Contribution to a Special Issue of Human-Computer Interaction Journal on "Context-Aware Computing"

Articulating the Task at Hand and Making Information Relevant to It

Gerhard Fischer
Center for LifeLong Learning and Design (L3D)
Department of Computer Science and Institute of Cognitive Science
University of Colorado at Boulder

Abstract

Building truly "context-aware" environments presents a greater challenge than using data transmitted by ubiquitous computing devices: it requires shared understanding between humans and their computational environments. This essay articulates some specific problems that can be addressed by representing context. It explores the unique possibilities of design environments that model and represent domains, tasks, design guidelines, solutions and their rationale, and the larger context of such environments embedded in the physical world. Context in design is not a fixed entity sensed by devices, but it is emerging and it is unbounded. Context-aware environments must address these challenges to be more supportive to all stakeholders who design and evolve complex design artifacts.

Contents

List of Figures

Introduction

The anchor article of this special issue, "Dey, Salver, and Abowd (2001 [this special issue])" [Dey et al., 2001] focuses on a narrow notion of context, namely, *location-based information captured automatically by hardware and software sensors.* As that article demonstrates, this intentional simplification leads to interesting research problems (including the development of a Context Tool kit). While the current focus of the ubiquitous computing community is mainly on understanding and handling context that can be sensed automatically in the physical environment, there are numerous other dimensions for context which can and should be taken into account.

Our work focuses on *collaborative design and problem solving*, and we have created numerous domain-oriented design environments (DODEs) that support design [Fischer, 1994]. Although context awareness provided by location-based services can be successfully used in design environments, *additional* context mechanisms (which *complement not replace* location-based services of Dey et. al.) the are needed to effectively support designers. Design deals with ill-defined problems, making the integration of problem framing and problem solving a necessity [Schön, 1983]. A particular challenge for context-aware applications in design domains is that context *emerges* throughout the design process, and that a determination of whether some information or action should be considered as relevant or irrelevant context can be determined only during the design process, not beforehand. Design environments support awareness of the following contexts: (1) the domain in which the design activity takes place; (2) the artifact under construction; (3) the user's intentions and goals as articulated (in addition to the evolving artifact) with a specification component; (4) the background knowledge of individual users as assessed by a user modeling component; and (5) the social interactions taking place in the design communities.

This essay first describes some problems occurring in design problem solving for which context-aware environments can contribute to solutions. It then discusses a conceptual framework for context-aware environments and shows how the systems that we have developed instantiate the framework. Design brings together stakeholders from different disciplines. In the context of our work, we need to differentiate between two types of stakeholders: (1) *environment developers* (e.g., software designers who collaborate with domain designers in the creation of DODEs at design time), who, in the article by Dey et.al., would use the Context Tool kit to develop context-aware applications; and (2) *domain designers* (e.g., professionals who use DODEs to do their work), who are knowledgeable in their domain and collaborate at use time with clients to design and evolve artifacts in a specific domain. The information needs of these stakeholders are different and more elaborate then those of the users in the Dey et.al article, which can be served by location-based services.

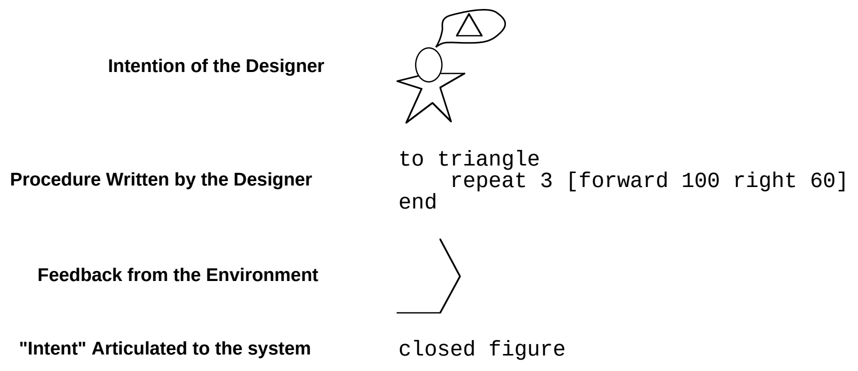
Problems for Context-Aware Environments

Creating context-aware applications is not an end in itself, but it is a *means to an end*. Our research attempts to exploit contextual awareness to support design processes. We address the question, *"How can contextual information empower users to work, learn,* *and collaborate more easily and more productively?"* The following problems describe some of the requirements for context-aware applications to support design.

Increasing the Resources for Interpretation. Interactions (e.g., posing a query) with computational artifacts are often part of a larger activity, such as a complex design task, but computer systems do not "understand" the larger activity. Ubiquitous computing [Weiser, 1993], embedded communication [Reeves, 1993], and usage data [Hill et al., 1992] make an attempt to reduce the unnecessary separation of computational artifacts from the physical objects they represent and from the discussions surrounding them. The belief that the ***"****interaction between people and computers requires essentially the same interpretive work that characterizes interaction between people"* [Suchman, 1987] raises the following interesting challenges: (1) How can we capture the *larger (often unarticulated) context* of what users are doing (especially beyond the direct interaction with the computer system)? (2) How can we increase the *richness of resources* available for computer programs to understand their uses (or what they are told about their users) and to infer from what they are observing their users doing (inside the computational environment and outside) [Horvitz et al., 1999]?

Information Overload. The challenge of future computer systems (derived from the belief that the scarce resource for most people is human attention) is not to provide information *"anytime and anywhere,"* but to *"say the 'right' thing at the 'right' time in the 'right' way,"* which can be done only with context-aware environments. Without some awareness of the tasks users are performing, and without some "understanding" of the knowledge background of the users with respect to these tasks, computational environments (and human collaborators) can make only limited determinations of the relevance of information. An example of a *context-unaware* technology is Microsoft's Tip-of-the-Day, which presents a randomly chosen tip to the users, but makes no attempt to make the information relevant to a problem the user is actually experiencing [Fischer, 2001].

Unarticulated Design Intent. In design, a large fraction of context-relevant information cannot be inferred from the environment because the context resides outside the environment, is unarticulated, or exists only in the head of a designer. Figure 1 illustrates this problem with a simple example. A designer wants to write a simple LOGO procedure to draw an equilateral triangle [Papert, 1980]. The written procedure, however, draws a different figure when the instructions are executed. The resulting figure is not "wrong," per se; it is only wrong with respect to the unarticulated intent that exists only in the designer's mind. Without access to this intent, a system is unable to detect that a problem exists. If the system provide mechanisms to articulate intentions explicitly (e.g., the intent to draw a *"closed figure"* in Figure 1), and the designer was willing to do so, the additional context could be used to identify the breakdown situation and provide the designer with opportunities for reflection and learning.


Intention of the Designer
Procedure Written by the Designer
Feedback from the Environment
"Intent" Articulated to the system
closed figure

```
to triangle
    repeat 3 [forward 100 right 60]
end
```

Figure 1: Creating context by articulating intentions

A Framework for Context-Aware Environments

To support design problem solving with context-aware application requires a broader understanding of context. This section briefly describes such a framework, emphasizing specifically context mechanisms related to the emerging task at hand that the stakeholders define, evolve and attempt to solve.

Dimensions of Context

Efforts to capture and exploit contextual information have targeted several different notions of "context," as described here.

Active badges [Harper et al., 1992] represent an early technological development for the location-based services discussed in the Dey et al. article. The information provided by active badges can be used to support *circumstantial indexing* [Bolt, 1984] as an information retrieval strategy. Circumstantial indexing does not solely rely on the object being retrieved, but it exploits the circumstances surrounding the interaction with the object (e.g., physical characteristics of the object, the date and time, location in space, people present at the time, etc.).

Latent semantic analysis (LSA) [Landauer & Dumais, 1997] enables text searches to retrieve items most similar to a sample, based on a statistical analysis of word usage and patterns in the sample compared to that of the available items. Latent semantics are meanings that are embedded in the patterns of text rather than in specific words. For this reason, proponents can claim that LSA can retrieve items based on their context. Using different samples, this type of approach can be used *to personalize and tailor information* to the knowledge background and skill level of individual user [Fischer, 2001].

Social interaction approaches, such as *usage data* [Adachi, 1998; Hill et al., 1992], *recommender systems* [Terveen et al., 1997], *social navigation* [Dieberger et al., 2000], and *attention-sensitive alerting* [Horvitz et al., 1999], all capture aspects of the user's interaction with computational artifacts. Like location-based information, these approaches capture information in the background, requiring little extra effort from users. Social interaction approaches emphasize the notion of artifacts and indications of use, and, when made explicit by context-aware applications, can provide contextual clues for future users who interact with the artifact.

Critiquing mechanisms actively and selectively present information when designers need it. This allows the designer to spend more time attending to design rather than hunting for information. *Critics* analyze the design and construction of an artifact [Fischer et al., 1998]. They use domain-specific knowledge to evaluate design artifacts, and they can intervene to present critiques when they detect a problematic situation. In this case, the critics become part of the design context by saying: *"Attention! Based on my understanding of your context, the following problematic situation may exist. If you would like to see more information about this context, it is available."* If the user chooses to see more information, this is an affirmation that the critic's inference may be correct, and this establishes a shared context for interpreting the information the system presents.

Articulating the Task at Hand with Contextual Information

Design Context and Use Context. One of the fundamental problems of system design is how to write software for millions of users (at design time), while making it work as if it were designed for each individual user (who is known only at use time).

Two stages need to be differentiated in the design and use of an artifact: in the original design context, environment developers create systems and they have to design users, situational contexts, and tasks that they can only partially anticipate [Kyng & Mathiassen, 1997]. The use-time context is determined by the actions that users perform. This aspect of context is deeply intertwined with the particular problem being solved, the individual user, and other aspects that cannot be predicted at the time the system is designed.

An important point about context-aware applications is that design context and use context become blurred. If the system is constantly adapting or being adapted to users, use context becomes a different kind of design context [Henderson & Kyng, 1991]. The challenge is to build systems that infer a use-time context based on the actions of users, and extend this inferred context by allowing the users to articulate other contextual factors that will add to the information available to the system. This can be supported with *specification components* [Nakakoji, 1993], *end-user modification and end-user programming* [Girgensohn, 1992; Repenning et al., 1999], and *incremental formalization* [Shipman, 1993].

Domain Orientation. A domain-oriented strategy attempts to support users in their *own domain of knowledge* by making assumptions about classes of users and sets of tasks in which they want to engage, and by building specialized support for the target domain [Fischer, 1994]. Domain-oriented systems build a default context into the system at design time. They enable the system to interact with the user in terms of domain objects, rules, and relationships. An important aspect of domain orientation is that domains are not assumed to be static, and users must have the means to override, adjust, and evolve some of the assumptions made at design time [Fischer et al., 1995a].

The Task at Hand. Aspects of context that cannot be anticipated at design time are those that the user determines in the use context. These include, what steps the user has performed, what decisions have been made, and what issues have been considered. Domain-oriented systems can analyze the user's actions and the resulting artifact to infer a design context and to determine information's relevance to that context. Information is relevant to the task at hand if it (1) helps someone to understand a specific problem, and (2) is made available when the need for it arises.

Externalizations in Collaborative Design. An important prerequisite for shared understanding among communities is the incremental creation of externalizations [Bruner, 1996] to capture and articulate the task at hand. Externalizations enhance mutual understanding and intelligibility by serving as a resource for assessing the meaning and relevance of information within the context of collaboration. The explicit grounding provided by externalization is a necessary substitute for the implicit and shared background that is present in everyday communication between people who collaborate often, but it is present in only a limited form among design stakeholders coming from different disciplines and is almost entirely absent in most collaborative human-computer systems.

Externalizations create context by creating "situations that talk back" to us [Schön, 1983]. In some situations, such as the drawn object in Figure 1, the "back-talk" will be easily recognized. In other situations, however, the back-talk may need to be amplified. For example, in large and complex designs our ability to recognize problematic situations by visual inspection or through detailed analysis is limited. In our research over the last decade we have developed a variety of mechanisms that increase the "back-talk" based on a shared context : (1) feedback from human stakeholders involved in the design process, (2) computational critics, and (3) simulation components that illustrate the behavior of an artifact. These mechanisms aim to provide back-talk that is relevant to the actual design situation and articulated in a way that the designer can understand.

Examples from our Work in Context-Aware Computational Environments

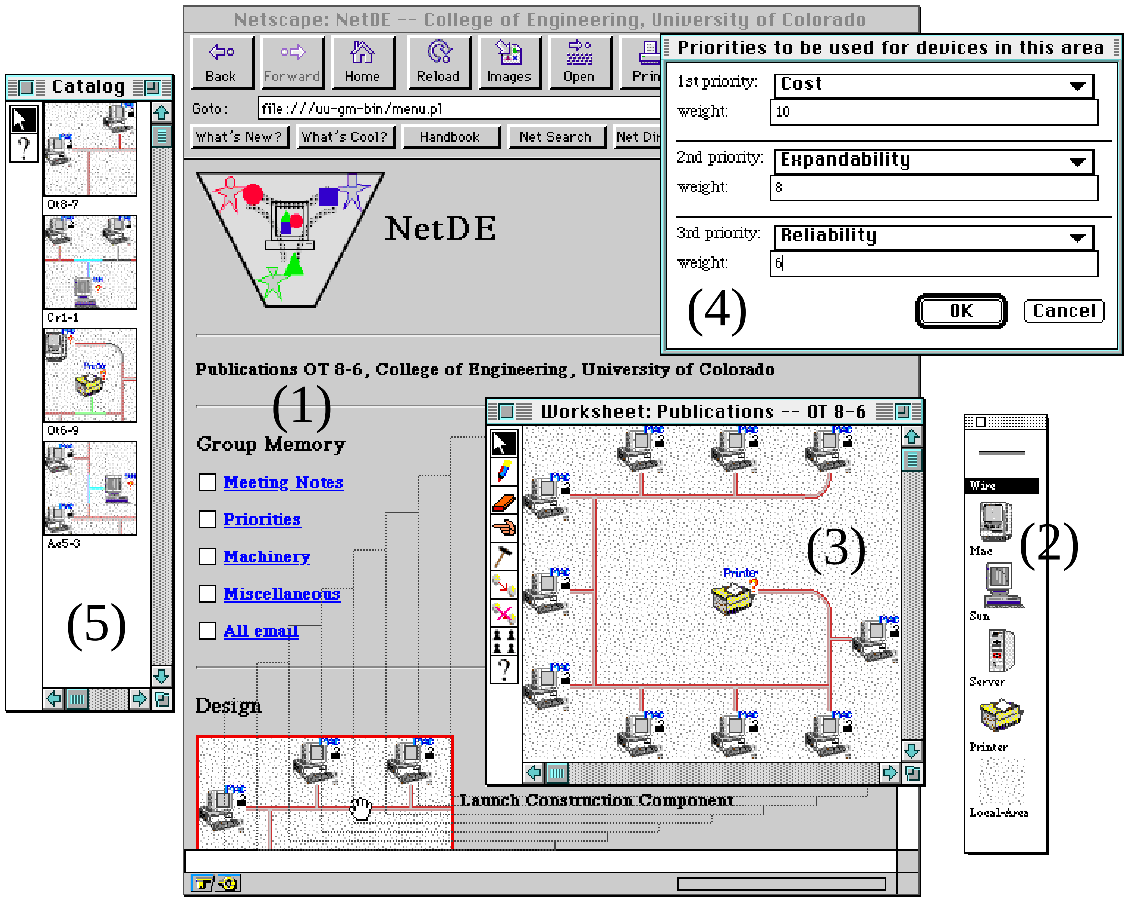
Our research activities are focused on a human-centered approach toward collaborative human-computer systems. The ends for our systems are defined by empowering human beings and the technologies developed to serve these ends. Two of our major research objectives over the last decade, domain-oriented design environments and the Envisionment and Discovery Collaboratory, are context-aware applications that illustrate some of the objectives articulated earlier in this essay.

Domain-Oriented Design Environments

Domain-oriented design environments [Fischer, 1994] support context-awareness with the following components and mechanisms (the example chosen is a DODE for designing local area networks; see Figure 2):

1. The *domain orientation* allows a default context to be assumed, namely, the creation of an artifact in the given domain.
2. The *construction situation* can be "parsed" by the system, providing the system with information about the artifact under construction. The construction situation is shaped by selecting domain objects from the gallery (pane 2) and placing them in the worksheet (pane 3).
3. The *specification component* (pane 4) allows the user to explicitly communicate high-level design intentions to the system.
4. The *simulation component* (pane 3) enriches the notion of context to include dynamic behavior.
5. The *critiquing component* (not shown) analyzes the task at hand; infers higher-level goals from low-level operations [Nardi et al., 1998]; identifies breakdowns and

information needs; and presents contextualized knowledge for designers. This knowledge is stored in a *catalog* of previous design solutions (pane 5), and an *argumentation* component that stores design rationale and alternative decisions (pane 1).


Netscape: NetDE -- College of Engineering, University of Colorado
Back
Forward
Home
Reload
Images
Open
Prin
Goto: file:///uu-gm-bin/menu.pl
What's New?
What's Cool?
Handbook
Net Search
Net Di
NetDE
Catalog
Ot8-7
Cr1-1
Ot6-9
Ae5-3
(5)
Priorities to be used for devices in this area
(4)
OK
Cancel
Publications OT 8-6, College of Engineering, University of Colorado
(1)
Group Memory
Meeting Notes
Priorities
Machinery
Miscellaneous
All email
Design
Worksheet: Publications -- OT 8-6
(3)
Printer
Launch Construction Component
Wire
Mac
Sun
Server
Printer
Local-Area
(2)

1st priority: Cost
weight: 10
2nd priority: Expandability
weight: 8
3rd priority: Reliability
weight: 6

Figure 2: A DODE for computer network design

DODEs provide an explicit context for design at two levels: (1) the generic domain level (computer network design in the example used here), and (2) specific design artifacts developed using the generic substrate (the artifacts created in the worksheet in pane 3 and stored in the catalog in pane 5 in Figure 2). Because a DODE is tuned to support design of artifacts for a particular domain, a generic shared context 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 specific design artifacts, they are incrementally representing their intentions and creating context for *a specific* artifact. As the design progresses, the designers increasingly make their intent more explicit, and the shared context for the artifact becomes more specific [Fischer et al., 1995b].
Critics embedded in design environments [Fischer et al., 1998] exploit the shared context, defined by the construction and specification, to compute what information is relevant to the task at hand. This shared context enables task-specific intervention by critics, reduces annoying interruptions, and increases the relevance of information delivered to designers.

By exploiting additional context and information, critics let designers access new knowledge in the context of actual problem situations. They inform users when they (1) are getting into trouble, (2) are missing important information, and (3) come up with problematic solutions. Critiquing systems are context-aware applications that provide the intelligent assistance for (1) contextualizing learning by integrating it into work rather than relegating it to a separate design phase; (2) letting designers see for themselves the usefulness of new knowledge for actual problem situations; and (3) making new information relevant to the task at hand.

The Envisionment and Discovery Collaboratory

The first generation of our DODEs [Fischer, 1994] was created based on an important *simplification*: our focus on design simplified the process of "context-awareness" because all activities happened *inside* the computational environment rather than in the external world. The *Envisionment and Discovery Collaboratory (EDC)* [Arias et al., 2000] represents second-generation DODEs that support social interaction by creating *shared understanding* among various stakeholders, *contextualizing* information to the task at hand, and creating *boundary objects* as externalizations in collaborative design activities [Arias & Fischer, 2000]. The EDC extends the original DODE approach by integrating computational environments and (computationally enriched) external physical worlds with mechanisms in an attempt to capture the larger (often unarticulated) context of what users are doing.

The EDC framework is applicable to different domains, but our initial effort has focused on the domain of urban planning, specifically transportation planning and community development. Figure 3 shows the current realization of the EDC environment.

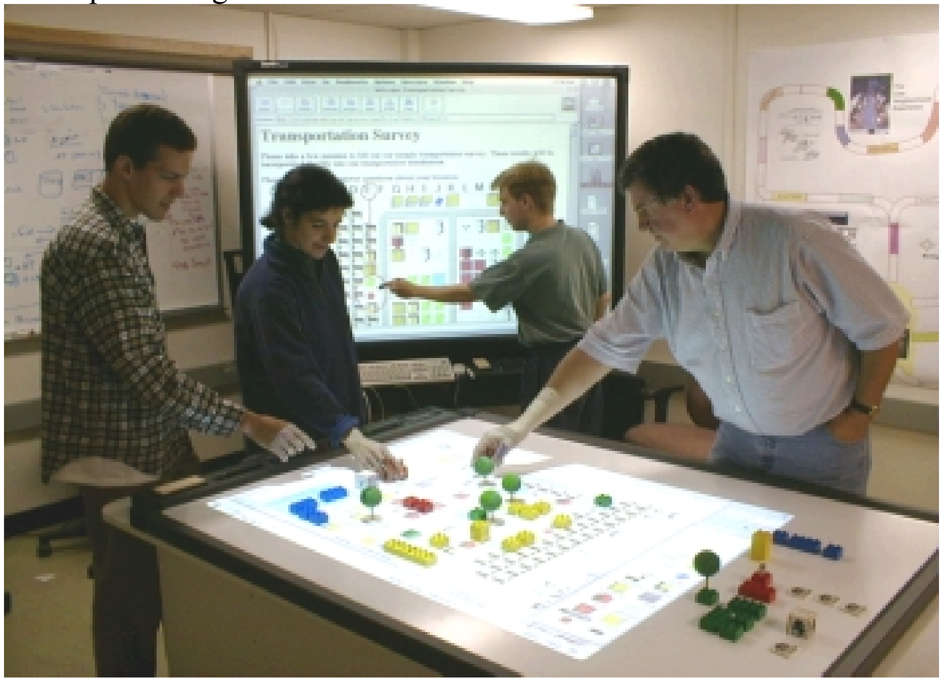


Figure 3: The EDC environment

The EDC supports stakeholders in creating information by articulating their own knowledge in a form that other people can understand. The use of a *shared physical*

context is one of the important ways to help people to articulate their knowledge and communicate with others and with the computational environment. In the EDC, the physical representation serves as an externalization through which users can express their views and create contexts that can be shared among all stakeholders and between the stakeholders and the computational environment.
The EDC uses context to support not only access to information, but *informed participation*. As argued before, context emerges in design problem solving. The evolvable nature of the EDC supports a synergy between the access of existing information and the creation of new knowledge. The information grows over time by allowing users to extend the environment with their objectives and ideas. The vision behind the EDC is to shift the focus of future developments away from the computer and toward an increased understanding of the human, social, and cultural environment that defines the context in which systems are used.

Brief Assessment

Beyond Location-Based Services. To assess the different levels of context awareness, we can ask which services in the EDC might be provided by the location-based services discussed in Dey et al. These services could provide answers to "who, what, when, where" questions associated with different design sessions, and this information could be successfully exploited by other software systems. But location-based services are only a part of the context-awareness that we support with our design environments and the EDC. To achieve additional levels of context-awareness requires (as argued in this essay) modeling the domain in which the design activities take place; facilitating the analysis of the artifact under construction; supporting users in articulating their intentions; assessing the background knowledge of individual users; and capturing some aspects of the social interactions of the stakeholders involved in the design activity.
Utility equal Value over Effort. Environments and tools have a high utility factor when they either (1) represent a high value and contribution toward achieving our goals or (2) require a small effort. The location-based services described in the Dey et al. article are successful because they require a small effort at use time; they exploit information provided *automatically* by sensors and by software mechanisms that track users' interactions with other designers and with artifacts. This is also true for techniques such as Latent Semantic Analysis [Landauer & Dumais, 1997], which assist us in extracting meaning from text without additional efforts. In contrast, some of the techniques that we have pursued in our work try to create a larger value for the designers, thereby making an additional effort more feasible. Providing additional contextual information (e.g., asking designers to articulate their intentions with a specification component or to provide design rationale) requires an additional effort, which stakeholders are willing to contribute if the value gained by doing so still achieves a high utility factor for them.
Formality. Formality is the degree to which a system can interpret the semantics of an artifact. The more formally designers represent their intent, the more context the system and the designers can share. Design environments and the EDC contain a number of different representations with varying degrees of formality. However, imposing the computer's formality on designers forces them to represent design actions in an unfamiliar language, undermines their expressive ability, and forces them to spend

considerable efforts on tasks they may consider only secondary. Incremental formalization [Shipman, 1993] is a technique to combine the best of these two approaches by allowing designers to formalize information at a time chosen by them.

Future Challenges and Conclusion

Context-aware applications present an important research topic for human-computer interaction in design problem solving because they address issues discussed in this essay. DODEs help designers to create and evolve designs and to understand the effects of individual design moves. One hypothesis for future work derived from this essay is that capturing context (either directly from the physical world or inferring it from user interactions) is meaningless without an understanding of the user's goals and objectives. To capture this contextual information will require domain, task, and user modeling. One fundamental challenge is that in design problems, the context does not preexist, but is created and emerges in the process of solving a problem. This requires design approaches in which problem framing and problem solving are tightly integrated. Design problems cannot be defined “up front,” before any attempt at a solution is made. Design environments are a promising approach to creating context-aware applications that will empower users in their activities.

Acknowledgments

The author would like to thank the members of the Center for LifeLong Learning & Design at the University of Colorado, who have made major contributions to the conceptual frameworks and systems described in this essay. Ernesto Arias, Hal Eden, Andy Gorman, and Eric Scharff have been the major contributors to the development of the Envisionment and Discovery Collaboratory. Jonathan Ostwald, Anders Morch, Hal Eden, Eric Scharff, Rogerio dePaula, and the reviewers of this essay deserve a special thanks for their criticism of an earlier version. The research was supported by (1) the National Science Foundation, Grants REC-9631396 and IRI-9711951; (2) Office of Naval Research, Cooperative Agreement No., N66001-00-1-8964; (3) Software Research Associates, Tokyo, Japan; (4) PFU, Tokyo, Japan; and (5) the Coleman Foundation, San Jose, CA.

References

Adachi, T. (1998) *Utilization of Usage Data to Improve Organizational Memory*, M.S. Thesis, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Arias, E. & Fischer, G. (2000) "Boundary Objects: Their Role in Articulating the Task at Hand and Making Information Relevant to It," *International ICSC Symposium on Interactive & Collaborative Computing (ICC’2000), University of Wollongong, Australia, ICSC Academic Press, Wetaskiwin, Canada*, pp. 567-574.

Arias, E. G., Eden, H., Fischer, G., Gorman, A., & Scharff, E. (2000) "Transcending the Individual Human Mind—Creating Shared Understanding through Collaborative Design," *ACM Transactions on Computer Human-Interaction,* 7(1), pp. 84-113.

Bolt, R. A. (1984) *The Human Interface,* Lifetime Learning Publications, Belmont, CA.

Bruner, J. (1996) *The Culture of Education,* Harvard University Press, Cambridge, MA.

Dey, A. K., Salber, D., & Abowd, G. D. (2001) "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Human-Computer Interaction,* 16, xxx - xxx [this special issue]

Dieberger, A., Dourish, P., Höök, K., Resnick, P., & Wexelblat, A. (2000) "Social Navigation: Techniques for Building More Usable Systems," *Interactions,* 7(6), pp. 36-45.

Fischer, G. (1994) "Domain-Oriented Design Environments," *Automated Software Engineering,* 1(2), pp. 177-203.

Fischer, G. (2001) "User Modeling in Human-Computer Interaction," *User Modeling and User-Adapted Interaction, Dordrecht, The Netherlands: Kluwer Academic Publishers,* (to appear).

Fischer, G., Lindstaedt, S., Ostwald, J., Stolze, M., Sumner, T., & Zimmermann, B. (1995a) "From Domain Modeling to Collaborative Domain Construction." In G. M. Olson & S. Schuon (Eds.), *Proceedings of DIS'95 Symposium on DesigningInteractive Systems: Processes, Practices, Methods, & Techniques,* ACM, New York, pp. 75 - 85.

Fischer, G., Nakakoji, K., & Ostwald, J. (1995b) "Supporting the Evolution of Design Artifacts with Representations of Context and Intent." In G. M. Olson & S. Schuon (Eds.), *Proceedings of DIS'95 Symposium on Designing Interactive Systems: Processes, Practices, Methods, & Techniques,* ACM, New York, pp. 7-15.

Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., & Sumner, T. (1998) "Embedding Critics in Design Environments." In M. T. Maybury & W. Wahlster (Eds.), *Readings in Intelligent User Interfaces,* Morgan Kaufmann, San Francisco, pp. 537-561.

Girgensohn, A. (1992) *End-User Modifiability in Knowledge-Based Design Environments*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Harper, R., Lamming, M. G., & Newman, W. M. (1992) "Locating Systems at Work: Implications for the Development of Active Badge Applications," *Interacting with Computers,* 4(3), pp. 343-363.

Henderson, A. & Kyng, M. (1991) "There's No Place Like Home: Continuing Design in Use." In J. Greenbaum & M. Kyng (Eds.), *Design at Work: Cooperative Design of Computer Systems,* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp. 219-240.

Hill, W. C., Hollan, J. D., Wroblewski, D., & McCandless, T. (1992) "Edit Wear and Read Wear." In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), *Proceedings of CHI'92 Conference on Human Factors in Computing Systems,* ACM, New York, pp. 3-9.

Horvitz, E., Jacobs, A., & Hovel, D. (1999) "Attention-Sensitive Alerting." In *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence (Stockholm, Sweden),* Morgan Kaufmann, San Francisco, pp. 305-313.

Kyng, M. & Mathiassen, L. (Eds.) (1997) *Computers and Design in Context,* MIT Press, Cambridge, MA.

Landauer, T. K. & Dumais, S. T. (1997) "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge," *Psychological Review,* 104(2), pp. 211-240.

Nakakoji, K. (1993) *Increasing Shared Understanding of a Design Task Between Designers and Design Environments: The Role of a Specification Component*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Nardi, B. A., Miller, J. R., & Wright, D. J. (1998) "Collaborative, programmable intelligent agents," *Communications of the ACM,* 41(3), pp. 96-104.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas,* Basic Books, New York.

Reeves, B. N. (1993) *The Role of Embedded Communication and Artifact History in Collaborative Design*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Repenning, A., Ioannidou, A., & Phillips, J. (1999) "Collaborative Use & Design of Interactive Systems," *Proceedings of the Computer Supported Collaborative Learning (CSCL '99) Conference*, pp. 475-487.

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

Shipman, F. (1993) *Supporting Knowledge-Base Evolution with Incremental Formalization*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

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

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997) "PHOAKS: A System for Sharing Recommendation," *Communications of the ACM,* 40(3), pp. 59-62.

Weiser, M. (1993) "Some Computer Science Issues in Ubiquitous Computing," *Communications of the ACM,* 37(7), pp. 75-83.