Interview With Mark Bernstein

Claus Atzenbeck
Aalborg University Esbjerg, Denmark

Mark Bernstein is chief scientist at Eastgate Systems, Inc. A graduate of Swarthmore College, he received his Ph.D. (in physical chemistry) from Harvard University. After helping to organize Eastgate in 1982, he joined the research staff at the Artificial Intelligence group at E. I. duPont de Nemours & Co., where here remained until returning to Eastgate in 1987. At Eastgate, he has overseen Eastgate's influential hypertext publishing program, which includes many of the most respected fiction and nonfiction hypertexts that have appeared to date. He has managed development of the Storyspace hypertext writing environment and designed Tinderbox, a content assistant for making, analyzing, and sharing notes. A part of nearly every ACM Hypertext Conference program committee since 1989, he has twice been its program co-chair, and was its keynote speaker in 1999. He is currently program chair of WikiSym '08, the 4th ACM Symposium on Wikis. His book, *The Tinderbox Way*, was published by Eastgate in 2006.

DOI: 10.1145/1377501.1377505 http://doi.acm.org/10.1145/1377501.1377505

What is your current primary research interest?

I'm very interested in *nobitic* hypertext – notes we write to ourselves, or to our intimate circle. Tinderbox[1] is a spatial hypertext system for making, analyzing, and sharing research notes. It draws heavily on the hypertext research literature; indeed, a large part of it was originally designed in an impromptu meeting at a Hypertext conference.

I've been thinking a lot about links – their history, the way we use them. That work probably leads to a book on *The Natural History of the Link*. I think it's about half finished.

I'm also interested in *artisanal software*: software that's made by individuals and small teams and that embodies their vision, software that is crafted in workshops, irregular and inspired.

Please tell us about your vision in this field of research.

One thing shared by all three areas is rejection of merely user-centered design, an willingness to accept that real problems and real data have structure, and that structure should inform and influence our design.

We have assumed for years that users know just what they want. They tell us what to do and we comply, on budget and on time. Sometimes this works, at other times it leads us to designs that alter with each breeze of politics and with each fashion trend. In consequence,

[1] http://www.eastgate.com/Tinderbox/

design judgment is placed into the hands of those who do not care about the subtle structure of their task. Instead, they choose what's best for their career, their next promotion, the annual report. Is "code to spec" not "slave's work unredeem'd"?

At the same time, we've learned how immensely complex texts really are, even simple workmanlike texts like personal notes, journals, diaries. Multivalence is not a vice; you can't divide meaning into separate univalent atoms. We're just beginning to understand and making explicit subtle relationships that were previously obscured by the crude expressive and rhetorical treatments that were available to us. Look at the way twenty years of hypertext have transformed our understanding of how writing works, how the structure of text and its reception actually functions.

It's not our job to replicate old structure or the status quo. Our greatest ambition need not be to streamline existing workflow or to allow people to fill out forms slightly faster or a little bit more accurately.

How has hypertext influenced your work?

It's been my work. When I left DuPont in 1987, I looked around for research areas that a small company might plausibly pursue. Hypertext, direct-manipulation interfaces, and Literate Computing then seemed the most promising options. But the second has become so ubiquitous as to be invisible, and the development of hypertext made literate computing less necessary; instead of tangling and weaving your code into a book, you can just link your books to your code.

But hypertext has turned out to be much deeper and more interesting than we expected.

Would you briefly outline how your personal view of hypertext has evolved over time?

In the earliest days, hypertext systems were hard to build. Nobody had much experience reading hypertexts, or writing them. I don't think anyone appreciated how interesting and deep the issues of reading and writing hypertext – the question of hypertext rhetoric – would turn out to be.

My earliest hypertext work was deeply concerned with the Navigation Problem, the fear that people would get lost in hypertexts. I argued that we should provide lots of tools to help people find their way: bookmarks, margin notes, landmarks, breadcrumbs. These have all turned out to be useful, but the underlying concern was specious: people are no more prone to get lost in a hypertext than they are in any text.

If you had the choice, would you dis-invent any technological advancement?

Certainly not. Knowledge is always preferable to ignorance.

Have you noticed a focus change of the hypertext community since you became active in this field?

In the 1980's, the hypertext world was filled with people who built systems that embodied their technical and theoretic vision. The demos at Hypertext '87 were literally at the center of the conference. One big room, lots of big systems, systems we'd been reading about for years but that you'd never actually seen before. There in one room: Ted Nelson's Xanadu, Engelbart's NLS/Augment, Walker's Symbolics Document Explorer, Joyce and Bolter with Storyspace, my Hypergate, Meyrowitz and Landow and Yankelovich and van Dam with Intermedia. There was Scott Johnson's Black Magic, and Ben Shneiderman's HyperTIES, and Atkinson's HyperCard, and Peter Brown's Guide. You had lots of ideas and lots of software.

We lost that somewhere along the line. So did Computer Science as a whole.

What is special about hypertext? What role does hypertext have within computer science?

Big picture: it is now clear that the future of serious writing lies on the computer display. The link has transformed journalism, permanently altered fiction and film, and has changed the process of scholarly research and composition. Even in unlinked media, we are always aware of where the links would be, or of their absence.

The growth of real hypertext has always been much slower than I expected. I'm the guy, after all, who wanted to know (at Hypertext '89), "Where are the hypertexts?" But the change is real, and it's vitally important that the change continues. Look, for example, in the botched reconstruction of Iraq in 2003; the people we sent to figure out what needed to be done and to do it were simply unable to learn enough, fast enough. Some of them were incompetent, some were scoundrels, a few were fools, and some were criminals. But many were simply trying to do the job with the information tools they had: PowerPoint and cell phones. It wasn't enough. We can (and must) do better.

So, that's the role of hypertext in society. What about within computer science?

First, it's a microcosm of the most interesting ideas of computer science, a terrific example that combines the grand concepts and controversies of the discipline with palpable and immediate utility. We've been arguing about links since the beginning: Are they unstructured, like the GOTO statement? Or are they merely atomic, like the s-expression? Is the link another layer of indirection, the proverbial cure for all computer science ills? How shall links be represented? Is the link "owned" by the source document, the destination document, or by the link creator? We can see all our old familiar controversies – are pointers good or ill? strong vs. dynamic typing? – refracted in a context with widespread application and broad impact.

Second, hypertext has a fascinating relationship to representation. At AAAI-88, Ted Nelson and Doug Lenat found themselves debating *The One True System* vs. *The One True Ontology*, a tension that remains very real. The history of hypertext is filled with systems that were chiefly about representation: gIBIS, HyperSet, SEPIA, MacWeb, Aquanet. But other systems have discovered great strength and expressive power in informality and

emergent structure, among them VIKI, VKB, and Tinderbox.

Hypertext transformed the everyday practice of software engineering. What is an IDE? It's a tool that automatically builds generic links in your source code, and that links your source code to the system documentation.

I think the most important impact for Computer Science, though, lies in the way hypertext reintegrates the technical and literary arts. To update the Bauhaus manifesto: *The ultimate goal of all creative activity is the Web site!* We set out to build a hypertext because we want to express something: that sounds like literature. But we find ourselves reaching for the IDE and the debugger and the graphic design tools as often as we reach for the dictionary. Are links writerly, or are they computational artifacts? To do this well, code and words must work together – they must be one. There is no other way to do what we need to do.

What is your take on future developments in hypertext? What one thing would you most like to see in hypertext?

The earliest great hypertexts were self-consciously experimental. Some were Modern, some Postmodern, but all viewed themselves in the tradition of the literary *avant-garde*.

The second wave of great hypertexts were so intent on abjuring the avant-garde and indeed the arts that we had trouble viewing them as texts, much less great. But the original Yahoo directory was great, and so was Google, and Suck, Salon, and the cadre of weblogs that lead from *Links from Underground* and Jennicam to today's blogosphere. They insisted on their artlessness (and, often, their seeming linklessness) but they were artful and they were hypertexts.

What still remains to be done is to discover effective styles for writing densely interlinked books and essays that are compelling, coherent, and complete[2]. I cannot imagine, for example, writing narrative history today without hypertext, but we still lack good models for how best to approach this task. We know much less than we ought to know about controlling energy in hypertexts, about writing a hypertext page-turner.

What will be the next big thing in IT technology in general?

The next big thing in IT technology is almost always a buzzword, standing either for a concept that everyone knows to be sound or that never has or will exist. So it hardly matters what it will be called; it won't mean much.

I think we may be on the threshold of a new age of *artisanal software* – systems designed and built by individuals or small teams and that embody their aspiration, imagination, and also their individual style. Many of the achievements of the past decade – modern dynamic languages, modern programming environments, agile development, patterns, test-driven development, Web apps – work to extend the reach of the programmer/designer. For a time, too, people got tired of new software; they wanted to rest, to stick with familiar tools

[2]George Landow's ongoing work in *The Victorian Web* is an important and under-appreciated resource for observing this transformation in practice: a vast hypertext, edited over the entire history of hypertext by a single editor while moving across a host of disparate hypertext systems.

that did familiar things. I think that's coming to an end; we're prepared to learn about new tools if the reward is that we can do things we couldn't do before.

Dr. Claus Atzenbeck (http://www.atzenbeck.de) is an assistant professor in the Department of Software and Media Technology at Aalborg University Esbjerg, Denmark. His research interests are various structure domains, mainly spatial and navigational structures. Currently he works on representations of social structures.