Negotiation in Composite System Design

William Robinson

CIS-TR-91-11
May 1, 1991

Department of Computer and Information Science
University of Oregon
Eugene, OR 97403

Negotiation in Composite System Design*

William N. Robinson

**Department of Computer and Information Science,
University of Oregon, Eugene, OR, 97403, U.S.A.**

1. Introduction

In this position paper, we argue for a multiple perspective design methodology in exploratory domains. First, we present the library domain; a domain where a significant number of system tasks, agents, and responsibilities are derived during system specification. Such exploratory domains differ from more routine domains (e.g., elevator design) in that (1) interactions between system goals are not well understood and (2) system policies and agent responsibilities are not established *a priori*; composite design addresses these problems. Next, (§ 3) we describe how requirements negotiation is part of composite system design. Section 4 presents Multiple Perspective Specification Design, a type of composite system design for deriving specifications. Finally, (§ 5) we conclude that MPSD (1) captures part of the natural negotiation in design, (2) addresses agent motivation, a key composite design issue, and (3) produces good designs in exploratory domains.

2. The Library Domain

For most readers, the library domain should be familiar; it has provided other researchers with simple examples[12, 32]. Here, we would like to illustrate some of the domain's complexity.

Libraries come in many forms (e.g., public, private) and serve a variety of needs. Their charters are typically broad and demanding; a university library may be responsible for[17]:

- providing a collection of information resources which meet most of the needs of the university community;
- organizing, maintaining, and controlling collections; and
- providing bibliographic aids in identifying, locating, and using resources.

From such broad charters, more specific policy guidelines are developed (e.g., collection development[15], circulation policies[1, 16], interlibrary loan policies[13]). Finally, from such policies, specific library procedures are designed[20].

---

*Presented at the Stanford Spring Symposium on Composite Systems Design, March 1991.

Deriving specific library policies and procedures is a process of negotiation. Librarians (desk, circulation, collections), administrators, and patrons all have a stake in library operation. Fees, fines, loan periods, check-out, and renewal policies all result from placating various stakeholders. Two examples are:

- loan periods

  From a patron's perspective, loan periods should be as long as possible; this insures their ability to enjoy borrowed resources. On the other hand, a circulation librarian desires to insure equal access of resources to all patrons; hence, shorter loan periods provide higher turnover which enables greater access to a large population of patrons. A variety of loan periods result (e.g., 2 hours to indefinite); they can vary according to patron type (e.g., child, student, administrator, librarian); they can be extended (e.g., desk renewal, phone renewal); they can be terminated (e.g., recall, revoked privileges).

- information access

  Patrons want information without restriction. Librarians wish to assist patron information retrieval. However, the administration must protect the privacy of others. A library that allows a patron to view her own borrowing record, but not that of others is a compromise; it protects privacy, but leaves the possibility for illicit access through misrepresentation (e.g., stolen passwords).

Libraries try to satisfy the conflicting concerns of patrons, staff, and administrators; they employ many mechanisms to deal with both errorful and irresponsible behavior; they involve complex responsibility assignments among agents. Libraries are complex systems which require sophisticated analysis to derive adequate specifications of their embedded software.

> Real libraries are not simple. They involve more than just people, books, and a database. They have policies according to who the borrower is, what kind of book it is, what time of year it is, and, of course, exceptions to all of these policies[32].

In contrast, others view library specification as refinement of a generic database/tracking schema[21, 22]. This paradigm is based on: (1) capture and storage of library forms and (2) retrieval and modification of such forms for specification. This paradigm appears to be productive for routine situations; for example, primitive programming tasks such as sorting and searching[23]. However, this paradigm must be extended for exploratory tasks such as requirements engineering, especially for exploratory domains such as library science. Even where significant forms can be captured, the bulk of the requirements work lies in selection and modification. For tasks such as library specification, this means understanding policies, their derivation via negotiation, and their effects on alternative specifications. This negotiation-oriented paradigm has been combined with the capture-and-modify paradigm for labor negotiations[30]; it illustrates the need for knowledge beyond data schemas for assistance in complex domains.

3. Composite Design

The library domain illustrates how an artifact's complexity can result from its environmental interface rather than its internal processing. The actual algorithms used in an automated circulation and inventory system may be simple, but the policies they implement may be the result of complex negotiations between system stakeholders. Hence, system complexity may be due to system specification, rather than algorithmic derivation; algorithmic enhancements are simple, but system changes must be negotiated with

stakeholders.

Composite system design attempts to address the complexity of the design process. Sprung from closed-specification research, composite design models the environment[2, 33]; however, composite design researchers are explicitly focused on human-computer interactions[6]. In particular, they focus on how an artifact can be designed to combine with an existing environment to effectively achieve goals[10].

Composite design coordinates an artifact and its environment to achieve goals. Coordination is achieved through agents. Composite design consists of designing motivated agents to responsibility and reliably execute assigned tasks which achieve system goals. Some agents will be animate and others will be artifacts. Based on these agents and their environment, composite design must address five basic issues:

(1) What tasks can achieve the desired system goals?

(2) What abilities do agents have in the existing environment and which must be added?

(3) Who will be responsible for executing which tasks?

(4) How can task execution be reliably achieved (assuming this is an implicit system goal)?

(5) How can agents be motivated to execute their tasks?

Given these basic issues, a composite design model must describe how these interrelated issues are settled during the design process. For example, one could use system goals and current agent abilities to derive an initial task structure; next, agents could be assigned tasks by ability, reliability, and motivation.

Multiple Perspective Specification Design is a type of composite design; it addresses the interaction of agents with their environment. Models of agent goal beliefs are combined with environmental models of goal interaction and costs. Such modeling enables an automated MPSD system to assist in composite design; specifically, it is aimed at understanding agent motives and their ability to satisfy goals.

Agent motivation is our main composite design focus; it is addressed through agent negotiated requirements. By involving system stakeholders in the requirements process, we can tailor systems to their work structures and gain their commitment when such structures are changed.

4. Multiple Perspective Specification Design

Initially, our research on Multiple Perspective Specification Design was purely technical; we wanted to automate the merging of independent specification design states[24]. Our resulting tool would assist analysts in their construction of composite specifications[8, 9]. While this initial work did produce a design tool and some simple merge assistance, it also illustrated the difficulty of merging specifications[25]. We found that merge models based exclusively on program development records or language semantics are inadequate; such models can only merge noninterfering programs[5, 11]. This lead us to consider agent modeling as a basis for deciding how interfering specifications could be integrated[7, 26]. Now, we are developing a model which explicitly addresses the conflicts derived from integrating specifications; these conflicts arise from the multiple participants of a software system[10, 27, 28]. Through our automation attempts, we have also developed a requirements methodology.

Our methodology captures aspects of negotiation between system stakeholders. Commonly, requirements are developed through client interviews; sometimes the interviewees are potential operators

of software system, but often they are presumptuous managers who impose their own requirements without any user consultation. In any case, multiple agent goals are typically unrepresented in requirements engineering models. We prefer a more direct approach; we model system participants, called stakeholders, who might affect or be affected by the proposed system. Mumford calls such an approach participative systems design[18].

Multiple Perspective Specification Design is consistent with soci-technical participative design[19]. Both advocate independent consideration of systems aspects from stakeholder perspectives. However, Mumford's approach restricts independent consideration along two issues: social and technical; whereas, MPSD has no issue restrictions. Multiple requirements and multiple specifications are also unique to MPSD; we advocate the production of specifications for each stakeholder. Finally, MPSD advocates the integration of "incompatible" specifications; the process of their integration reveals the negotiation between stakeholders and can be assisted.

Multiple Perspective Specification Design calls for: (1) representing stakeholder beliefs, (2) constructing separate specifications for each stakeholder, and (3) integrating specifications using negotiation techniques. Figure 1 illustrates the MPSD integration paradigm.

System support consists of: (1) agent modeling, (2) development "bookkeeping", and (3) negotiation assistance. A domain model is provided to model agent and environmental interactions: values and preferences represent what an agent wants to achieve; goal relationships and constraints represent how the environment obstructs or supports agent goals. The domain model is our requirements language.

Requirements acquisition consists of using the domain model to construct stakeholder perspectives. Perspectives represent the interests of a stakeholder in the proposed system; they are individual requirements. Acquisition is supported by providing a model of the domain and tools to assist the analyst in tailoring it to individuals.

Analysts use perspectives to guide their construction of specifications; each requirement is mapped to specification components. Specification construction is supported by language-oriented editing commands, assisted linkage of requirements and specification components, and managing of multiple perspectives and specifications.

Integrating specifications is the focus of this research; hence, this is where our automation efforts lie. We support a single arbitrator in the integration of specifications. Unlike binding arbitration, our arbitrator does not choose an alternative; instead, he combines the specifications. The arbitrator is assisted in negotiated arbitration through an interactive problem-solving model of conflict resolution. Such a model is more comprehensive than a multiple view model; those models merge the same information represented in different forms[14].

5. Conclusions

The process of requirements acquisition and specification design can be characterized as an interdependent negotiation process[28]. This may be particularly valid for exploratory domains; when a decision maker cannot *recognize* when requirements interact, she must explore interactions by exploring alternatives and identifying how well those alternatives meet her goals. We believe that many requirements engineering tasks are in exploratory domains. Swartout and Balzer's theory of the interwining between specification and implementation supports this; incomplete modeling suppresses the discovery of undesirable

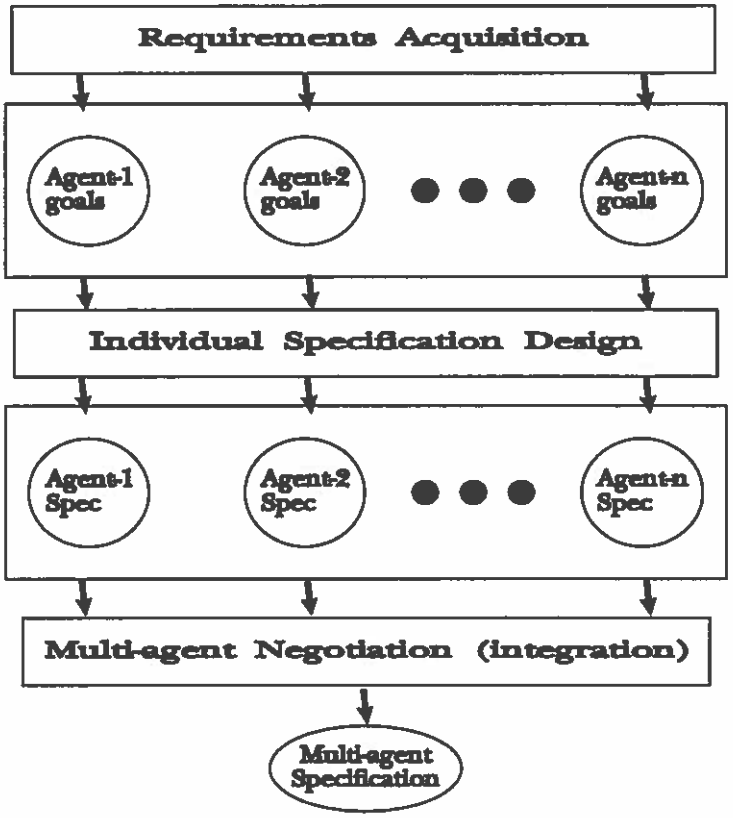

Figure 1. The MPSD Integration Paradigm.

interactions until implementation[29]. The proliferation of rapid prototyping techniques is further evidence of this[3, 4, 31].

Multiple Perspective Specification Design captures (and supports) the exploratory negotiations found in requirements engineering. It does so by modeling agents and assisting the derivation of negotiated specifications. Specifically, it considers individual agent requirements (motivation and constraints) and assists the cooperative exploration of group requirements. By explicitly supporting multiple agent requirements, we can document what has been an implicit process; moreover, we can improve the negotiation process, thereby producing better designs.

REFERENCES

1. American Library Association, *Circulation policies of academic libraries in the United States, 1968*, American Library Association(1970).
2. R. Balzer and N. Goldman, "Principles of good software specification and their implications for Specification Languages," *Proceedings of IEEE Conference of Specifications of Reliable Software*, (1979) 58-67.
3. R. Balzer, N. Goldman, and D. Wile, "Operational specification as the basis for rapid prototyping," *Sigsoft Software Engineering Notes* 7 (December 1982) 3-16.
4. L. Beck and T. Perkins, "A survey of software engineering practice: tools, methods, and results," *Transactions on Software Engineering* SE-9 (September 1983) 541-561.
5. V. Berzins, "On merging software extensions," *Acta Information* 23 (1986) 607-619.
6. E. Doerry, S. Fickas, R. Helm, and M. Feather, "Deriving interface requirements through composite system design," *Submitted to Human Computer Interfaces*, (July 1990)
7. M. Feather, S. Fickas, and W. Robinson, "Design as elaboration and compromise," in: *Proceedings of the AAAI-88 Workshop on Automating Software Design*, Kestrel Institute , AAAI-88, St. Paul, MN (August 25, 1988) 21-22.
8. M.S. Feather, "Language support for the specification and development of composite systems," *Transactions on Programming Languages and Systems* 9 (April 1987) 198-234.
9. M. S. Feather, "Constructing specifications by combining parallel elaborations," *Transactions on Software Engineering* 15 (February 1989) To appear (Also available as RS-88-216 from ISI).
10. S. Fickas, J. Anderson, and W.N. Robinson, "Formalizing and automating requirements engineering," CIS-TR-90-03, University of Oregon (April 6, 1990).
11. S. Horwitz, J. Prins, and T. Reps, "Integrating non-interfering versions of programs," #690, University of Wisconsin-Madison (March 1987).
12. R. Kemmerer, "Testing formal specifications to detect design errors," *Transactions on Software Engineering* SE-11 (January 1985) 32-43.
13. D.F. Kohl, *Circulation, interlibrary loan, patron use, and collection maintenance: A handbook for library management*, ABC-Clio Inc.(1986).
14. J.C.S. do Prado Leite, *Viewpoint resoluiton in requirements elicitation*, Univerisity of California Irvine(1988).
15. Association of Research Libraries, "Collection development policies 1977," *Systems and Procedures Exchange Center*, (November 1977)
16. Association of Research Libraries, "Automated circulation," *Systems and Procedures Exchange Center*, (April 1978)
17. Association of Research Libraries, "SPEC kit on goals and objectives 1979," *Systems and Procedures Exchange Center*, (October 1979)
18. E. Mumford and D. Henshall, *Aparticipative approach to computer systems design*, Halsted Press, New York(1979).
19. E. Mumford and M. Weir, *Computer systems in work design—the ETHICS method*, Associated Business Press, London(1979).
20. J.W. Perkins and P.N. Clingen, *Inglewood public library circulation procedures*, Inglewood public library(1972).
21. C. Rich, R.C. Waters, and H.B. Reubenstein, "Toward a requirements apprentice," *4th International workshop on software specification and design*, (April 3-4, 1987) 79-86.

22. C. Rich and R.C. Waters, "The programmer's apprentice: a research overview," *Computer*, (November 1988) 10-25.

23. C. Rich and R.C. Waters, *The programmer's apprentice*, ACM press, New York(1990).

24. W.N. Robinson, *Towards formalization of specification design*, Masters thesis, University of Oregon(June 1987).

25. W.N. Robinson, "Automating the parallel elaboration of specifications: preliminary findings," Technical Report CIS-TR-89-02, University of Oregon (February 1989).

26. W.N. Robinson, "Integrating multiple specifications using domain goals," *5th International workshop on software specification and design*, (1989) 219-226 (Also available as CIS-TR-89-03 from the University of Oregon).

27. W.N. Robinson, "Negotiation behavior during requirement specification," in: *Proceedings of the 12th International Conference on Software Engineering*, IEEE Computer Society Press , Nice, France (March 26-30 1990) 268-276 (Also available as CIS-TR-89-13 from the University of Oregon).

28. W.N. Robinson and S. Fickas, "Negotiation freedoms for requirements engineering," CIS-TR-90-04, University of Oregon (April 6, 1990).

29. W. Swartout and R. Balzer, "On the inevitable intertwining of specification and implementation," *CACM* 25 (1982) 438-440.

30. K.P. Sycara, "Resolving adversarial conflicts: an approach integrating case-based and analytic methods," GIT-ICS-87/26, Georgia Institute of Technology (1987).

31. A. Wasserman, "Software tools in the user software engineering environment," in: Eds. D. Barstow, H. Shrobe, E. Sandewall, *Interactive programming environments*, McGraw-Hill (1984) 370-386.

32. J.M. Wing, "A study of 12 specifications of the library problem," *Software*, (July, 1988) 66-76.

33. P. Zave, "The operational versus the conventional approach to software development," *CACM* 27 (February, 1984) 104-118.