AAAI-93 FALL SYMPOSIUM ON HUMAN-COMPUTER COLLABORATION: RECONCILING THEORY, SYNTHESIZING PRACTICE;
WORKSHOP NOTES PP.63-68 RELEIGH NC, 22-24 OCTOBER, 1993

# Knowledge Delivery: Facilitating Human-Computer Collaboration in Integrated Design Environments

**Kumiyo Nakakoji[1,2] and Gerhard Fischer[1]**

[1]Department of Computer Science and Institute of Cognitive Science
University of Colorado
Boulder Colorado 80309-0430, USA

[2]Software Engineering Laboratory at Boulder
Software Research Associates, Inc.
1-1-1 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan

kumiyo@cs.colorado.edu, gerhard@cs.colorado.edu

**Abstract.** This paper illustrates our approach of building an integrated knowledge-based design environment, which facilitates knowledge delivery mechanisms. The KID (Knowing-In-Design) design environment provides a specification and a construction components which allow a user to specify design problem requirements at abstract levels, and to form a solution by direct manipulation in a domain of kitchen floor plan design. A partial problem specification and solution construction given through the components represent the user's task at hand to be shared with the design environment. The environment dynamically identifies and informs of a possible breakdown in a partial design, and delivers information from the knowledge bases that are inferred to be relevant to the identified task at hand. The study of KID indicates not only that the system has improved design processes, but also that the delivery mechanism is an effective method to elicit knowledge from designers, addressing the knowledge acquisition problem.

## Collaborative Problem Solving

*Design* tasks, such as architectural design, writing, music composition or software design, have two characteristics (Nakakoji, 1993a). First, design problems are *ill-defined* (Simon, 1981) that one cannot specify a design problem completely before starting to solve it; designers have to gradually refine both the problem specification and the design solution. Second, design problems are *open-ended* that knowledge necessary for a design can never be completely articulated a priori. It is neither possible to identify all the relevant knowledge for the design nor to formalize it for generating a design that fits to varieties of changing needs of design tasks.

Such design activities are best supported by taking a human-computer cooperative problem-solving approach. The approach augments the skills of human designers with *integrated, domain-oriented, knowledge-based design environments* instead of generating solutions for designers as typified by the design automation approach. Design environments are computer systems that provide design tools and information repositories that designers use for understanding, reflecting on, and framing both a problem and a solution. In this approach, a computer system becomes a collaborative assistant and an intelligent agent for designers.

In this paper, first we describe observed characteristics in human-human collaborative problem solving. Then, we discuss how to apply such characteristics to a design of human-computer collaborative problem solving systems, and present two principles. Finally, we present the mechanisms and assessment of the KID design environment (Nakakoji, 1993a), which has been prototyped according to the principles.

## Observations in Human-Human Collaborative Problem Solving

Through empirical studies, we have identified three characteristics in human-human collaborative problem solving: coevolution of a problem and a solution, roles of critics in expert-novice collaboration, and shared understanding of a task for retrieving relevant information.

**Coevolution of a Problem and a Solution.** In many situations, people are initially unable to articulate complete requirements for problems (Fischer, Reeves, 1992). Current goals lead to a partial solution, while the gradually changing overall solution suggests new goals (Simon, 1981). Professional practitioners have at least as much to do with defining the problem as with solving the problem (Rittel, 1984). Problem framing and solving cannot be separated.

Starting with a vague incomplete problem requirement, designers sketch out a partial solution. By seeing the partial solution, designers identify portions of the problem that have not yet been understood, gain an understanding of the problem, and then refine the solution (Snodgrass, Coyne, 1990). By iterating this reflection, understanding of the problem gradually emerges.

**Roles of Critics in Expert-Novice Collaboration.** Thus, understanding what the problem is plays a major part in design activities. Miyake [1986] studied the cycle of *"non-understanding"* and *"understanding"* by observing an expert and a novice interacting while trying to understand the mechanism of a sewing machine. Miyake accounted for the role *critics* play in augmenting human understanding, viewing critics as the expression of validation checks from different points of view.

Miyake observed that there are two types of critiquing: (1) experts criticize novices, and (2) novices criticize experts; and that the latter takes place more often. Criticisms given by the novice forced the expert to try to understand the problem better by applying different points of view. Miyake's research showed that in collaborative problem solving one does not necessarily have to be smarter than the other in order to constructively critique each other.

**Shared Understanding of a Task for Retrieving Relevant Information.** Pollack [1985] observed that in human-human cooperative problem-solving, people often do not know what information they need to obtain in order to achieve their goals. When an expert answers questions asked by novices, therefore, the expert provides more information than what has been literally asked by identifying inappropriate questions and infer the goals behind them.

In human-human communication, both participants can adapt their own behavior according to the characteristics of the partners by gradually gaining shared understanding. The communication process enriches and refines the knowledge of both partners about the task to solve and about each other. The shared understanding enables the partners to improve the communication process, to accelerate the discovery of either common or conflicting goals, to optimize the efficiency of the communication, and to increase the satisfaction of the partners (Oppermann, 1992).

## Inadequateness of Traditional Approach

Traditional AI techniques, expert systems approaches, and information retrieval techniques are not sufficient to support such characteristics in human-computer collaborative problem solving.

The need for coevolution of a problem and a solution implies that a design model that separates problem analyses from solution syntheses (Cross, 1984) (e.g., the waterfall model) is inadequate because it requires problem specification to be completed before starting to form a solution. Thus, any kind of formal system that automates design processes will be unsuccessful because it is based on the assumption that a problem specification never changes once developing a solution has started.

Research has been done in implementing computational critiquing mechanisms (Fischer et al., 1993). Most of such systems put their emphasis on identifying what to critique and are implemented as a *one-shot dialog*, which requires critiquing knowledge to be quite complete. As observed in the user studies described below, when a critic was fired, the subjects often disagreed with it and wanted to argue against the argument underlying the critic. Critiquing is a process, not a one-time event; a critiquing mechanism has to accommodate to users' responses and corrections to the critics (Silverman, Mezher, 1992).

Existing database retrieval systems are built on the assumption that people searching for information know what information they need and how to ask for that information, which does not hold in solving ill-defined problems (Fischer, Henninger, Redmiles, 1991). Because many traditional information retrieval techniques are done in the context of text retrieval systems, their main emphasis is to provide access to many documents as efficiently as possible; little effort has been directed toward helping the user formulate a query (Thompson, Croft, 1989).

## Knowledge Delivery in Integrated Design Environments

We have studied and prototyped *integrated, knowledge-based, domain-oriented design environments* (Fischer, Nakakoji, 1992) for various domains as human-computer collaborative problem solving systems. Two prerequisites for such a design environments to effectively support users are:

- to allow users to coevolve a problem and a solution, and
- to interactively provide users with feedback and information relevant to the task at hand.

A knowledge delivery mechanism is one through which the right knowledge, in the context of a problem or a service, is delivered at the right moment for a human designer to consider (CSTB, 1988). A challenge for designing knowledge delivery mechanisms is how to deliver the *right knowledge*, at the *right time*, in the *right style*.

A partially framed problem and a solution in a design environment represents the user's task at hand to be shared by the system. The knowledge delivery mechanism embedded in such a design environment can use such shared context to determine the relevance to the task at hand, addressing the challenge.

# The KID Design Environment

We have developed the KID (Knowing-In-Design) design environment (Nakakoji, 1993a), which has evolved from the JANUS kitchen floor plan design environment by adding KIDSPECIFICATION. The system is implemented in the CLOS programming language, and runs on Symbolics Genera 8.1.

KID consists of (see Figure 1):

1. KIDSPECIFICATION, which enables an explicit representation of the designer's goals and intentions with respect to the current design;
2. KIDCONSTRUCTION, which provides designers with a palette of domain abstractions and supports them to construct design artifacts using direct manipulation styles;
3. the argumentation-base, which stores design rationale represented in the IBIS structure (Conklin, Begeman, 1988) (i.e., a network of nodes, consisting of issues, answers and arguments); and
4. the catalog-base, which stores completed floor plans (construction) together with associated specifications.

KIDSPECIFICATION and KIDCONSTRUCTION provide the explicit representations of a problem specification and a solu-

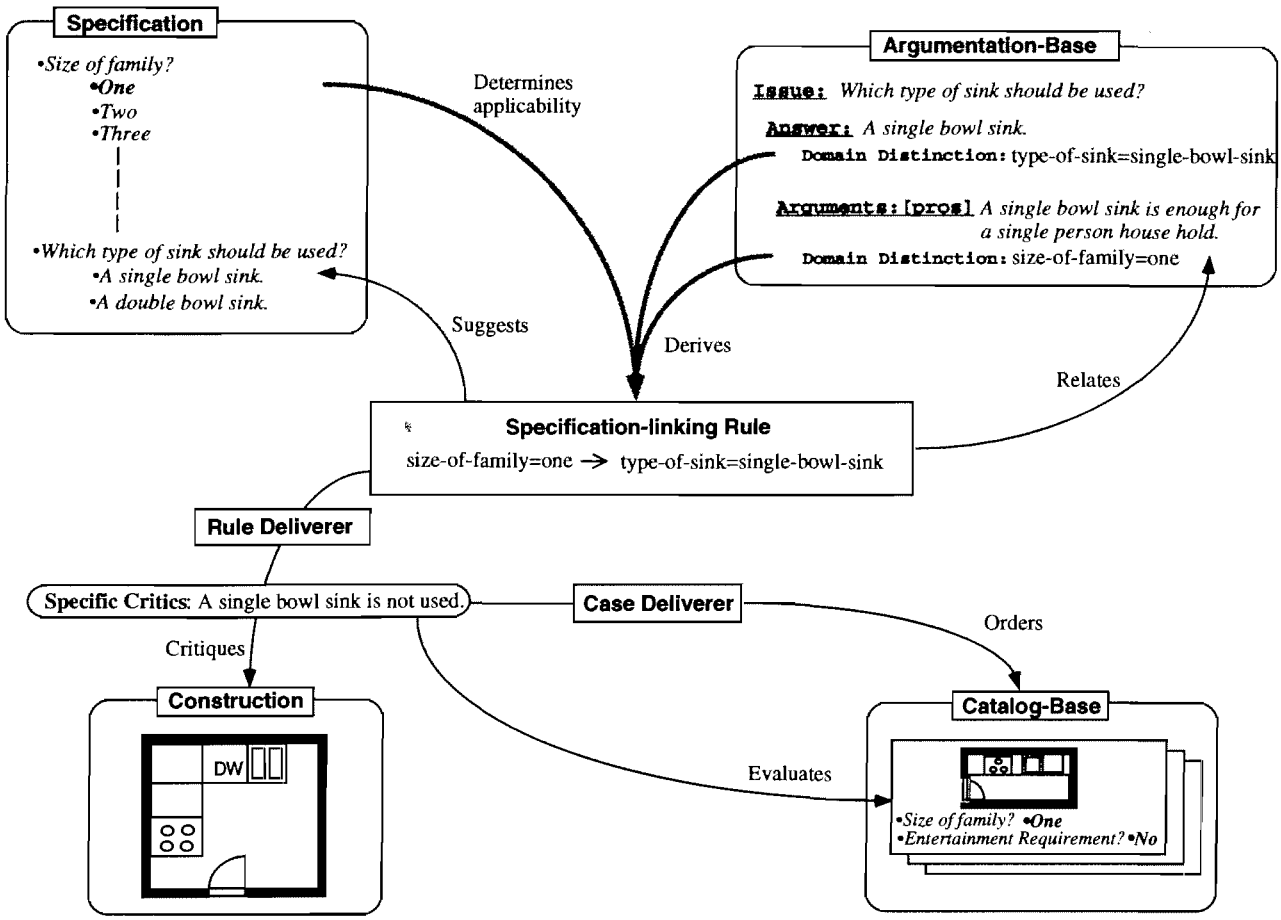


**Figure 1:** Integration of Components of KID via Specification-Linking Rules

Specification-linking rules are derived from the argumentation and a partial specification determines the applicability. Derived specification-linking rules are used (1) to make suggestions in KIDSPECIFICATION, (2) to show a related argument in the argumentation base, (3) to identify relevant critics as specific critics by RULE-DELIVERER, (4) to evaluate a catalog example using the specific critic, and (5) to order the catalog examples by CASE-DELIVERER (see Nakakoji[1993] for detail).

tion construction, which enables KID to satisfy the above two prerequisites.

1. Coevolution of problem specification and solution construction is supported because designers can concurrently reflect on explicitly represented partial specifications (using KIDSPECIFICATION) and constructions (using KIDCONSTRUCTION).
2. Information given through the two components increases the system's shared understanding about the designers' intentions for the current task. Using the shared understanding about the task at hand, KID can deliver task-relevant information for the designers' perusal.

Two knowledge-delivery mechanisms, RULE-DELIVERER and CASE-DELIVERER, are provided by KID. RULE-DELIVERER delivers design principles in a form of argumentation, and CASE-DELIVERER delivers case-based information from the catalog-base. The delivered design knowledge supports designers to reflect on and evolve their partial design.

An advantage of this approach is that increasing quality of information delivery can be achieved at no cost of users. KID delivers task-relevant information without asking users for any additional efforts except specifying a problem with KIDSPECIFICATION and forming a solution with KIDCONSTRUCTION. Because specification of a problem and construction of a solution are essential activities in design, designers do not perceive any additional cost but only benefit in using these two components.

## Mechanism: Specification-Linking Rules in KID

Specification-linking rules represent interdependencies among a partial specification and a partial construction. The rules are used by the two knowledge mechanisms of KID, RULE-DELIVERER and CASE-DELIVERER, to locate and deliver information from the argumentation-base and catalog-base that is relevant to the design task at hand.

**KIDSPECIFICATION.** KID's specification component supports designers in framing their design problems informally — that is, specifying design goals, objectives, criteria, or constraints. Being able to state the problem informally is an essential strategy for dealing with the complexity of design problems. Features such as abbreviation, ambiguity, poor ordering, incompleteness, contradiction, and inaccuracy are frequently observed in human-human cooperative problem solving (Rich, Waters, 1986).

KIDSPECIFICATION has been designed after analyzing questionnaires used by professional kitchen designers to elicit design requirements from clients. The representation of a specification is a set of issue-answer pairs.

KIDSPECIFICATION is a hypertext interface, built on top of the argumentation base. Using KIDSPECIFICATION, designers can specify their design priorities by selecting and annotating alternative design decisions documented in the argumentation-base (see Figure 2).

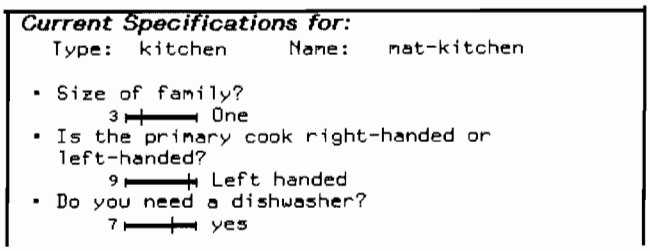


**Figure 2:** A Partial Specification in KIDSPECIFICATION

The summary of currently selected answers in KIDSPECIFICATION is provided in this window. Users can assign weights of relative importance to selected answers by moving associated sliders. In this figure, the user put most importance to the left-handed requirement (i.e., 9 in the 1-10 scale) and little importance to the single-person household requirement. The state of the specification component (i.e., a set of selected answers with assigned weights) is referred to as the current *partial specification.*

Although many of such issue-answer pairs (i.e., design alternatives) will have already been articulated through previous design efforts and have been accumulated by recording design rationale in the argumentation-base, if no prestored alternatives express their position, designers can add or modify information in the underlying argumentation-base using a property sheet. Designers can assign weights to the selected answers to articulate the relative importance of specified items.

The state of the specification component (i.e., a set of selected answers with assigned weights) is referred to as the current *partial specification.* The issues and answers selected in the specification component represent a framework in which to evaluate design moves rather than a list of concrete requirements for the design. The specified goals and requirements are therefore considered partial and subject to modification throughout the design process.

**Specification-Linking Rules.** A specification-linking rule represents a computable interdependency between two issue-answer pairs; for example, *"Size-of-family=one → Type-of-sink=Single-bowl-sink"* implies that there is a relation between the size of a household and the type of sink to be used in the kitchen design. This rule is based on the associated argument to the selection of a type of a sink, which says that a single-bowl-sink is enough for a single-person house-hold.

Specification-linking rules are derived by a mechanism that calculates the design constraints implied by a partial specification (see Figure 1). The above rule is derived by finding issue-answer pairs that are implicated by the specification of *"Size-of-family=one."* In this case, the answer, *"Type-of-sink=Single-bowl-sink,"* to the issue, *"Which type of sink should be used,"* is implied because it is supported by an argument *"A single bowl sink is enough for a single-person household,"* which is associated with *"Size-of-family=one."* The mechanism is described further in Nakakoji [1993].

Because Specification-linking rules are derived dynamically, they enable the system to provide information relevant to a changing design context. Each high-level goal or design criteria expressed in the partial specification triggers the derivation of new specification-linking rules. Designers can easily modify their specification to understand the implications of various design requirements.

**Delivery Mechanisms.** Some of the issue-answer pairs of KIDSPECIFICATION are relating to a construction situation, such as a need for a dishwasher or a type of sink. In order to link the text representation of KIDSPECIFICATION to a graphic representation of KIDCONSTRUCTION, the system provides pre-defined predicates over the construction.

The representation of a construction includes a list of design units used in a partially designed floor plan and their configuration information. The predicates are used to determine whether the associated conditions are satisfied in the partial construction, such as the existence of a single-bowl-sink, or whether the dishwasher is right to a sink. Using a property sheet provided by KIDSPECIFICATION, users can associate one of those predicates with an issue-answer pair in textual representation. When users create a new issue and answer (e.g., Where should a dishwasher be? — within a reach of sink) then they can define a predicate (e.g., *Within-Reach(Dishwasher, Sink)*) by using the MODIFIER system (Girgensohn, 1992).

Thus, when a construction situation is involved in an interdependency implied by a specification-linking rule, the rules can detect inconsistencies between the partial specification and the partial construction. Such inconsistencies are notified to designers in a form of *specific critics* (Fischer et al., 1993).

KIDCONSTRUCTION provides two types of critics that monitor the designer's actions and intervene when a potentially problematic construction situation is detected. *Generic critics* detect violations of general design principles such as building codes, and are applicable to all designs in the domain. *Specific critics*, on the other hand, are only enabled when their applicability is implied by the partial specification.

Specific critics detect construction constraints that are implied by a specification-linking rule but not satisfied in the construction. For example, the specific critic enabled by the above specification-linking rule would check the construction situation for the existence of a single-bowl sink. If this constraint is not satisfied, the specific critic would be fired in KIDCONSTRUCTION.

**RULE-DELIVERER** identifies the collective of specific critics enabled by the partial specification. Through the enabled specific critics, KIDCONSTRUCTION intervenes in the

design process to alert designers to a conflict in the construction situation in terms of the partial specification. This is accomplished by presenting the designer with a simple critic message such as *"a single bowl sink is not used."* The critic message is a mouse-sensitive link to the location of related argumentation (see Figure 1). Selecting the critic message with a mouse accesses the related argumentation, and provides a starting point for browsing the argumentation-base.

**CASE-DELIVERER** uses the set of enabled specific critics to order the precedent cases stored in the catalog-base according to their "appropriateness" to the partial specification (Nakakoji, 1993b). CASE-DELIVERER computes a set of relevant cases by applying the enabled specific critics to the floor plan (construction) of each catalog example. Applying a specific critic to a catalog example means checking whether the example has features that are implied by the partial specification. A weight assigned by the designer to specification items is used to compute the importance value of each satisfied critic rule. The sum of the values of the satisfied critics rules is assigned to the example as an appropriateness value. After computing values for all the catalog examples, CASE-DELIVERER orders the examples according to these values.

### User Observation

KID has been studied by observing several subjects using the system, including both domain-experts and novices. Test sessions were videotaped and the protocols were analyzed. When design knowledge (argumentation via critic rules or ordered catalog examples) is delivered to them, subjects were observed to respond in the following three ways: (1) applied the delivered knowledge to reframe their partial design, (2) explored the related information space to the delivered knowledge, or (3) articulated new design knowledge by arguing against delivered knowledge, or underlying delivery rationale (i.e., *how* the knowledge is relevant to the current task).

KIDSPECIFICATION and KIDCONSTRUCTION enable the KID design environment to have shared understanding of a designer's task at hand, and consequently, KID supports reflection in action (Schoen, 1983) during a design process by delivering the design knowledge relevant to the task at hand. The reflection on their current partial construction and specification was often triggered by ordered catalog examples. Delivery of sometimes *unexpected* information was found to be an effective way to trigger the subjects to reflect on their task at hand. The subjects often discovered new features, which were breakdowns or important considerations they had not been aware of before, by critics presented by RULE-DELIVERER, or in catalog examples presented by CASE-DELIVERER. For the subjects, it was easier to challenge what others (e.g., a computer system) had done than what they had done themselves. Since the judgement of relevance used in delivery is made by KID, the subjects felt free to *critique* the system's judgement. Then, the subjects applied the discovered features to their own design task and often found that their own design had, or lacked, the same features.

The subjects often reacted to delivered knowledge and argued against it in terms of their task at hand. When being given an object to think with, people start thinking about it and trace associations, which may be linked to tacit part of design knowledge (Polanyi, 1966). Thus, it was easier for the subjects to become able to articulate design knowledge, which had been tacit before, than to start articulating design knowledge given no context.

Another benefit of knowledge delivery found was that it encouraged the subjects to explore the system's information space. There is evidence that people search longer for answers to questions when they believe they know the answer (Reder, Ritter, 1992). Thus, high *feelings of knowing* correlate with longer search time. When KID delivered information that was relevant to the task at hand but not quite right, then they gained this "feeling of knowing," which made their information search longer.

## Conclusion

The specification and construction components embedded in a design environment provide the shared understanding between designers and the system. In this context, the expression *machine understanding* implies that the machine reacts or behaves according to human intuitions, expectations, or context. In our work, shared understanding is used to identify the designers' information needs, to locate relevant information in supporting the designers' task at hand, and to deliver the information to the designers. By having the shared understanding, knowledge delivery mechanisms are more tuned toward delivering the right knowledge, at the right time, in the right style (Fischer et al., 1993).

The knowledge delivery mechanisms in KID retrieve information for designers that they might not have been able to access otherwise. The critiquing mechanism of KID gives designers access to related arguments, and KIDSPECIFICATION allows designers to add an argument in response to the accessed arguments. The newly added argument is used to dynamically derive a new specific critic rule using specification-linking rules. Thus, critics in KID partially address the problem of knowledge acquisition. CASE-DELIVERER retrieves case-based information, which provides mapping abstract evaluation criteria to structural constraints that requires design knowledge and expertise (Kolodner, 1991; Bonnardel, 1991).

Although this position paper discusses the approach we have explored to support human-computer collaborative problem solving, it ultimately supports long-term indirect human-human collaboration (Fischer, Nakakoji, Ostwald, 1993). Knowledge delivered by KID is originally input by other designers. In this manner, the approach is a natural implementation of the idea of Winograd and Flores [1986], who argue that computers should be regarded as media that convey human intelligence rather than regarded as yet another intelligent agent.

## Acknowledgements

We thank the HCC group at the University of Colorado, who contributed to the conceptual framework and the systems discussed in this paper. We also thank Barbara Gibbons of Kitchen Connection at Boulder, Colorado, for her valuable time and commenting on our work, and Takanori Hoshi of SRA Inc., for proof reading the manuscript. The research was supported by: the National Science Foun-

dation under grants No. IRI-9015441 and MDR-9253425; the Colorado Advanced Software Institute under grants in 1990/91, 1991/92, 1992/93; US West Advanced Technologies; NYNEX Science and Technology Center, and by Software Research Associates, Inc. (Tokyo).

## References

N. Bonnardel 1991. Criteria Used for Evaluation of Design Solutions, in Y. Queinnec, F. Daniellou (ed.), in Designing for Everyone and Everybody: 11th Congress of the International Ergonomics Asssociation, Paris, France.

J. Conklin, M. Begeman 1988. gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *Transactions of Office Information Systems*, 6(4), October:303-331.

N. Cross 1984. *Developments in Design Methodology*, John Wiley & Sons, New York.

Computer Science and Technology Board 1988. *The National Challenge in Computer Science and Technology*, National Academy Press, Washington, D.C..

G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, T. Sumner 1993. Embedding Critics in Design Environments, *The Knowledge Engineering Review Journal*, (in press).

G. Fischer, S.R. Henninger, D.F. Redmiles 1991. Intertwining Query Construction and Relevance Evaluation, in Human Factors in Computing Systems, CHI'91 Conference Proceedings (New Orleans, LA), 55-62, ACM, New York.

G. Fischer, K. Nakakoji 1992. Beyond the Macho Approach of Artificial Intelligence: Empower Human Designers - Do Not Replace Them, *Knowledge-Based Systems Journal*, 5(1):15-30.

G. Fischer, K. Nakakoji, J. Ostwald 1993. Facilitating Collaborative Design through Representations of Context and Intent, in Working Notes of the AAAI 1993 Workshop on AI in Collaborative Design, AAAI, 293-312, Washington, DC.

G. Fischer, B.N. Reeves 1992. Beyond Intelligent Interfaces: Exploring, Analyzing and Creating Success Models of Cooperative Problem Solving, *Applied Intelligence, Special Issue Intelligent Interfaces*, 1:311-332.

A. Girgensohn 1992. *End-User Modifiability in Knowledge-Based Design Environments*, Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, Also available as TechReport CU-CS-595-92.

J.L. Kolodner 1991. Improving Human Decision Making through Case-Based Decision Aiding, *AI Magazine*, 12(2), Summer:52-68.

N. Miyake 1986. Constructive Interaction and the Iterative Process of Understanding, *Cognitive Science*, 10:151-177.

K. Nakakoji 1993. *Increasing Shared Understanding of a Design Task between Designers and Design Environments: The Role of a Specification Component*, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado, Also available as TechReport CU-CS-651-93.

K. Nakakoji 1993. Case-Deliverer: Making Cases Relevant to the Task at Hand, in Proceedings of the First European Workshop on Case-Based Reasoning, Springer-Verlag, Kaiserslautern, Germany.

R. Oppermann 1992. Adaptively Supported Adaptability, in Sixth European Conference on Cognitive Ergonomics, Human-Computer Interaction: Tasks and Organization (Balatonfuered, Hungary), 255-270.

M. Polanyi 1966. *The Tacit Dimension*, Doubleday, Garden City, NY.

M.E. Pollack 1985. Information Sought and Information Provided: An Empirical Study of User/Expert Dialogues, in Human Factors in Computing Systems, CHI'85 Conference Proceedings (San Francisco, CA), 155-159, ACM, New York.

L.M. Reder, F.E. Ritter 1992. What Determines Initial Feeling of Knowing? Familiarity With Question Terms, Not With the Answer, *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(3).

C.H. Rich, R. Waters (eds.) 1986. *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann Publishers, Los Altos, CA.

H.W.J. Rittel 1984. *Second-Generation Design Methods*, in N. Cross (ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 317-327.

D.A. Schoen 1983. *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York.

B.G. Silverman, T.M. Mezher 1992. Expert Critics in Engineering Design: Lessons Learned and Research Needs, *AI Magazine*, 13(1), Spring:45-62.

H.A. Simon 1981. *The Sciences of the Artificial*, The MIT Press, Cambridge, MA.

A.S. Snodgrass, R. Coyne 1990. *Is Designing Hermeneutical?*, Technical Report, Department of Architectural and Design Science, University of Sydney, Australia.

R.H. Thompson, W.B. Croft 1989. Support for Browsing in an Intelligent Text Retrieval System, *International Journal of Man-Machine Studies*, 30:639-668.

T. Winograd, F. Flores 1986. *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corporation, Norwood, NJ.