Selfridge, P. G. 1991. Knowledge representation support for a software information system. In *Proceedings of the 7th IEEE Conference on AI Applications*, pp. 134–140, Miami Beach, Florida.
Selfridge, P. G., Terveen, L. G., and Long, M. D. 1992. Managing design knowledge to provide assistance to large-scale software development. In *Proceedings of the Seventh Knowledge-Based Software Engineering Conference (KBSE-92)*, pp. 163–170, Tyson's Corner, Virginia, September 22–25. Available from the IEEE Press.
Terveen, L. G., Selfridge, P. G., and Long, M. D. 1993. From "Folklore" to "Living Design Memory". In *Proceedings of INTERCHI-93*, pp. 15–22, Amsterdam, The Netherlands, April 24–29. Available from ACM.
Terveen, L. G., Selfridge, P. G., and Long, M. D. (in press) *Living Design Memory for Software Development: Framework, System, Lessons Learned.*
Waters, R. C., and Chikofsky, E. J. (eds.) 1993. *Working Conference on Reverse Engineering, Baltimore, Maryland, May 21–23.* Available from the IEEE Computer Society.
Zave, P. 1993. Feature interactions and formal specifications in telecommunications. *IEEE Computer*, 26:20–31.

Automated Software Engineering, 1, 223–229 (1994)
© 1994 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Domain-Oriented Design Environments: Reply to Commentaries

GERHARD FISCHER
Department of Computer Science and Institute of Cognitive Science, University of Colorado, Boulder, Colorado 80309

A. Sutcliffe, J. Ning, P. Selfridge and D. Setliff have commented on my article "Domain-Oriented Design Environments" and I would like to thank them for their insightful comments. One should not be surprised that researchers hold different views about an area as complex and as volatile as Software Engineering, especially because my paper not only describes some past achievements, but also outlines a research agenda for the future. I would like to thank the editors of *Automated Software Engineering* who have granted me the privilege of replying to the comments by A. Sutcliffe, J. Ning, P. Selfridge, and D. Setliff. I have organized my reply around themes, using the names of these individuals as references. I have chosen often to use "we" instead of "I" to acknowledge the group of collaborators at CU Boulder and elsewhere, who share with me the same view.

Design

Design is concerned with "how things ought to be in order to attain goals, and to function" (Simon, 1981). Design understood this way is more than "the act of translating requirements into specifications and constraints" (Setliff). Design complements the natural sciences, whose primary goal is to analyze. Designers not only solve given problems by reasoning about formal representations, but they (architects, industrial designers, curriculum designers, or software designers) have to get actively involved in framing problems. Designers are not the sole owners of problems. They have to collaborate with all stakeholders (clients, customers, other designers) in a mutual education process to understand problems and construct the knowledge for solving them. Design methods will be deeply influenced by the artifacts developed. The design of computational artifacts to empower humans faces different issues than the design of technical systems, such as VLSI CAD design (Setliff) or compilers.

Problems of Domain-Oriented Design Environments

What is a Domain?

Sutcliffe raises the issue that "there is no sound theory about what a 'domain' is." I agree that domains cannot be precisely defined—they are part of the design activity themselves (so they change when goals change). We try to define domains in our environments (such as departments in universities, or professional societies), and they serve as useful constructs. But at the same time, we call for interdisciplinary research to acknowledge that real world problems do not fit into our preconceived domains. Domains and their boundaries will undergo change as our world changes. This is specifically acknowledged in our work by postulating our model of seeds, evolutionary growth and reseeding.

I disagree with the assertion that "it is difficult to imagine that DODEs can be built for immature domains" (Ning). Our research has demonstrated that it can be a very fruitful endeavor to create DODEs for immature domains (and we have done so for lunar habitat design, for computer networks, etc.). By creating DODEs through intensive collaboration with domain experts, we have shown that these efforts can make major contributions toward deepening our understanding of a domain.

What is the Price of Working in a Domain?

Sutcliffe asserts that "DODE's domain-specific nature will limit application to a small set of related problems, leaving only an outline architecture as a more general result." This is an adequate characterization and it is supported by the results of our work. We are aware of the tension and the design trade-off between the Turing Tar-Pit (as articulated by Alan Perlis) "The Turing Tar Pit: everything is possible but nothing of interest is easy" and the inverse of it "The over-specialized system: everything is easy, but nothing of interest is possible." Referring back to human organizations and domain expertise again: our society educates its members in domains, and switching from one domain to another is a non-trivial undertaking. So why should we expect that we will get DODEs for free? There is growing wide spread recognition and a growing number of computational artifacts that demonstrate that domain orientation will allow us to develop new generations of human-centered computational artifacts (e.g., Mathematica for mathematicians, spreadsheets for planning and decision making, drawing and painting software for artists, etc.) by supporting *human problem-domain* communication with the goal of narrowing the gap between subject domain and computational substrate.

We are working on substrates and layered architectures to increase the sharing of components between DODEs in related domains. But without paying the price of working in a domain, our computational environments will be severely limited in the amount (1) of support they can provide (e.g., there would be no work-triangle critic without domain knowledge), and (2) of end-user control and interest (e.g., end-users are not interested in the computer per se, but in their tasks).

Knowledge Acquisition

Ning observes that "an effective DODE will require a large amount and a variety of domain-oriented knowledge." Our process model, based on seeds, evolutionary growth, and reseeding (Fischer et al., 1994), is an important alternative to the conventional approaches of

knowledge acquisition as well as the futuristic approaches of machine learning pursued in AI-oriented research efforts. Our model explicitly acknowledges the fact that (1) human knowledge is tacit (Polanyi, 1966) (so the best we can hope for is a seed), (2) knowledge changes over time (requiring support for evolutionary growth), (3) the breakdowns based on lack of knowledge will be experienced by the domain designers and not by the environment developer (making end-user modifiability a necessity rather than a luxury), and (4) social incentives and rewards for providing and documenting this knowledge (e.g., in the form of design rationale) may be more important than the particular formalism chosen for its representation.

How, Not Why

Ning states "that the real question today is how to develop, rather than why we should or should not develop DODEs." Our research prototypes (see references in my paper) demonstrate that we have some understanding of "how" one goes about building DODEs. Beyond that, we assisted others in developing DODEs and demonstrated the practical value of some of our DODEs in industrial research environments (e.g., the voice dialog design environment in use at USWest Advanced Technologies (Repenning and Sumner, 1992), the service-provisioning environment in use at NYNEX (Ostwald, Burns, and Morch, 1992), and the lunar habitat environment in use by a NASA contractor (Stahl, 1993)).

An important aspect of DODEs is the possibility to construct them *incrementally* (e.g., the voice dialog design environment existed and was used by domain workers for more than a year before a critiquing component was added), and to emphasize different components for different domains (e.g., the simulation component is of great importance in the voice dialog design environment).

Scaling Up

Scaling up is a critical issue for DODEs as it is for any other computational environment. Our work so far demonstrated (1) that the "seeds—evolutionary growth—reseeding" model provides a good foundation for scaling up, and (2) that many of the integration components assist users in dealing with information spaces that are too large to be explored by browsing only. DODEs acquire a partial understanding of the task at hand by analyzing the partial construction and the partial specification. The CONSTRUCTION-ANALYZER and CATALOG-EXPLORER exploit this partial understanding to locate relevant argumentation and catalog examples for the user. Following Sutcliffe's observation that for large information spaces "intelligent retrieval engines will be necessary," we have explored such mechanisms for several years (Fischer, Henninger, and Redmiles, 1991) and incorporated them in our DODEs (Nakakoji, 1993).

The Proper Role of Automation

Understanding the Proper Role of Humans and Computers in Joint Human-Computer Systems

Even strong advocates of automated systems such as expert systems' researchers acknowledge that "most knowledge-based systems are intended to be of assistance to human endeavor; they are almost never intended to be automatic agents. A human-machine interaction subsystem is therefore a necessity" (Feigenbaum and McCorduck, 1983). The proper role of humans and computers has been explored in numerous areas (to name just a few examples: in machine translation (Kay, 1980), in cockpit design (Billings, 1991), and in the general foundations for tool and system design (Illich, 1973; Fischer, 1990)). The question of the proper role of automation is raised succinctly by Billings (1991): "During the 1970's and early 1980's . . . the concept of automating as much as possible was considered appropriate. The expected benefits were a reduction in pilot workload and increased safety. Although many of these benefits have been realized, serious questions have arisen and incidents/accidents have occurred which question the underlying assumption that the maximum available automation is always appropriate or that we understand how to design automated systems so that they are fully compatible with the capabilities and limitations of the humans in the system" (p. 4). Contrary to Sutcliffe's claim that "every domain will have to have an exhaustive analysis to find all the principles, rules, guidelines etc., for good design," critiquing components embedded in DODEs do not require any kind of completeness. While it is highly desirable that a substantial amount of critiquing knowledge gets accumulated over time, a system with a just a few critiquing rules can greatly increase the usability of a DODE.

Lack of any Theoretical Basis for Cooperation

Sutcliffe observes that "unless design of software tools is based on a sound analysis of how the user and machine cooperate to achieve designs we run the risk of providing inappropriate functionality which may either over-automate or under-support the designer's job." Understanding cooperation is a critical challenge not only for KBSAs and DODEs, but for all intellectual teamwork (Galegher, Kraut, and Egido, 1990). Our work on DODEs should not and cannot wait until the theoretical basis for cooperation will exist, but we attempt with our efforts to contribute to the creation of this basis. Our work is guided by principles for collaboration, such as (1) all stakeholders must be involved (to account for the "symmetry of ignorance" (Rittel, 1984)), (2) to be involved, the stakeholders must be informed in an understandable way (requiring that representations are developed that can serve as "languages of doing" (Ehn, 1988)), and (3) there must be shared knowledge (including knowledge of each other's intent (Resnick, Levine, and Teasley, 1991)).

Integrating KBSAs and DODEs

In Setliff's view, "current software synthesis architectures are well on their way toward stantiating Fischer's DODE design process"—indicating that many recent research ef emerging from the instantiation of the original KBSA effort moved toward some of goals of DODEs. I see a natural symbiosis between the two research directions: KB emphasize downstream activities and DODEs emphasize upstream activities. This vie shared by Selfridge when he observes about our work: "The most important distinctic the focus on the 'upstream' activities of problem *understanding*, as opposed to prob *solving*." Obviously, either approach cannot ignore the other phase (e.g., we have built eral computational substrates serving as lower layers in DODEs (Repenning and Sum 1992), and the KBSA efforts have pursued upstream activities in the context of requ ments engineering (Proceedings, 1993). But the different emphasis has led to a nun of differences: KBSAs and DODEs investigated different classes of problems, looked different disciplines for help and ideas, and approached the human role and assessn studies from different angles.

Problems are Different

Sutcliffe observes that "for safety critical domains, formal approaches and automatic gramming is not only desirable but essential. However, Fischer reminds us that m problems do not fall into this class." There is no doubt that we need correct and effic programs (just as we need buildings that do not collapse), but what is the value of th programs if they are not relevant, suitable, adequate, or enjoyable to users in their situation (just as houses are judged by more criteria than that they do not fall down also claim that the scientific community needs a better understanding of the limitation formal methods in safety critical systems (e.g., the accident in the Persian Gulf in wh an airliner relying on the AEGIS system was shot down represents a design disaster formal methods would not have prevented (Lee, 1992)).

Where Do We Look for Ideas and Help?

Historically, computer science has looked to mathematics and logic to create a foun tion (and these disciplines served well for improving "downstream" activities). But a the foundations have been established, other disciplines may be more important, such cognitive psychology (to better understand the human part), social sciences (to underst collaboration), evolution (to understand the nature of complex systems), and architect (to understand design as an activity that needs to define and create contexts and not o operate in given contexts). In the long run, I think that "Software Engineering" may the wrong term because it focuses on the medium rather than on the characterization domains (in mature design domains, we do not speak of "steel" or "concrete" engineeri but of "civil" or "electrical" engineering).

Do Not Postulate a New Human

Simon (1981) acclaims the framers of the U.S. Constitution by noting that "they did not postulate a new man to be produced by the new institutions but accepted as one of their design constraints the psychological characteristics of men and women as they knew them, their selfishness as well as their common sense" (p. 163). It may be that what is wrong with the logical and mathematical design methods is that they are the product of a mode of reasoning alien to design (Rittel, 1984). A human-centered view toward design should take into account that "logic is most definitely not a good model of human cognition. Humans take into account both the content and the context of the problem, whereas the strength of logic and formal symbolic representations is that the content and context are irrelevant. Taking content into account means interpreting the problem in concrete terms, mapping it back onto the known world of real actions and interactions" (Norman, 1993) (p. 228). This and other observations such as (1) humans enjoy doing and deciding, (2) humans act until breakdowns occur, (3) humans operate by using information in the world as an important resource, and (4) domain-orientation preserves content and context, have served as guiding principles for our work on DODEs to complement the more formal approaches pursued in the KBSA communities.

References

Billings, C. E. 1991. *Human-Centered Aircraft Automation: A Concept and Guidelines.* NASA Technical Memorandum 103885, NASA Ames Research Center, Moffett Field, CA, August.

Ehn, P. 1988. *Work-Oriented Design of Computer Artifacts.* Stockholm, Sweden: Almquist & Wiksell International.

Feigenbaum, E. A., and McCorduck, P. 1983. *The Fifth Generation. Artificial Intelligence and Japan's Computer Challenge to the World.* Reading, MA: Addison-Wesley Publishing Company.

Fischer, G. 1990. Communications requirements for cooperative problem solving systems. *The International Journal of Information Systems* (Special Issue on Knowledge Engineering), 15(1):21–36.

Fischer, G., McCall, R., Ostwald, J., Reeves, B., and Shipman, F. (in press). Seeding, evolutionary growth and reseeding: Supporting incremental development of design environments. In *Human Factors in Computing Systems, CHI'94 Conference Proceedings (Boston, MA).*

Fischer, G., Henninger, S. R., and Redmiles, D. F. 1991. Cognitive tools for locating and comprehending software objects for reuse. In *Thirteenth International Conference on Software Engineering (Austin, TX)*, IEEE Computer Society Press, ACM, IEEE, Los Alamitos, CA, pp. 318–328.

Galegher, P., Kraut, R., and Egido, C. (eds.) 1990. *Intellectual Teamwork.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Illich, I. 1973. *Tools for Conviviality.* New York: Harper and Row.

Kay, M. 1980. *The Proper Place of Men and Machines in Language Translation.* Technical Report CSL-80-11, Xerox Palo Alto Research Center, October.

Lee, L. 1992. *The Day The Phones Stopped.* New York: Donald I. Fine, Inc.

Nakakoji, K. 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 Technical Report CU-CS-651-93.

Norman, D. A. 1993. *Things That Make Us Smart.* Reading, MA: Addison-Wesley Publishing Company.

Ostwald, J., Burns, B., and Morch, A. 1992. *The Evolving Artifact Approach to System Building, Working Notes of the AAAI 1992 Workshop on Design Rationale Capture and Use*, AAAI, San Jose, CA, July, pp. 207–214.

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

Proceedings of IEEE International Symposium on Requirements Engineering. 1993. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, January.

Repenning, A., and Sumner, T. 1992. Using Agentsheets to create a voice dialog design environment. *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, ACM Press, pp. 1199–1207.

Resnick, L. B., Levine, J. M., and Teasley, S. D. (eds.) 1991. *Perspectives on Socially Shared Cogni* Washington, DC: American Psychological Association.

Rittel, H. W. J. 1984. Second-generation design methods. In *Developments in Design Methodology*, edite N. Cross, pp. 317–327. New York: John Wiley & Sons.

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

Stahl, G. 1993. *Interpretation in Design: The Problem of Tacit and Explicit Understanding in Computer Sup of Cooperative Design.* Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boul CO.