjoins the conflicting loan period requirements of the patron and librarian and sends them to the planner to determine if alternative functions can achieve both requirements. In this case, Oz does not have a function in its library domain model which can achieve both requirements. Similarly, alternative functions cannot be found for the overdue notice requirements.

(b) Compromise. Oz generates compromises for both the loan period duration and overdue notice conflicts. Each compromise value must be within ranges specified in the generic domain model and must not violate any constraints. In the UM problem, loan period duration compromises range from 0 to 365 days and overdue notice numbers range from 0 to 5. Both ranges were specified in the generic library model and no other constraints apply.

(c) Reformulation. Oz reformulates conflicting requirements through: (i) generalization, (ii) specialization, and (iii) augmentation. For example, the conflicting loan period requirements concerning borrow(patrons, loan_period, resources) can be specialized to borrow(faculty, loan_period, resources) & borrow(undergraduate, loan_period, resources), where the conflicting values of 365 and 14 are associated with the specialized loan periods for faculty and undergraduates, respectively. Hence, a conflict may be *dissolved* by specializing a requirement and distributing conflicting values over the resulting set. Conversely, if there were already such specialized patron borrow requirements, the generalized requirement of borrow(patrons, loan_period, resources) with a single blanket loan period value could be generated.[3] In general, Oz can reformulate requirements by substituting or augmenting related requirements for conflicting requirements. In the UM example, Oz can generate resolutions containing requirements of patrons buying, recalling, or renewing resources. These can be substituted for the borrow requirement; for example, the library could become a bookstore. Similarly, the new requirements can augment the original ones; for example, the 14 day loan periods will be applied, but resource selling and recalling will be added to *compensate* for poor patron satisfaction.

The following table presents some reformulations. For example, the first specialization was generated by finding on_loan in the relation hierarchy and moving down one link to the specializations of patron (student and faculty). Additionally, the conflicting loan_period.duration values have been distributed over the relation specializations.

Method	Resolution

Specialization	on_loan(student resource loan_period.duration=14)
	on_loan(faculty resource loan_period.duration=365)

Specialization	on_loan(patron periodical loan_period.duration=14)
	on_loan(patron book loan_period.duration=365)

Specialization	on_loan(undergrad book loan_period.duration=14)
	on_loan(faculty periodical loan_period.duration=365)

Generalization	own(patron book)
Generalization	recall(graduate periodical)
Generalization	renew(faculty book)

The table also presents generalizations derived from the operator hierarchy. For example, given the on_-loan relation, the own(patron resource) ^ own(library money) relations are generated through the buy_resource operator.

(4) *Resolution choice.* Oz does not blindly apply all resolution methods to all conflicts. Instead, a human analyst chooses which methods to apply to which conflicts and in what order. For example, the analyst could direct the specialization of the loan period conflict, producing requirements for faculty and undergraduates. Then, the analyst could apply generalization to the new requirements to produce requirements for faculty and undergraduate renewal. Thus, all generation methods can be applied to the original conflicts and the subsequent resolutions. This allows the human analyst to incrementally explore alternative resolutions.[4] Finally, the analyst selects the requirements to include in the integrated perspective. In our strict rederivation of the UM study, we choose a blanket 21 day loan duration requirement and two overdue notices. Later, we produced a

3. Given such specializations as input, Oz can detect them as "potential conflicts", meaning that they represent varying but non-interfering requirements. Sometimes it is useful to negotiate such requirements, as they indicate special-case policies rather than blanket policies. Hence, Oz presents all requirements differences for negotiation.
4. This also finesses automating the difficult processes of: (1) resolution generation control, (2) resolution choice, and (3) resolution feasibility. As an example of resolution feasibility, note that Oz currently does not have to determine whether *substituting* a renewal requirement for a loaning requirement is a good choice—it is not.

specification containing specialized loan period requirements.

(5) *Re-specification.* Once the analyst has chosen the new requirements, Oz derives a single group perspective. Original requirements from one perspective can be transformed (with a little effort) to produce the integrated requirements by using the following formula:

original requirements - conflicting requirements + resolved requirements

The resulting perspective is then given to the planner which then derives the integrated specification.

3 Conclusions

While Oz is still a prototype, our experience leads us to conclude that negotiated requirements development can be effectively supported through: (1) automated conflict detection, characterization, and resolution generation, and (2) resolution decision-making support. The balance between automation and human interaction has been a key to Oz's success. Our model recognizes the bounded rationality of humans and allows for the acquisition of preferences in the context of conflicting specifications. Additionally, our interactive resolution procedures fits within the more general framework of interactive negotiation support:

- Stakeholder participation
- Resolution generation
- Negotiation records

4 References

[1] Adams, E., Fagot, R., A model of riskless choice, in: Eds. W. Edwards, A. Tversky, *Decision making*, (1967) 284-289.

[2] Adler, M., Davis, A., Weihmayer, R., and Worrest, R., *Conflict-resolution strategies for nonhierarchical distributed agents*, Morgan Kaufmann Publishers Inc.(1989).

[3] Anderson, J., Farley, A., Plan abstraction based on operator generalization, In *Proceedings of the 1988 AAAI Conference*, Minneapolis

[4] Anderson, J., Fickas, S., Viewing Specification Design as a Planning Problem: A Proposed Perspective Shift, In *5th International Workshop on Software Specification and Design*, Pittsburgh, 1989 Also in *Artificial Intelligence and Software Engineering*, D. Partridge (ed), Ablex, 1991

[5] Bazerman, M., *Judgment in managerial decision making*, John Wiley & Sons(1986).

[6] Burkhalter, B.R., Race, P.A., An analysis of renewals overdues, and other factors influencing the optimal charge-out period, In *Case studies in systems analysis in a university library*, 1968, The Scarecrow Press, Inc., Metuchen, N.J., 11-33.

[7] Bui, T., Co-oP: *A group decision support system for cooperative multiple criteria group decision making*, Springer-Verlag, 1987

[8] Chen, M., Nunamaker, J., The Integration of GDSS and CASE: A Metasystem Approach, *Proceedings of 2nd International Workshop on CASE*, 1988

[9] Conklin, J., Interissue dependencies in gIBIS, STP-091-89, MCC (February 10, 1989)

[10] Conry, S., Meyer, R., and Lesser, V., Multi-stage negotiation in distributed planning, Eds. A.H. Bond, L. Gasser, *Readings in distributed artificial intelligence*, Morgan Kaufmann, San Meteo, California (1988) 367-384.

[11] Curtis B., Krasner H., Iscoe I., A field study of the software design process for large systems, ACM, *CACM*, 31 (11) November 1988, 1268-1287.

[12] Easterbrooke S., Domain modeling with hierchies of alternative viewpoints, IEEE, International Symposium on Requirement Engineering, January 4-6, 1993, 65-72.

[13] Festinger, L., *Conflict, Decision, and Dissonance*, Tavistock Publications, Ltd., London(1964).

[14] Fickas, S., Robinson, W., Feather, M., Conflict and compromise in specification design, In *Proceedings of the AAAI-88 Automated Software Development Workshop*, Minneapolis, 1988

[15] Finkelstein, A., Fuks, H., Multi-party specification, *5th International workshop on software specification and design*, (1989) 185-195.

[16] Fuks, H., Negotiation using commitment and dialogue, Imperial College of Science Technology and Medicine, London(February,1991)

[17] Janis, I., Mann, L., Decision making : a psychological analysis of conflict, choice, and commitment, The Free Press, New York(1979).

[18] Keen, P., Scott-Morton, M., *Decision support systems: an organizational perspective*, Addison-Wesley, 1978

[19] Keeney, R., Raiffa, H., *Decisions with multiple objectives*, John Wiley and Sons, New York(1976).

[20] Klein, M., Supporting conflict resolution in cooperative design systems, IEEE, *Transactions on Systems, Man, and Cybernetics*, 21 (6), November 1991, 1379-1390.

[21] Kraemer, K., King, J., Computer-based systems for cooperative work and group decision making, *Computing Surveys* Vol. 20 (June 1988) 115-146.

[22] Kwa, J., Tolerant planning and negotiation ingenerating coordinated movement plans in an automated factory," *Proceedings of the first international conference on industrial and engineering applications of artificial intelligence*, (1988)

[23] Lander, S., Lesser, V., A framework for the integration of cooperative knowlege-based systems, *Workshop on integrated architectures for manufacturing*, IJCAI , Detroit, Michigan (August 24, 1989)

[24] Mazer, M., A knowledge-theoretic account of negotiated commitement," CSRI-237, Univerity of Toronto (November 1989).

[25] Mostow, J., Voigt, K., Explicit integration of goals in heuristic algorithm design, IJCAI87 (January 1987)

[26] Robbins, S., *Organizational behavior: concepts, controversies, and applications*, Prentice Hall, NJ(1983).

[27] Robinson, W., Integrating multiple specifications using domain goals, *5th International workshop on software specification and design*, (1989) 219-226

[28] Robinson, W., Negotiation behavior during requirement specification, *Proceedings of the 12th International Conference on Software Engineering*, IEEE Computer Society Press, Nice, France (March 26-30 1990) 268-276

[29] Robinson, W., Automated negotiated design integrateion: formal representations and algorithms for collaborative design, CIS-TR-93-10, University of Oregon, (April 1993).

[30] Robinson, W.N., Fickas, S. Supporting Multi-Perspective Requirements Engineering, International Conference on Requirements Engineering, IEEE, (April 18-22 1994).

[31] Rosenschein, J., Genesereth, M., Deals among rational agents, *Proceedings of the 1985 IJCAI*, (August 1985) 91-99.

[32] Rosenschein, J., Ginsberg, M., Genesereth, M., Cooperation without communication, *Proceedings of AAAI-86*, Morgan Kaufmann Publishers, Inc. (1986) 51-57.

[33] Rosenschein, J. Breese, J., Communication-free interactions among rational agents: a probabilistic approach," in: Eds. L. Gasser, M.N. Huhns, *Distributed artificial intelligence*, Morgan Kaufmann Publishers Inc. (1989)

[34] Ross, D.T.,Structured Analysis (SA): A language for communicating ideas, IEEE, *Transactions on Software Engineering*, 3 (1) January 1977, 16-34.

[35] Sathi, A., Morton, T., Roth, S., Callisto: an intelligent project management system, *AI Magazine*, (Winter 1986) 34-52.

[36] Sathi, A., Fox, M., Constraint-directed negotiation of resource reallocations, Eds. L. Gasser, M.N. Huhns, *Distributed Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Mateo (1989) 163-193.

[37] Scacchi, W., Bendifallah, S., Work structures and shifts: an emperical analysis of software specification teamwork, IEEE, 11th International conference on software engineering, May 1989.

[38] Scacchi, W., Managing software engineering projects: a social analysis, IEEE, Transactions on software engineering, 10 (1) January 1984, 49-59.

[39] Shepard, R., On subjectively optimum selections among multi-attribute alternatives," Eds. W. Edwards, A. Tversky, *Decision making*, (1967) 257-283.

[40] Swartout, W., Balzer, R., On the inevitable intertwining of specification and implementation, *CACM* Vol. 25 (1982) 438-440.

[41] Sycara, K., Resolving goal conflicts via negotiation, *Proceedings of the AAAI-88*, (1988) 245-250.

[42] Wilensky R., Planning and understanding, Addison-Wesley, 1983.

[43] Werkman, K., Knowledge-based model of using shareable perspectives, *Proceedings tenth international conference on distributed artificial intelligence*, (October 1990) 1-23.

[44] Yakemovic, K., Conklin, J., Experience with the gIBIS model in a corporate setting, *Proceedings of CSCW 90*, Los Angeles, 1990

[45] Zeleny, M., *Multiple criteria decision making*, McGraw-Hill(1982).