DRAFT: TO APPEAR IN KNOWLEDGE-BASED SYSTEMS JOURNAL: SPECIAL ISSUE ON HUMAN-COMPUTER COLLABORATION, 1995

Intertwining Knowledge Delivery, Construction, and Elicitation: A Process Model for Human-Computer Collaboration in Design

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

[1]Practical Software Engineering Laboratory
SRA Inc.,
1-1-1 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan

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

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

Abstract. Collaboration among designers can be described with an ''action-reflection-critique'' model in which the explicit representation of the design contributes to a shared understanding and to the articulation of design knowledge. We describe how domain-oriented design environments based on this model support human-computer collaboration by intertwining knowledge *delivery*, *construction*, and *elicitation*. The KID (Knowing-In-Design) system has a shared understanding about the designers' task at hand through a partial design requirement specification and a solution. KID delivers design knowledge relevant to the task at hand and helps designers uncover and elicit tacit design knowledge, which results in the evolution of the system's knowledge-bases. The evolution affects the system's subsequent behavior by tuning the delivery toward the designers. Through this cycle of knowledge delivery, construction, and elicitation processes supported by KID, designers gradually coevolve a design requirement and a solution.

Introduction

Design tasks are *ill-defined* (Simon, 1981) and *open-ended* (Rittel, Webber, 1984). Much of the expert knowledge required for design remains tacit (Polanyi, 1966). Such design activities are best supported by taking a human-computer collaborative problem-solving approach. In this paper, we describe a model, an architecture, and a prototype system that support designers in collaborating with a computer system. *Integrated, domain-oriented, knowledge-based design environments* (Fischer, Nakakoji, 1992) augment skills of designers instead of generating solutions for designers, as typified by the design automation approach. Design environments are computer systems that provide (1) design media and tools with which designers can represent their design, and (2) intelligent agents that support designers in using the systems' design knowledge for understanding and reflecting on their partial design.

Based on theories of design and models of human-human collaborative problem solving, we have developed the *action-reflection-critique* model for understanding collaboration among stakeholders.[1] Applying this model to human-computer collaborative design has led us to develop an architecture for a design environment that intertwines knowledge *delivery*, *construction*, and *elicitation*. Knowledge delivery is the presentation of information by a design environment relevant to the partially constructed design represented in the environment. Knowledge construction, which is the creation of a mental representation for a piece of knowledge (Kintsch, 1988), takes place in designers' minds (Papert, 1993) in response to an explicit representation of the partial design and the delivered design knowledge. Knowledge elicitation is the articulation of design knowledge by designers responding to the delivered knowledge; designers explicitly represent the knowledge and store it in the system.

In this paper, we illustrate our approach with the KID (Knowing-In-Design) (Nakakoji, 1993) kitchen design environment and demonstrate how the system intertwines knowledge delivery, construction, and elicitation to support designers in coping with design tasks.

A Model for Collaborative Design

Collaboration in design requires not just coordinating divided design tasks; it is also important that designers learn and refine their performance in the course of solving a problem based on the evolving shared understanding.

In order to model collaboration in design and further apply it to human-computer collaborative design, we have integrated design theory with discourse models of human-human collaboration.

The Action-Reflection Model of Design

[1]Stakeholders of a design task include people with various roles, such as designers, clients, and end-users. In order to make our discussion focus on issues of collaboration between people and computers rather than among different people, we use the term ''designers'' to refer to stakeholders in general, and do not distinguish among the different roles of stakeholders.

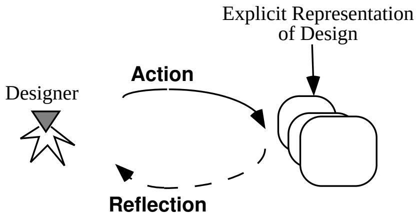

Figure 1: An Action-Reflection Cycle in Design

An action results in modification of an external design representation, whereas reflection occurs in a designers' minds and may not be explicitly represented. External design representations (such as paper, mockups, or computational artifacts) are essential to support designers to recognize unanticipated consequences of their actions and to reflect upon subsequent actions.

According to Schoen's theory, designers work in an alternating cycle of action and reflection (Schoen, 1983). The designer *acts* to shape the design situation by creating or modifying design representations, and the situation ''talks back'' to the designer, revealing unanticipated consequences of the design actions. In order to understand the situation's back-talk, the designer *reflects* on the actions and consequences, and plans the next course of action (see Figure 1).

An Action-Reflection-Critique Model for Collaborative Design

In collaborative problem solving, people are initially unable to articulate complete requirements for problems (Fischer, Reeves, 1992). Through a means of critiquing, which reminds designers of other points of view (Miyake, 1986), people identify portions of the problem that have not yet been understood and refine the solution. In conversation, shared meanings are accrued incrementally, along with evidence of what has been understood so far (Brennan, Hylteen, 1993). People not only respond to what has been articulated immediately before, but also gradually develop a context to be coherent throughout the conversation (Stein, Thiel, 1993).

The action-reflection model illustrates the design activity of a single designer. Applying the action-reflection model to collaborative design requires that the result of reflection be made explicit to maintain a shared understanding among the stakeholders. We call this articulation of the result of reflection a *critique*, and suggest an action-reflection-critique model for collaborative design.

Figure 2 illustrates the action-reflection-critique model of the collaborative design task between designers. In the model, the shared context consists of representations of a a partial design and criticisms including pointing out possible breakdowns in the partial design and articulated design knowledge as relevant to the shared context. Through the cyclic processes, designers both evolve a representation of their design and gradually construct and accumulate criticisms as articulated knowledge. This shared understanding helps the designers coevolve individual understanding of a problem and a solution, and increase the knowledge about the design domain.

Intertwining Knowledge Delivery, Construction, and Elicitation

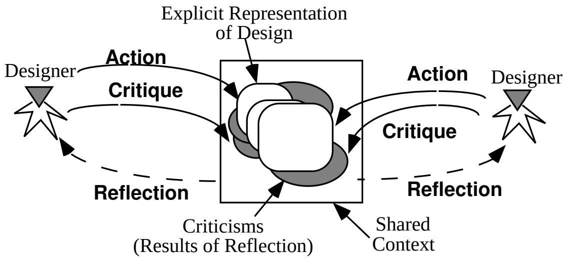

Figure 2: The Action-Reflection-Critique Model in Collaborative Design

Criticisms include pointing out possible breakdowns in the partial design and articulated knowledge as relevant to the shared context.

Our design environments are human-computer collaborative problem-solving systems based on the extension of the action-reflection-critique model. A challenge in applying the model to human-computer collaboration has been to support communication and to establish a shared context between a designer and a computer system.

Design media of a design environment with which designers specify requirements and construct a solution provide a context to be shared by designers and the design environment. Monitoring what designers have been doing with the media allows the design environment to have a shared understanding about the design task. A design environment can critique the partial design, and designers can embed their criticisms within the partial design. The evolving explicit representations of the partial design and criticisms serve as a shared understanding between designers and a design environment.

Figure 3 illustrates the model and describes how the action-reflection-critique model is applied to human-computer collaboration in design tasks. We characterize critique of a design environment as *knowledge delivery*, implying that the system delivers the information relevant to the designers' task at hand in a timely manner without explicit requests to the system. The system's critiquing invokes designers' reflective thinking, and designers may become aware of tacit design knowledge — *knowledge construction* — that will lead them to critique back the system. We characterize designers' criticisms to the current situation as *knowledge elicitation*. The elicited knowledge is taken into account by the system to change the subsequent behavior of the system.

Technical Challenge

As illustrated in Figure 3, a design environment consists of two major parts: a design medium and an intelligent agent. The intelligent agent interacts with designers by delivering knowledge from a knowledge-base and changes and tunes its behavior according to the knowledge elicited by the designers during a design process. In this section, we focus our discussion on technical challenges of implementing knowledge delivery and elicitation mechanisms.

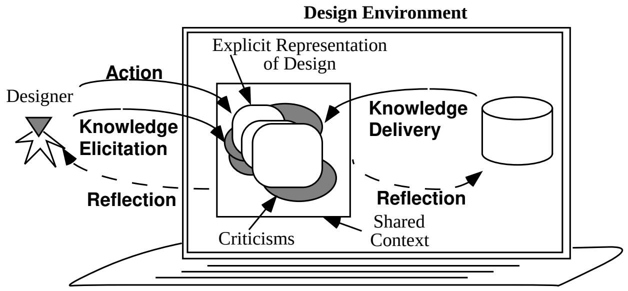

Figure 3: Applying the Action-Reflection-Critique Model to Human-Computer Collaboration

A design environment plays roles both as a design medium and as an intelligent agent that collaborates with designers. In our approach, because a design environment does not produce or modify a design representation automatically, there is no action arrow from the system. The partial design constructed in the design environment provides the shared context between the designer and the design environment. Based on the identified shared context, the system delivers the information related to the context. Designers may become aware of tacit design knowledge as relevant to the context, and elicit the knowledge into the system as criticisms. Reflection by the system means that the results of elicitation are taken into account by the system to modify its subsequent behavior.

Requirements for Knowledge Delivery

Challenges for effectively implementing knowledge delivery mechanisms include how to deliver the *right knowledge* at the *right time*.

Whether a piece of knowledge is ‘‘right’’ can be determined in terms of the relevance of information to the current task. If the system provides a medium with which designers can represent their design, this partial design can be used to form a model of the task at hand. Using this model, the delivery mechanisms can retrieve relevant information from the knowledge-base. Determination of the relevant portions of knowledge presents a challenge because different design situations may need to view a piece of knowledge differently; thereby the definition of relevance changes dynamically. It is impossible to anticipate all possible design situations a priori (Suchman, 1987), which makes a static indexing scheme for design information of the knowledge-bases inapplicable.

Delivery mechanisms continuously monitor user actions and intervene with them when a potential information need is detected. They are especially effective because delivered information is related to a specific condition in the current design. Intervention immediately after a suboptimal or unsatisfactory action has occurred has the advantage that the problem context is still active in the designer’s mind and the designers still know how they arrived at the problematic situation (Fischer et al., 1993). A challenge for designing such delivery mechanisms is how to intervene with users in a non-distracting manner in order to prevent short-term memory loss.

Requirements for Knowledge Elicitation

It is crucial that the elicitation of knowledge is taken into account by the system to refine its subsequent behavior. In collaborative design, designers adapt their own behavior according to the evolving shared understanding (Pollack, 1985). Our experience of building a design rationale system shows that people are not motivated to elicit knowledge if they do not see immediate benefit by doing so (Fischer et al., 1991).

A challenge for implementing such a knowledge elicitation mechanism is that the newly created representation must be manipulatable by the system. This requires designers to represent the knowledge in the system’s languages — often formal and very different from designers’ domain languages. Forcing designers to state their design knowledge using a knowledge representation language undermines their expressive ability. One approach for this problem is to ask knowledge engineers to formulate their knowledge whenever they want to elicit knowledge into the system, but this is not a feasible solution because there are risks of erroneous interpretation by knowledge engineers (Bonnardel, 1993), and it cannot be expected that knowledge engineers will be available throughout the lifetime of system usage.

Our approach is to support the evolutionary growth of a knowledge-based system through a seeding, evolutionary-growth, and reseeding cycle (see Figure 4) (Fischer et al., 1994). The system is first "seeded" by knowledge engineers in collaboration with designers (end-users), with several mechanisms that would allow designers to represent their knowledge during the use of the system. As a second phase during use, the designers elicit their knowledge into the system using the mechanisms, "gradually evolving" the knowledge-base. Such mechanisms must allow designers to be able to represent their thoughts, ideas, and arguments directly and distinctly using their own language. End-user modifiability (Fischer, Girgensohn, 1990) of a design environment has been studied to allow designers to elicit computer-manipulatable design knowledge without using underlying knowledge representation languages. Lastly, the system is occasionally "reseeded" by knowledge-engineers in collaboration with the designers to reorganize knowledge constructed by the designers that might have introduced inconsistencies or insufficient formalisms.

The KID Design Environment

We have developed the KID design environment to instantiate our approach. KID is implemented in the CLOS programming language on Symbolics Genera 8.1. It consists of the four major components:

1. KIDSPECIFICATION, which allows designers to specify their design requirements and intentions (Figure 5);
2. KIDCONSTRUCTION, which provides designers with a palette of domain abstractions and supports them to construct design artifacts using direct manipulation styles (Figure 6);[2]

[2]Note that we use the term "construction" with two different meanings. KIDCONSTRUCTION supports designers to construct a floor plan, whereas knowledge construction refers to mentally constructing a representation of design knowledge.

3. an 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) (see Figure 5); and
4. a catalog-base, which stores completed floor plans (construction) together with associated specifications (see Figure 6).

Designers can coevolve a problem and a solution by going back and forth between KIDSPECIFICATION and KIDCONSTRUCTION. We define the representations given through the two components as ''a partial design.'' The partial design provides KID with a shared understanding about the designers' task at hand. KID delivers knowledge relevant to the task at hand by indicating potential breakdowns in the partial design and retrieving useful cases from the catalog-base for their design. The delivery helps designers not only refine the partial design but also become aware of tacit design knowledge by evoking their reflective thinking. Designers may elicit this uncovered knowledge by presenting it as argumentation or as new design objects, which results in the evolution of the system's knowledge-bases. KID dynamically takes this evolution into account and refines its behavior while supporting designers.

Through iterating this process, the designers gain an understanding of their design task as well as knowledge about the domain, and articulate and accumulate design knowledge into the system. The system tunes its behavior and delivery according to the gradually evolving shared understanding and the constructed knowledge.

Scenario

A kitchen designer, Jeff, designs a kitchen floor plan. Jeff specifies requirements for his design task using KIDSPECIFICATION (Figure 5). Jeff starts constructing a floor plan using KIDCONSTRUCTION (Figure 6). When he puts a dishwasher on the right side of a double-bowl sink, a critic fires, notifying him that he should put the dishwasher on the left side of the sink (see the *Message* window in Figure 6). Wondering why, he clicks on the critic message. The corresponding argument is presented, describing that a kitchen should have a dishwasher on the left side of a sink because he specified that this kitchen is for a left-handed cook (see Figure 5). In fact, catalog examples, each of which has a dishwasher on the left side of a sink, have been suggested by the system in the *Catalog* window (Figure 6) to be most appropriate for his design requirements specified in KIDSPECIFICATION. He reflects on the argument and thinks about specializing the kitchen for a left-handed person. Then he remembers that the resale value of the kitchen is actually a very important concern. He adds this requirement using KIDSPECIFICATION, and creates an argument that having a dishwasher on the left side of a sink may affect the resale value (see the *Argumentation* window in Figure 5). The system takes this argument into account and defines a new interdependence between the location of a dishwasher with regard to a sink and the concern for a resale value. In the *Catalog* window, the catalog examples are automatically reordered according to this newly defined interdependence, showing a catalog example that has a dishwasher on the right side of a sink on top of the list (not shown in the figures).

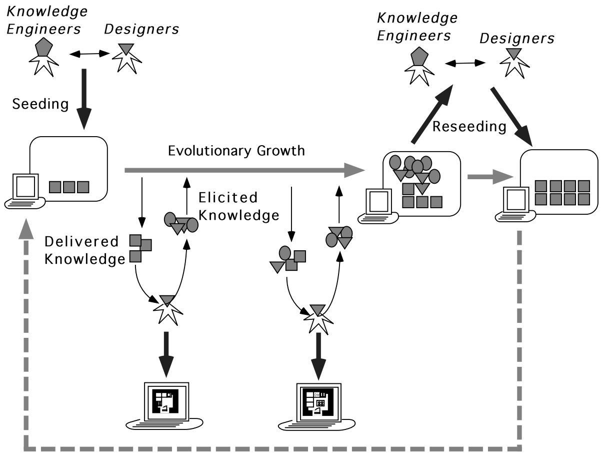

Figure 4: A Seed, Evolutionary Growth, and Reseeding Cycle of Development Processes

Through the use of a design environment, designers gradually elicit knowledge by designing and responding to the knowledge delivery. Some part of the added knowledge may be inconsistent or not totally manipulatable by the system. During the reseeding process, knowledge engineers, with the help of designers, can examine the knowledge-base and reconstruct it. Support for seeding and reseeding is discussed in Fischer et al. [1994].

Architecture and Mechanisms of KID

Figure 7 illustrates the architecture and mechanisms of the KID design environment. KID delivers three types of design knowledge from its knowledge-bases:

1. Catalog examples are ordered according to the appropriateness to the current specification.
2. Critic messages notify potential conflicts in the floor plan in terms of the current specification and building code.
3. Argumentation describes alternatives to design decisions, interdependences among design decisions, and how the critic messages are related to the current specification.

KID allows designers to elicit domain knowledge in the following manner:

1. to add arguments to the argumentation-base;
2. to add, modify, or delete a question and answers in KIDSPECIFICATION;
3. to store completed specification and construction as a catalog example;
4. to add, modify, or delete palette items in KIDCONSTRUCTION; and
5. to define or redefine a construction-analyzing rule: for example, to change the definition of ''near'' from 20 inches to 18 inches, or define a new relationship, such as ''facing.''

Both delivery and elicitation mechanisms use *specification-linking rules*. These rules represent interdependences among domain concepts and are used to infer the relevance between different types of design representations. Each rule is derived from an argument stored in the argumentation-base.

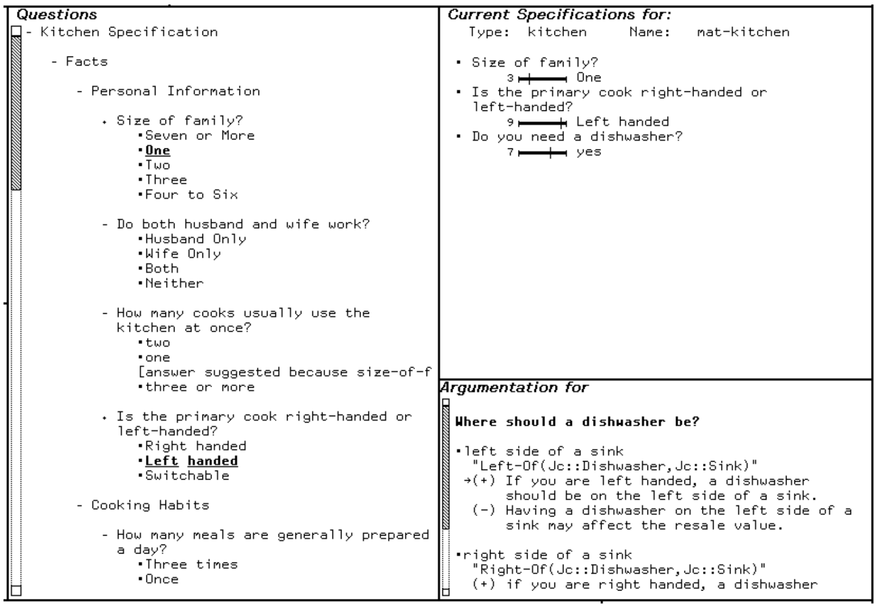

Figure 5: KIDSPECIFICATION

Designers can select answers presented in the *Questions* window. The summary of currently selected answers appears in the *Current Specifications* window. Each answer is accompanied by a slider that allows designers to assign a weight representing the relative importance of the answer (e.g., most importance to the left-handed cook requirement (i.e., 9) and little importance to the single-person household requirement (i.e., 3)). The *Argumentation For* window provides further explanation about how a presented critic message (i.e., a location of a dishwasher with regard to a sink) (see Figure 6) is related to the current specification (i.e., one of the selected answers - a left-handed cook), as well as alternatives for the location of a dishwasher including the one Jeff has just created.

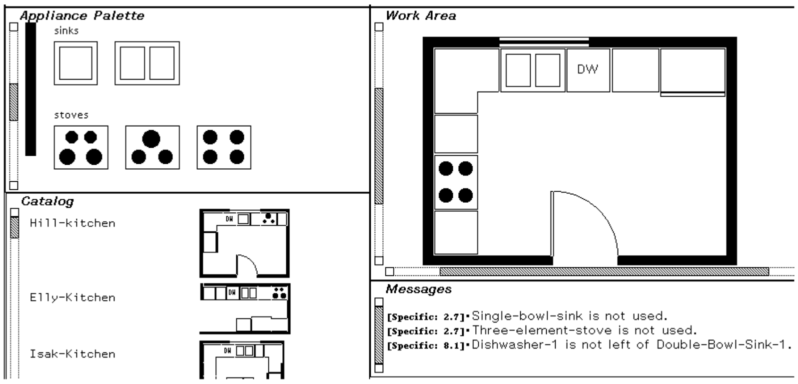

Figure 6: KIDCONSTRUCTION

Designers construct a kitchen floor plan in the *Work Area* using a direct manipulation style to select and place design units from the appliance palette. Designers may copy an example from the *Catalog* window, where catalog examples are presented in the order of accordance with the current specification (see Figure 5). The *Messages* window presents critiquing messages that are detected by KID. Numbers indicate computed relative importance of each critiquing message in terms of the current specification.

The argumentation-base is structured as a network of nodes consisting of issues, answers, and arguments. An issue-answer pair represents a design decision, or a domain concept, in terms of function, structure, or behavior at various levels of abstraction. An interdependence between two domain concepts (i.e., issue-answer pairs; for example, *‘‘Size-of-family=one’’* requires *‘‘Type-of-sink=Single-bowl-sink’’*) can be captured through the associated argument (e.g., *‘‘a single-bowl-sink is enough for a single-person household’’*). By associating a predefined vocabulary of domain concepts to issue-answer pairs and arguments, KID can compute an interdependence between the two concepts. A collection of specification-linking rules forms an interdependence network among domain concepts.

Knowledge Delivery in KID

The *Delivery Manager* of KID monitors KIDCONSTRUCTION and KIDSPECIFICATION and reformulates the interdependence network represented by specification-linking rules each time a modification is made in the two components. Because KIDSPECIFICATION has been built as a hypertext interface to the argumentation-base, selecting answers in KIDSPECIFICATION is reflected in reorganizing the

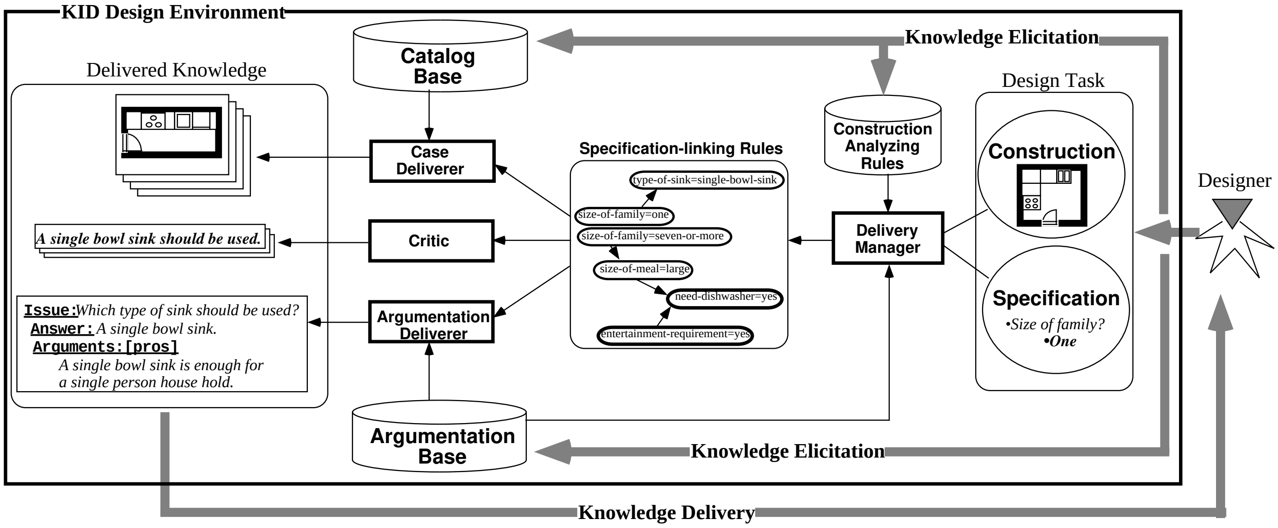

Figure 7: Knowledge Delivery and Construction in KID

A designer performs a design task through specification and construction components. The partial specification and construction provides KID with shared understanding of the task at hand. Using this understanding, a delivery manager of KID derives specification-linking rules from the argumentation-base. These rules are used by delivery mechanisms (Case Deliverer, Critic, and Argumentation Deliverer), which deliver corresponding knowledge for the designer. The designer can elicit knowledge by adding design objects to the catalog-base, by adding arguments to the argumentation-base, and by adding construction-analyzing rules using MODIFIER.

specification-linking rules. In order to reflect the current construction to the formation of the specification-linking rules, the delivery manager consults with the construction-analyzing rules that can parse the *Work Area* of KIDCONSTRUCTION and identify notable characteristics, such as whether a dishwasher is on the left side of a sink.

The interdependence network formulated from specification-linking rules is used by three mechanisms to deliver three types of design knowledge. The *Case Deliverer* mechanism applies the rules to each catalog example, computes an appropriateness value for each example, and orders the examples in the *Catalog* window by the computed values. The *Critic* mechanism applies the rules to the current construction, detects conflicts in the construction in terms of the rules, computes the importance of each conflict in terms of the current weighted specification, and presents critiquing messages with a number indicating the importance in the *Message* window. Finally, the *Argumentation Deliverer* mechanism maintains the derivation of specification-linking rules from the argumentation-base, and presents related arguments in the *Argumentation* window when designers request further explanation for the presented critiquing messages. The mechanisms are detailed in Nakakoji [1993].

Knowledge Elicitation in KID

KID provides a property sheet with which designers can add (or modify) an issue, an answer, or an argument to the argumentation-base as a simple textual statement. This allows designers to add questions and answers in KIDSPECIFICATION because the component is an interface to the argumentation-base.

In order for KID to be able to manipulate the content of these statements so that the elicited knowledge affects the system's subsequent behavior, the property sheet also allows designers to define domain concepts and associate one of them to a textual statement. For example, the statement ‘‘*Is the primary cook right-handed or left-handed?* — left-handed’’ is associated with ‘‘*left-right=left.*’’ When an answer is related to a condition over a floor plan, designers may relate the answer to one of pre-defined construction-analyzing rules. For example, the text description ‘‘*a dishwasher should be on the left side of a sink*’’ is associated with ‘‘*Left-of(Dishwasher, Sink),*’’ which is a Lisp predicate that can parse a constructed floor plan in the *Work Area* window and return true or false accordingly. Furthermore, if a statement for an argument is associated with a related domain concept, for example, as Jeff in the scenario associates ‘‘*Having a dishwasher on the left side of a sink may affect the resale value*’’ with ‘‘*resale-value-concern=important,*’’ then the Delivery Manager automatically derives a specification-linking rule (‘‘*resale-value-concern=important*’’ → ‘‘*not Left-of(Dishwasher, Sink)*’’) and reformulates the interdependence network.

KID allows designers to store their specification and construction into the catalog-base with the the *Store into Catalog* command. The catalog-base is reconstructed on the fly. This allows designers to store several versions of their design as catalog examples.

The MODIFIER system (Fischer, Girgensohn, 1990) allows designers to create or modify design objects of KIDCONSTRUCTION, including palette items and construction-analyzing rules. MODIFIER provides a property sheet interface to designers for a design object that is internally represented as a CLOS object. MODIFIER helps designers to edit the attributes of the object by making suggestions and providing help messages.

User Study

KID has been studied by observing ten subjects using the system, including both domain-experts and novices. Test sessions were videotaped and the protocols were analyzed. When design knowledge was delivered, subjects responded in one of the following ways: (1) they applied the delivered knowledge to refine the partial design, (2) they explored the related information space to the delivered knowledge, or (3) they elicited new design knowledge by arguing against the delivery.

Active knowledge delivery was often found to trigger the subjects to reflect on their partial design. Subjects often discovered breakdowns or important considerations about which they had not previously been aware. Subjects often reacted to delivered knowledge and argued against it in terms of their task at hand.

An unexpected benefit of knowledge delivery 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). High *feelings of knowing* correlate with longer search times. When KID delivered information that was relevant to the task at hand but not quite right, the subjects then gained this "feeling of knowing," which made their information search longer.

Discussion

Information overload problems. The large and growing discrepancy between the amount of potentially relevant information and the amount any one designer can know and remember limits the ability of designers to take advantage of high-functionality knowledge-based design environments (Fischer, Henninger, Nakakoji, 1992). Designers may not know what information they need, how to ask for that information, or that potentially useful information exists in the system.

The knowledge delivery mechanisms of KID address this problem. The system uses the partial design represented in the system to identify the designer's information needs and to deliver information relevant to the task at hand. The partially identified current task through KIDSPECIFICATION and KIDCONSTRUCTION can be used as queries submitted to information retrieval mechanisms, providing the background context for an information search (Fischer, Nakakoji, 1991).

Knowledge acquisition problems. People are good at responding to a given situation, but it is difficult for domain experts to start articulating domain knowledge given no context.

Delivering knowledge by a design environment stimulates designers to elicit tacit knowledge (Brown, Duguid, 1992). In the user study, we have observed that the subjects successfully articulated their criticisms to what the system had done. They often disagreed with the rationale behind KID's delivery, and started to articulate argumentation. They often wanted to store the arguments into the system and to define a new interdependence between design concepts ''to see what would happen.''

Long-term indirect collaboration among designers. An argumentation-base and catalog-base of a design environment facilitate long-term indirect communication among designers (Fischer et al., 1992). Designers communicate through accrued design representations in a design environment instead of talking or writing directly. A portion of argumentation articulated and stored by a designer can be used to produce a specification-linking rule, which helps other designers at other times to solve other design problems. Designers store a design into a catalog-base, which helps other designers to produce ideas for a solution. Thus, a design environment that intertwines knowledge delivery, construction, and elicitation supports designers to naturally carry out design as an argumentative process (Rittel, Webber, 1984) among designers over a long period of time.

Collaborative intelligent agents. Knowledge delivery and elicitation mechanisms embedded in a design environment make the system an intelligent collaborative problem-solving agent rather than just a construction kit, a database, or a design analysis tool. The partial design through KIDSPECIFICATION and KIDCONSTRUCTION serves as a shared understanding between designers and the design environment. The delivery mechanisms of a design environment monitor a designer engaged in a design task *''over the shoulder,''* and present information relevant to the task at hand without the designer's explicit requests. Knowledge elicitation mechanisms allow designers to respond to the delivery, evolving the partial design and the system's knowledge-bases. The system refines their subsequent behavior according to this increased shared knowledge about both the problem and the domain.

Thought-provoking mechanisms. Finally, it is noteworthy that delivering *not-quite-right* knowledge may also help designers. We observed during the user study that even when designers found the delivered information to be of little or no relevance, they often were willing to attend to the delivered knowledge and modify the knowledge-base by adding new design principles or concepts in response to it, leading to knowledge elicitation. A computational agent does not always have to provide the *right* answer to the problem at hand; it can be thought provoking, as a novice's critiquing enhanced an expert's performance in Miyake's study (Miyake, 1986). The system's delivery of both ''right'' and ''not-quite-right'' knowledge reminds designers of potentially important design information, and invokes them to uncover their tacit knowledge.

Conclusion

The specification and construction components embedded in KID provide the shared understanding between designers and the system. In our work, shared understanding is used to identify the designers' information needs, to locate relevant information to support the designers' task at hand, and to deliver the information to the designers. By having shared understanding, knowledge delivery mechanisms are more tuned toward delivering the right knowledge at the right time. The delivered knowledge helps designers uncover breakdowns in a partially represented design, provokes designers in their reflective thinking, and helps them to become aware of tacit design knowledge and elicit it. The elicited knowledge by designers contributes to the evolution of a knowledge-base, and the system tunes their subsequent behavior based on the evolving shared understanding.

We have presented a model for supporting human-computer collaborative design and discussed an architecture for the model. The study of KID, which has been built based on the architecture, has demonstrated that intertwining knowledge delivery, construction, and elicitation is critical for supporting human-computer collaboration.

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 her comments on our work. We thank Loren Tarveen of AT&T Bell Labs, for his comments and suggestions on an earlier version of the paper. The research was supported by the National Science Foundation under grants No. IRI- 9015441 and MDR-9253425; Software Research Associates, Inc. (Tokyo); the Colorado Advanced Software Institute; US WEST Advanced Technologies; and NYNEX Science and Technology Center.

References

N. Bonnardel 1993. Expertise Transfer, Knowledge Elivitation, and Delayed Recall in a Design Context, *Behaviour and Information Technology*, 12(5):304-314.

S.E. Brennan, E.A. Hulteen 1993. Interaction and Feedback in a Spoken Language System, in Working Notes of the AAAI Fall Symposium Workshop on Human-Computer Collaboration: ReconcilingTheory, Synthesizing Practice, AAAI, 1-5, Raleigh, NC.

J.S. Brown, P. Duguid 1992. *Enacting Design for the Workplace*, in P.S. Adler, T.A. Winograd (eds.), *Usability: Turning Technologies into Tools*, Oxford University Press, New York, NY, 164-197.

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

G. Fischer, A.C. Lemke, R. McCall, A. Morch 1991. Making Argumentation Serve Design, *Human Computer Interaction*, 6(3-4):393-419.

G. Fischer, J. Grudin, A.C. Lemke, R. McCall, J. Ostwald, B.N. Reeves, F. Shipman 1992. Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments, *Human Computer Interaction*, *Special Issue on Computer Supported Cooperative Work*, 7(3):281-314.

G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, T. Sumner 1993. Embedding Critics in Design Environments, *The Knowledge Engineering Review Journal*, 8(4), December:285-307.

G. Fischer, R. McCall, J. Ostwald, B. Reeves, F. Shipman 1994. Seeding, Evolutionary Growth and Reseeding: Supporting Incremental Development of Design Environments, in Human Factors in Computing Systems, CHI'94 Conference Proceedings (Boston, MA), 292-298, ACM.

G. Fischer, A. Girgensohn 1990. End-User Modifiability in Design Environments, in Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA), 183-191, ACM, New York.

G. Fischer, S. Henninger, K. Nakakoji 1992. *DART: Integrating Information Delivery and Access Mechanisms*, Unpublished Manuscript.

G. Fischer, K. Nakakoji 1991. Making Design Objects Relevant to the Task at Hand, in Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 67-73, Cambridge, MA.

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, 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.

W. Kintsch 1988. The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model, *Psychological Review*, 95:163-182.

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.

S. Papert 1993. *The Children's Machine*, Basic Books, New York.

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).

H.W.J. Rittel, M.M. Webber 1984. *Planning Problems are Wicked Problems*, in N. Cross (ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 135-144.

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

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

A. Stein, U. Thiel 1993. A Conversational Model of Multimodal Interactoin in Information Systems, in Proceedings of AAAI-93, Eleventh National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 283-288, Washington, DC.

L.A. Suchman 1987. *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK.