Supporting the Evolution of Design Artifacts with Representations of Context and Intent

Gerhard Fischer, Kumiyo Nakakoji and Jonathan Ostwald

Abstract

The design of complex artifacts is essentially an evolutionary process that requires collaboration among stakeholders. Domain-oriented design environments (DODEs) support the evolution of artifacts both by individual designers and by designers participating in long-term, indirect collaboration. DODEs provide representations for generic and specific levels of context. This context supports individual designers by making the information space relevant to the current design intent, and long-term collaboration among designers by allowing them to ground their communication around design artifacts. We demonstrate our approach using the KID system, articulate principles for representations of context and intent, and discuss various approaches to represent intent and context in design environments.

Keywords:

domain-oriented design environments, shared context, explicit representations for intent, communication of intent, evolution of design artifacts, knowledge-based information delivery, long-term indirect collaboration

Introduction

The design of complex artifacts is essentially a collaborative and ongoing process. The need for collaboration stems from the fact that the knowledge required to understand and solve problems spans many fields and is distributed among many stakeholders @cite[GreenbaumKyng1991, Walz.etal1994]. The need for ongoing design stems from the fact that complex design problems are ill-structured @cite[Simon1981], meaning that one cannot completely understand design requirements before making significant steps toward the solution. There will always remain

some requirements that can only be recognized when the artifact is used @cite<HendersonKyng1991>. Design theorists advocate iterative problem-solving processes that emphasize communication between designers and are grounded by design representations. For example, Rittel @cite[Rittel1984] views design as an argumentative process, and Schön @cite[Schoen1983] sees design as a conversation with the materials of the design.

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

Complexity in design arises from the need to synthesize different perspectives on a problem, to manage large amounts of information potentially relevant to a design task, and to understand the design decisions that have determined the possibly long-term evolution of a designed artifact. Our approach to supporting design with computers focuses on human-centered design support with domain-oriented design environments (DODEs). Design environments are human-computer collaborative problem-solving systems @cite<Terveen1995> 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 upon their emerging design artifacts. Rather than modeling the cognitive processes of designers, DODEs augment the abilities of designers to understand, manage, and communicate complexity.

This paper argues that design environments must enable the evolution of design artifacts by supporting collaboration between designers. Design environments support collaboration by serving as a media for communication. Because design environments house the evolving artifacts, they provide a context for communication. As representations to communicate intent accumulate in the design environment, they, in turn, contribute to the evolution of context that supports subsequent collaboration.

Comment: A construction component of a DODE allows designers to represent design intent in a solution form. Semantics of domain objects are encoded and interpreted in a universal manner. A specification component of a DODE allows designers to represent design intent at abstract level. Integration of those multiple representations serve as context that designers are being engaged. Using the representations in the components, the system delivers design knowledge relevant to the current designer's intent.

Representing design context and intent is not a well-defined problem. We have applied our DODE approach to several domains and developed the SER (Seeding, Evolutionary Growth, Reseeding) model to develop such representations. @secref<ser-model> argues that constructing such representations (i.e., a DODE itself) is yet another design task, and discusses several approaches.

In this paper, we first describe the evolution of design artifacts and the necessity of communication of design intent. Then, we discuss how our DODEs support the communication of intent with an example scenario using the KID (Knowing-in-Design) design environment. @secref<others> discusses issues in representations for context and intent and presents other approaches.

Evolution of Design Artifacts Through Communication of Intent

This section takes a close look at the relation between the evolution of design artifacts and the communication between collaborative designers. **In particular, we look at communication in terms of *intent* (the meaning behind the message) and *context* (the background against which the message is articulated and understood).** We will first describe everyday communication and then we will describe communication in design, mediated by design environments.

In everyday interpersonal communication, intent is often casually articulated and understood against a rich background of shared experience and circumstances. To describe the process of communication of intent, we use the terms "speakers" and "listeners" to refer to two roles: those who articulate their intent and those who try to understand (assign a meaning to) the articulated representation, respectively.

Speakers express their intent by implicitly using background context. Listeners, using their own context, try to assign a meaning to the representation. Ideally, the speaker's intent behind the representation and the assigned meaning by the listener should be the same. However, if the listener uses a context that is different from the one that the speaker used (and this is true for most cases) when interpreting the representation, miscommunication occurs.

As depicted in @figref<speaker-listener>, each stakeholder has a different context, as each person has a different understanding of a problem. Such context is vague and cannot be completely describable or expressible. Communication breakdown occurs when a speaker and a listener have little shared context because the assigned meaning to the representation by a listener mismatches to the original meaning of the representation that the speaker intended. As illustrated in @figref<speaker-listener>, the more context the speaker and the listener share (i.e., intersection), the less mismatch between the speaker's intent and assigned meaning by the listener.

Postscript="/homes/kumiyo/doc/DIS95/speaker-listener.eps",
The Shared Context and Communication of Intent in Design
People use background context to represent intent and understand the representation. If a speaker and listener have different context, communication breakdown occurs because a listener assigns a meaning to a representation that is different from the original meaning of the representation the speaker intended. The more context they share, the smaller the communication gap becomes.

In contrast to everyday interpersonal communication, designers express their intentions using explicit design representations. According to Schön's theory, designers work in an alternating cycle of action and reflection @cite<Schoen1983>. 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. Thus, designers are speakers when they act on a design representation, and listeners when they reflect on the representation. This interaction between designers as speakers and designers as listeners drives the evolution of artifacts.

Comment: As Sharples has noted, making ideas explicit "is not a matter of emptying out the mind but of actively reconstructing it, forming new associations, and expressing concepts in linguistic, pictorial, or any explicit representational forms" @cite<Sharples1993>. Therefore, design representations created by designers do not necessarily reflect their intentions. The implication for collaborative design and for design artifacts that evolve over a long period of time is that "design intentions as well as design solutions should be made as explicit as possible".

Complex design may span over a long period of time, and even expert designers cannot attend to all facets of a complex design task. Externally articulating a partially understood design intention is equally as important as externalizing the partial solution. As Sharples has noted, making ideas explicit @quote<is not a matter of emptying out the mind but of actively reconstructing it, forming new associations, and expressing concepts in linguistic, pictorial, or any explicit representational forms> @cite<Sharples1993>. When designers pause for reflection, they are directing attention toward an intimate relationship between the partially understood problem and solution. Because intentions become context for subsequent design moves, their original meaning are easily lost if not made explicit.

Comment: missing: communication from design environment is interpreted against a context that includes representations that were created as representations of intent in the past!!

Computer support for ongoing and collaborative design must provide representations of both intent and context in order for meaning to be communicated from speaker to listener over time. Representations created to communicate intent in the past can be reused as context to understand design problems in the present. The representations that are created to communicate intent in the present become part of the evolving context for the future because they are attached to artifacts. As part of the evolving context, these representations support interpretation of subsequent communication, which, in turn, also becomes part of the evolving context. Thus, the evolution of design artifacts is both driven by, and supported by, communication of intent.

In summary, supporting the communication of intent in design environments requires that (1) intent is explicitly represented, and (2) that representations of intent are accumulated and reused as context for understanding subsequent communication of intent. The next section describes how our DODEs support this communication of intent.

DODEs: Explicit Representation of Context

In our work, we have created collaborative systems named *domain-oriented design environments* in support of design. As the name suggests, DODEs are built to support design in a specific domain, such as kitchen design or LAN design.

DODEs are built upon a domain-independent architecture that provides a structure for domain knowledge, and mechanisms for delivering knowledge as it is needed to support design. We have developed our domain-independent architecture through numerous attempts to create domain-oriented design environments (for details see @cite<Fischer1992b>). The architecture consists of the following five components: (1) a construction component, (2) an argumentation component, (3) a catalog of interesting design examples, (4) a specification component, and (5) a simulation component. The individual components are linked by knowledge-based mechanisms: a construction analyzer (built as a critiquing system @cite<Fischeretal.1993b>), an argumentation illustrator and a catalog explorer.

DODEs instantiate the architecture for a particular design domain (e.g., the domain of kitchen design). A DODE instantiated to support kitchen design contains a construction component for constructing a floor layout, an argumentation component containing issue-based information relevant to kitchen design, a catalog for storing kitchens designed using the system, and a specification component for specifying abstract kitchen characteristics.

DODEs provide feedback to designers as they design, rather than requiring designers to construct a final product before receiving feedback. In this way, DODEs help designers to evolve designs and to understand the effects of individual design moves. The interaction between a designer and a DODE can be seen as a conversation in which the designer speaks by making a design move and listens to the feedback provided by the environment. Conversely, the DODE listens to the designer's design moves and speaks by providing feedback.

In what follows, we first describe how DODEs support generic and specific levels of context, and then we describe how DODEs use the context to support both individual and collaborative aspects of design through communication of intent, followed by the demonstration of our approach using the KID system.

Two Levels of Shared Context

DODEs provide an explicit context for design at two levels: (1) a DODE provides context as a generic domain-level substrate, and (2) a specific design artifact, developed using the generic substrate, serves as an evolving context.

Because a DODE is tuned to support design of artifacts for a particular domain, there is a generic shared context that provides the initial basis for communication between designers and a DODE. When designers first begin a design, the DODE can interpret the designers' actions only against this generic context. As designers construct the design, they are incrementally representing their intentions for *this* artifact. As the design progresses, the designers make more and more intent explicit, and the shared context for the artifact becomes more specific.

The Use of Shared Context

The context provided by DODEs supports:

- individual designers by making the information space relevant to the current design intent, and

- long-term collaboration among designers by allowing them to ground their communication around design artifacts @Cite<ReevesShipman1992>.

DODEs offer great capacity for storing large volumes of information and integrating diverse information sources, such as solutions to previous design problems and collections of argumentation. However, access to large information spaces creates a new problem for designers: information overload. In situations of information overload, the critical resource for designers is not information, but rather the attention needed to process information. When presenting designers with information, the primary concern is to present items that are relevant to the task at hand @Cite<FischerNakakoji1991>.

The specific context defined by the construction and specification components allow the system to provide information relevant to a dynamic context that is shared by the designer and the design environment. This shared context enables precise intervention by the system's knowledge delivery mechanisms (e.g., critics @Cite<Fischeretal.1993b>), reduces annoying interruptions, and increases the relevance of information delivered to designers.

In addition to supporting individual designers to do design, DODEs support long-term indirect collaboration between designers @Cite<Fischeretal.1992a>. Long-term collaboration is required in the design of complex and evolving artifacts, which are maintained and modified over a span of years. Such artifacts are not designed from scratch but are iteratively refined. In this sense, design artifacts are never complete but instead are constantly evolving.

As designers use a DODE for many design tasks over a long period of time, design artifacts as well as representations for intent are accumulated in repositories of the DODE. These repositories allow designers to utilize information and artifacts from prior designs.

Comment: Design documentation, reference materials, precedent solutions, and design guidelines are information sources that enable indirect, long-term collaboration with past designers. However, the existence of large amounts of design information does not guarantee effective collaboration. In information-intensive domains, the challenge is not merely to accumulate information, but rather it is to find information relevant to unanticipated problems that arise during the design task.

In our DODEs, communication of intent is centered around artifacts. By capturing the intentions and priorities of designers and associating them with artifacts, design environments are able to locate stored artifacts and information that are relevant to the designer's current intention, providing the designer with a rich context for assessing the relevance of delivered information.

An Example: the Use of KID

A kitchen designer, Jane, specifies requirements for her design task using @spec (@figref<spec-part-2>), and starts constructing a floor plan using @const (@figref<const-part-4>). When she puts a dishwasher on the right side of a double-bowl sink, critic messages appear on the screen, one of which notifies her that she should put the dishwasher on the left side of the sink (see the @i(Message) window in @figref<const-part-4>). Wondering why, she clicks on the critic message.

The corresponding argument is presented, stating that a kitchen should have a dishwasher on the left side of a sink because she specified that this kitchen is for a left-handed cook (see @figref<spec-part-2>). Jane understands this suggestion, but at the same time, she looks at the argument that having a dishwasher on the left side of a sink may affect the resale value (see the @i(Argumentation) window in @figref<spec-part-2>). Then, she remembers that the resale value of the kitchen is actually a very important concern and adds this requirement using @spec, leaving the dishwasher on the right side of the sink.

Later, another designer Bob uses the system for his design task. When Bob specifies that he would like to design a kitchen for a left-handed cook and starts his design, the same critic rule fires. He looks at the catalog window, then he notices that KID delivers Jane's kitchen as a good example for his plan although Jane's kitchen has the dishwasher on the right side of a sink (not shown). Wondering why, Bob looks at the argument, and then he understands that the reason was that for Jane's design task, the resale value was more important than left-handedness.

Postscript="/homes/kumiyo/doc/KBSystem95/spec-part-2.bin",
@caption<@spec>

The user interface is based on the questionnaire forms used by professional kitchen designers to elicit their clients' requirements. @spec provides an extensible collection of questions (issues) and alternative answers from which designers select the requirements associated with their current design intent (see the @i<Questions> window). The summary of currently selected answers appears in the @I<Current Specifications for> window, and designers can assign weights to the selected answers to represent the relative importance of the specified requirements. If no existing alternatives express their position, designers can add or modify information in the underlying argumentation base. @I<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 @figref<const-part-4>) 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.

Postscript= "/homes/kumiyo/doc/KBSystem95/const-part-4.bin",
@caption<@const>

Designers construct a kitchen floor plan in the @I<Work Area> using a direct manipulation style to select and place design units from the appliance palette. Designers may copy an example from the @I<Catalog> window, where catalog examples are presented in the order of accordance with the current specification (see @figref<spec-part-2>). The @I<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.

Discussion

Comment: this discussion has little to do with the scenario!! Should this next paragraph come before the scenario??

The KID system @cite<Nakakoji1993> is a design environment for creating kitchen floor plans that substantially extends the J@C<anus> system @cite<FischerMcCallMorch1989a>. Figures @ref<spec-part-2> and @ref<const-part-4> show screen images of the @spec and @const components of KID. The specification component supports designers in framing their design problem; i.e., specifying design goals, objectives, and criteria or constraints. The construction component supports designers in constructing the solution form of the design artifact. In the kitchen domain the solution form is a floor plan.

Support for the Evolution of a Design Artifact. In KID, the designer's intention is articulated by manipulating interface objects in the construction and specification components. The system uses the state of the interface objects as the current context. The specification provides information about the designer's high-level intentions. From the construction, the system obtains information about the design moves that have been made, which represent the designer's solution-level intentions. The representations in the specification and construction that define the design context are *shared* between the designer and the system because the state of the representations is accessible to both.

KID uses computational critic mechanisms @cite<Fischeretal.1993b> to alert designers to problematic design situations, such as a violation of domain design rules, and to provide information relevant to the situation. This mechanism allows designers to become aware of the current design context in which they are engaged.

Support for Long-term Collaboration. KID contains two collections of domain knowledge: an argumentation base that stores design rationale and a catalog base that stores design artifacts. The argumentation base is a semi-structured design space that expresses interdependencies

between design decisions as well as the contexts in which the interdependencies are relevant. The catalog base contains precedent design cases represented as a construction (floor plan) and a specification (design requirements), which are created by designers in the past. Thus, the domain knowledge serves as the generic domain-level context that users of KID can collaborate by communicating their intention over a long period of time.

Mechanisms. KID uses *specification-linking rules* to map from a preference articulated in the specification to a corresponding combination of constraints that should be satisfied in the construction. The specification-linking rules enable KID to detect design situations in which the construction and specification are in conflict. Such conflicts are brought to the designer's attention by two knowledge-delivery mechanisms:

- @rd locates information in the argumentation base corresponding to the conflict between the specification and construction detected through @i<specific critics> @cite<Fischeretal.1993b>. The argumentative information helps designers to understand the problem and alternative means for resolving it.

- @cd orders the catalog space so that examples relevant to the current design situation are easily accessible to the designer @cite<Nakakoji1994>. @cd computes the conformity of each catalog example to the current partial specification by (1) applying specific critics to each catalog example, (2) computing an appropriateness value for the example as the weighted sum of the critic evaluations, (3) ordering the examples according to the values, and (4) presenting the ordered catalog examples.

In summary, KID's explicit representations of a problem specification and solution construction help designers to achieve and maintain a common understanding of the problem and prevent them from overlooking important considerations. KID uses these representations as the current context, and use the representations to identify task-relevant information. KID's knowledge bases contain design information and artifacts accumulated through past design efforts, enabling designers to collaborate indirectly with their peers from the past.

The approach described here in terms of KID has demonstrated how the representations of specification and construction of a DODE serve as a context to facilitate communication of intent between designers and a system, and among designers, for a relatively mature, stable domain such as kitchen design. For *immature* or *unstable* domains, which are relatively new, still under exploration, or heavily dependent on state-of-the-art technologies, it is difficult to identify and design such representations. In the next section, we discuss other approaches to represent design intent in DODEs to serve as context.

Issues in Representations for Context and Intent

The representation of design context and intent is not a well-defined problem, and constructing such representations (i.e., a DODE itself) is yet another design task. We have constructed DODEs for varieties of design domains including user-interface design, kitchen design, LAN design, voice-dialogue design, and software design. With each prototype, we have studied a variety of representational formalities.

Footnote: By *formality*, we mean the degree to which the system can interpret the semantics (content) of the representation.

On the one hand, the more formally designers represent their intent, the better understanding of the intent the DODE will have and, consequently, the more context the system and the designers share. On the other hand, imposing the machine's formality on designers forces them to represent design actions in an unfamiliar language and thus undermines their expressive ability.

We have identified the following requirements for representations of context and intent that both DODEs and designers can use:

- *expressive*: Designers must be able to represent intended concepts directly and distinctly using familiar notations and languages;

- *associative*: Designers and/or DODEs must be able to associate representations with those for related concepts. "What something *means* lies in how it connects to other things we know" @cite<Minsky1991>. By providing links, relations, and connections among multiple representations, designers gain an understanding of the content of the design and the partial design task.

Construction kits and critiquing rules are considered to be formal representations because the association of the representation is automatically done by the system. Argumentation is a semi-formal representation where informal textual and graphical representations are linked by designers.

We have developed the Seeding, Evolutionary Growth, Reseeding (SER) model to support the gradual development and refinement of representational formalisms @cite<Fischeretal.1994>. In the model, the evolution of a DODE (i.e., representations of a DODE) is driven by its use in designing individual artifacts, which create new requirements. Explicit representations of context serve not only evolution of individual design artifacts but also the evolution of DODEs themselves.

In what follows, we describe several forms for a DODE for representing design intent. The first four representations are associative by systems, and the last one is associative by designers supported by systems.

The Construction. The representation in a construction component is a formal design form. A DODE provides a palette of objects pre-assigned with domain semantics and a workplace where a user can manipulate those objects in order to construct a design solution. This representation can be "parsed" by the system, providing the system with information about the artifact under construction. For example, with KID (as presented in @secref<example>), the system knows that a rectangle displayed in the workplace with a label "DW" represents a dishwasher, and that the dishwasher is next to a double-bowl sink, an adjacent rectangle with two smaller rectangles inside.

Comment: in what sense can the representation be interpreted into a "single meaning"? meaning to whom - certainly not humans!!

The representation for the design solution is interpretable into a single meaning and can be represented formally within a computer system. Research in formal approaches in the AI in Design field @cite<Coyneetal.1989> studies the interpretation of such solution representation.

The Specification. A specification component provides informal representations associated with formal rules. In some design domains, a set of natural language statements exists to represent goals, objectives, and constraints of the task, shared by a community of designers. In the kitchen design, for example, a questionnaire is used to elicit a client's requirements. In the LAN design, there are a set of typical questions asked by expert network designers before installing a network.

Such statements are associated with design decisions, and the same associations are used repeatedly in many design tasks within the domain. The interdependencies among those design decisions are captured as design rationale, or "arguments." As discussed in the previous section, KID provides an argumentation base that is based on the IBIS structure @cite<ConklinBegeman1988a>. Some of design decisions (i.e., answers in the IBIS) and interdependencies (i.e., arguments in the IBIS) are associated with predefined predicates over the construction, as described above. @spec allows designers to select some answers in the argumentation base, and KID infers interdependencies using the partial specification. These interdependencies are used as @i(specification-linking rules) to identify relevant critiquing rules (i.e., "specific critics"; see @cite<Fischeretal.1993b>) and relevant reusable design examples @cite<Nakakoji1994>.

Sketches. Sketches are graphics drawn by designers that can be parsed by a DODE. Designers use drafting paper and pencil to gain their own understanding about the design problem. Most of existing design drawing systems do not allow designers to deal with abstract, vague, or uncertain properties of a partial design. Designers do not feel comfortable in using such tools, especially at the initial stages of design. And yet, such rough drafting conveys very important information regarding design intent. For example, we observed that a kitchen designer drew several bubbles with labels such as *"working center, "cooking center," and "storage area"* on a piece of paper at the very beginning of her design process. The sketch gave her a rough idea of how the workflow might be in the design without worrying about detailed precise dimensions of each appliance.

It would have been possible to ask the designer to formally represent her design intention, such as the cooking center should be adjacent to the storage area. However, in general, designers will not expend the effort until they see the benefit of entering it into the computer @cite<Fischeretal.1991c>. Supporting early design in the way that designers are accustomed to doing it (i.e., such as drafting) enables DODEs to bear at a time when they can have the least cost and the most impact @cite<Gross1994>.

Gross et al. have attempted to use hand-drawn diagrams to index architectural design cases as well as retrieve useful design cases. The underlying Electronic Cocktail-Napkin system recognizes the elements of a diagram and interprets them in the context of the architectural design domain @cite<GrossZimringDo1994>. This syntactic analysis of a graphic has enabled the DODE for symbolic and numerical analysis such as critiquing, simulation, visualization, and retrieval of relevant cases. The @mude system, based on @mei @cite(Aoki1994) also provides a mechanism to retrieve images by analyzing pictorial data from a library of 1,000 images. The system uses free-hand writing by a user as a query to match the graphic by tracing the border of images (see @figref<mude-retrieve>). The system also allows users to color a picture and retrieve images that have similar hues.

Postscript="/homes/kumiyo/doc/DIS95/mude-retrieve.eps",
@caption<Retrieval of Images using Free-hand Drawing in @MUDE>

History. The @indy network design system @cite<ReevesShipman1992> supports representations of the history of design artifact. History provides the background context of design decisions in terms of temporal relationships. The order of design decisions made implies prioritization of dependencies and thus plays a crucial role in understanding the design. The @indy system keeps track of all the design decisions made by whom over a network design, and allows designers to play backward and forward to simulate how the design has evolved.

Descriptive Annotations. The @indy system discussed above allows designers to annotate a network design. Annotations are written in natural language and provide design rationale for associated design decisions.

The @xnetwork system @cite<ShipmanMcCall1994> for LAN design supports designers to associate electronic mail to a network design. In the LAN network design, change requests and bug reports are communicated through e-mail, which drives long-term evolution of the design. The system supports designers in incrementally formalizing the accumulated information. The system suggest to designers to which design objects an e-mail is likely to be related by parsing the e-mail and inferring attribute values for predefined attributes of design objects, such as machine types, names, users, capacity, and performance. Using this information, network designers make a final decision on how to structure the information space.

The @eva system @cite<ostwald1993> is a hypermedia substrate that allows system analysts, software designers, and end-users to collaboratively evolve software prototypes. The system integrates executable prototypes and descriptive representations, such as text, graphics, and e-mail. Users of @eva can assign semantics to an association among representations. Multiple designers gradually evolve the design space by visiting representations constructed by other designers, adding their design intent and understanding of their own, and associating them with

the existing information. Using @eva, mediating collaboration, design intent, solution, and context can coevolve by multiple designers.

Conclusion

This paper presented our domain-oriented design environment approach and demonstrated how the evolution of design artifacts can be supported with explicit representations of design intent. We have identified that such representations need to be expressive and associative in order to be useful both to designers and to the knowledge-based design support mechanisms in design environments. Because designers explicitly articulate their design intent and build design artifacts in DODEs, these representations can be reused as context for reflecting upon a partially constructed design as well as understanding previous design cases. A challenge is to identify appropriate representations for a given design domain. The formalisms described in the previous section are by no means an exhaustive enumeration of possible representations for design intent. Rather they should be viewed as a starting point for future research into representations of context and intent in design.

Acknowledgments. We thank the people at the L3D (LifeLong Learning and Design) Center at the University of Colorado, who contributed to the conceptual framework and the systems discussed in this paper. The research was supported by the National Science Foundation under grants No. IRI- 9015441 and MDR-9253425; Software Research Associates, Inc. (Tokyo); NYNEX Science and Technology Center; the Colorado Advanced Software Institute; US WEST Advanced Technologies; and the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement No. CDA-940860.