# A Career in Informatics

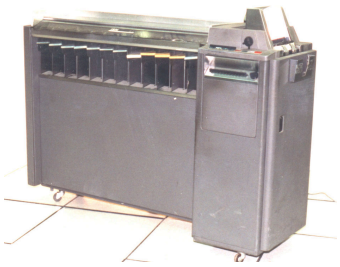
Gerry Stahl, March 2009

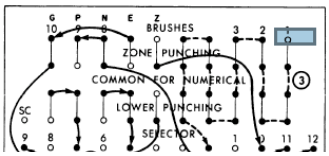
I grew up interested in science, math and philosophy. Perhaps the launching of Sputnik in 1957 contributed to this when I was 12 years old and finishing elementary school. I remember field trips of various kinds during my high school years, where I saw some of the US space rockets and the accelerator for training astronauts for high-g's during takeoff, as well as Univac where I saw the incredible wiring being put into new mainframe computers.


A friend of the family gave me a Brainiac kit to build my own computer when I was in junior high. By rotating pressboard circles with metal staples placed strategically, one could turn small light bulbs on and off, computing Boolean functions through simple wiring arrangements. I bought a second kit and added some more parts to program more complicated systems. The hardware was flakey and it took me decades to get more reliable computer hardware. But the Brainiac gave me good practice in binary arithmetic and a sense of computer architecture. When I was in high school, I read about formal logic systems and became a fan of Bertrand Russell, who took positivism to an extreme position.

I went to MIT to study math and physics, but got sidetracked into philosophy. It was more challenging for me and seemed to address the issues of science that really attracted me, as well as seeming to be relevant to social issues. Social issues had become more urgent as the McCarthyism of my youth in a conservative suburb turned into the student movement of the Vietnam years when I got to college.

At MIT, I took Noam Chomsky's course on social change and Hubert Dreyfus' courses on continental philosophy. Although the culture at MIT was strongly oriented to logical positivism, I was moving to the critiques of this perspective. I also took Marvin Minsky's introductory computer course and then his graduate course on artificial intelligence. Of course, there were no PCs in the early 1960's, but MIT had one of the first time-sharing mainframe computers. While Minsky lectured on neural networks, we students read the manual for an assembly language and a language like BASIC, punched programs on paper tape, submitted the tapes and got their output back the next day. For the AI course, I defined a term project of designing a program in LISP 1.5 to play Eleusis, a card game requiring inductive reasoning.






During the summer of 1965, I had my first computer job. I worked at a computer center at the U of Penn, across the street from where the first digital computer was built in the year of my birth. I helped a graduate student from India analyze his data on Indian politicians. He had a large stack of punched cards with his data. I programmed a card sorter, by arranging the wiring of the sorter (like an old fashioned telephone operator connecting calls) and running successive sorts. The next summer I spent in Baden, Switzerland, at a large electrical equipment factory. I wrote a

program in Fortran to do successive approximations to a calculus formula for electric charge distribution across a non-standard cable cross-section. The procedure there was to write the program on coding sheets and submit it for card punching and running over night. It would take several days just to fix the key punching errors, let along the programming syntax problems.

After college, I married and went to Heidelberg, Germany, to study philosophy for a year. There I began to understand many of the ideas that have become central to my recent writing on informatics theory.

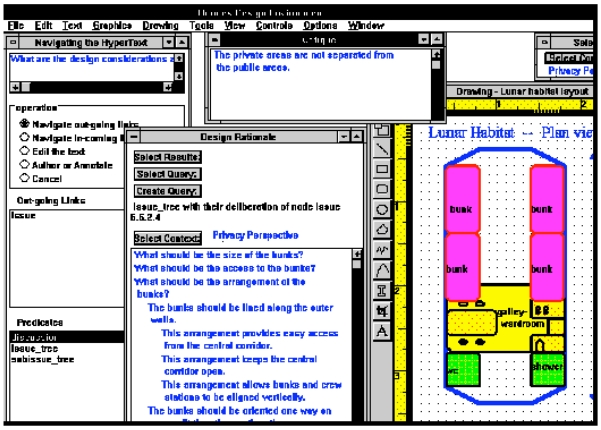
Returning to Philadelphia, I tried teaching math in the public schools without much success. I then became a systems programmer at Temple University on their two CDC 6400 supercomputers. These were state-of-the-art. In addition to programming in two assembly languages and Fortran, I had to operate the computers by myself on the weekend night shift to debug system software. This involved programming the boot sequence by setting binary switches, spinning up the dishwasher-sized hard disks and mounting data and program tapes on the refrigerator-sized tape drives. For fun, I programmed a 3-D tic-tac-toe game and a statistical sorting program for my brother's dissertation data.

I never felt really at home with the CDC 6400 operating system, the largest software system in the world for some time. So I went back to school at Northwestern and Frankfurt to complete my philosophy studies. My dissertation (Stahl, 1975) brought together Marx and Heidegger, the two thinkers who most directed attention to the social and everyday world as the foundation for thought and human life. While at Northwestern, I worked part-time at their computer center, also with CDC 6400s, along with the programmer of the world champion chess program.


Returning to Philadelphia, I resumed my job at Temple. I later took to the streets, becoming a community organizer in the neighborhoods and eventually settling in the Germantown neighborhood, raising dozens of grants for community development. At this time, the first personal computers were becoming available. I bought an Atari 400 to teach my kids programming. I used it to plan a revolving loan fund that capitalized the local neighborhood credit union. I later got an Amiga and taught myself C for doing computer graphics for an art video and some promo videos for my head-start software.

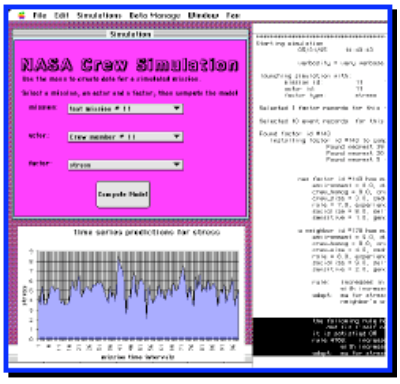
Shortly before the IBM PC with MS-DOS came out, I started a non-profit consulting service to computerize community organizations. From 1984-1989 I founded and ran the Community Computerization Project, serving dozens of community development corporations, neighborhood organizations, credit unions, energy conservation services, women's groups, university research institutes, and charitable organizations. I programmed custom database systems using dBase III and FoxPro on IBM PCs for client transaction tracking, fund accounting, and fund-raising; conducted organizational needs assessments for computerization and general management; installed hardware and custom or commercial software systems and trained personnel in their use. I designed, developed and marketed nationally software for Head Start reporting.

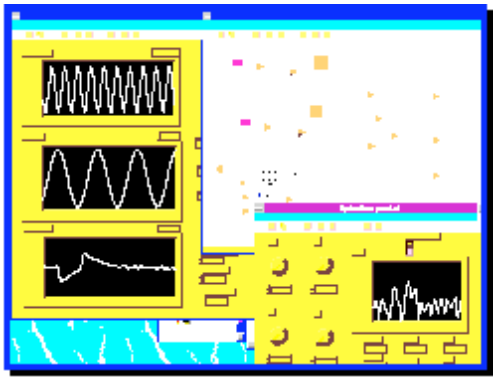
In 1989 I moved to Colorado and went to graduate school in computer science, concentrating in artificial intelligence and cognitive science. As a TA, I was the first to teach object-oriented programming to undergraduates there. I interned in US West's new research labs, programming interfaces in C++.



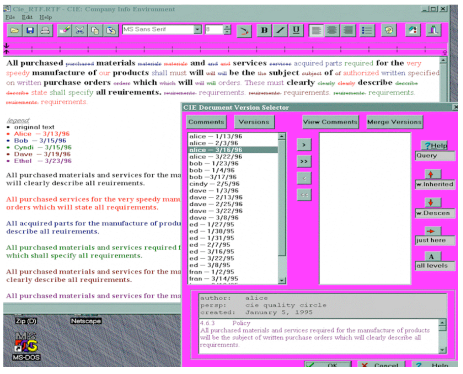

As an RA, I completely rewrote a prototype system for NASA design rationale, using object-oriented Pascal. This software evolved into both an online system for NASA's Manned Systems Manual and my own dissertation system, Hermes (Stahl, 1993). The software included an intelligent hypermedia system, an innovative mechanism for personal perspectives, and an end-user programming language for design rationale navigation that included a sophisticated query language with inferencing (see Stahl, 2006, Ch. 4). The custom object-oriented database incorporated virtual copying for efficient inheritance of object versions. I interviewed and videotaped engineers at a NASA subcontractor and also NASA astronauts, including the last man to walk on the moon, for domain knowledge about lunar habitat design.

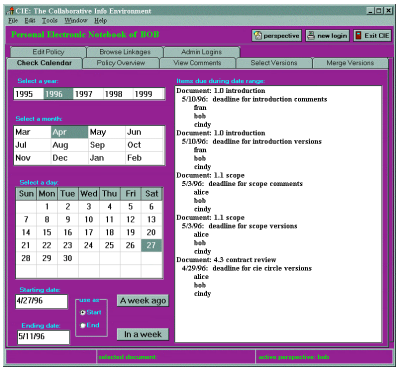
Upon graduation, I joined a start-up lab as director of research and development, working on both educational software and contracts with NASA. Major projects were the Teacher's Curriculum Assistant in Java (see Stahl, 2006, Ch. 1), the CREW system to simulate teams on missions to Mars in FoxPro (Ch. 3), and the neural net Optonet holographic stabilizer in MatLab.



Meanwhile, I developed a prototype business application based on my dissertation perspectives technology, called the Corporate Information Environment, developed in Java. I worked with an entrepreneur, but we did not succeed in attracting venture capital for this prototype.

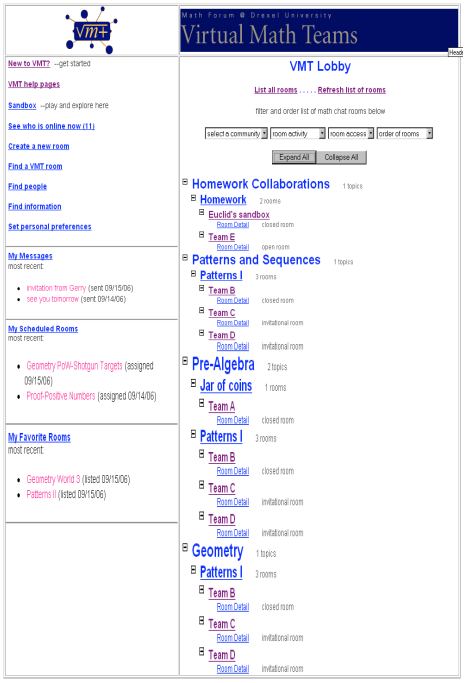


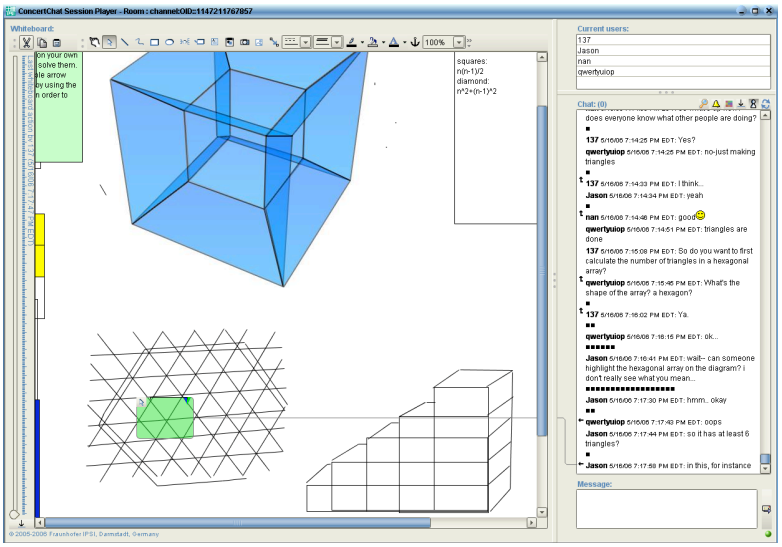


When project funding ended at the lab, I returned to the University of Colorado as a post-doc and then as a research professor. As a post-doc, I developed a number of web-based domain-oriented design environments and sketched other applications:

- WebNet: a design environment for a community of LAN managers (see Stahl, 2006, Ch. 5).
- State the Essence: middle school application for text summarization using latent semantic analysis—in Perl (see Stahl, 2006, Ch. 2).
- WebGuide: middle school application for threaded discussion with perspectives and modification of the threading structure—in Java (see Stahl, 2006, Ch. 6).
- SimRocket: a simulation of rocket launches used to support collaborative learning by 6th graders about data analysis—in Agentsheets (see Stahl, 2006, Ch. 12 and 13).
- AstroQuest: mock-up of a game for exploring the solar system and constructing knowledge using the web—in Lisp.
- Locations: mock-up of an interface to an organizational memory of weather station locations around the world and their identification codes.
- MedGuide: mock-up of a version of WebGuide for problem-based (PBL) collaborative learning by medical students.
- North Rim/Lake Valley Trails Committee: organizational memory of a community campaign for well-conceived open space trails.
- WebInquiry: a threaded discussion on the question, "Does the Web Support Student Inquiry?"

I decided to move to Drexel University for a faculty position, but first went to the CSCW lab new Bonn, Germany, for a year to work on a European Union CSCL project. There I led the design and implementation of BSCL, a threaded discussion system for schools in Europe—in Python (see Stahl, 2006, Ch. 7 and 8).



At Drexel, I have directed the Virtual Math Teams (VMT) project (see Stahl, 2009). The VMT environment integrates a Java applet supporting text chat and a shared whiteboard with a social networking portal in Java and a wiki using MediaWiki. We are currently scaling up the VMT software to support its release as a regular service of the Math Forum. I will not discuss here what we have learned from this project because we have published thousands of pages about it.

* * *

During the roughly 50 years of my interest and career in informatics, things have changed enormously. The year I was born, the electronic digital computer was also born. As I grew up, large mainframe computers were only found in government office buildings—working on number crunching for the census, military or space race—in major research universities or in large corporations, like insurance companies. These computers were limited to numerical manipulations, had continuous hardware failures, required large staffs and had severely limited power by today's standards. Personal computers evolved slowly, frequently changing operating systems, programming languages, application areas, techniques and cultures. A novice can now accomplish in minutes what used to take an expert weeks, if it was possible at all. Much of what visionaries once hoped for has been long since passed, and innovation now out-leads vision, with researchers falling far behind current realities. On the other hand, some expectations about intelligent behavior of computers and natural language understanding have advanced little in 50 years, except to realize how complex the issues involved really are. While one can now do wonderful things with computers, networked information and mobile digital devices, computers are still frustratingly hard to set up, learn to use and maintain.

My use of computers has changed dramatically. I used to worry a lot about hardware aspects: holes in tape or cards, wiring in circuits or between peripherals, processor down-time, movement of data between memory locations. Now the computer and Internet connections are reasonably dependable. At first, I used computers for special tasks, like manipulating numerical data or programming a visual pattern for a video game or animation. Now I do most of my work and communication on my laptop. Throughout the day, I cycle through multiple applications to maintain my activity at Drexel, with friends and in the international research community. My productivity in preparing for teaching, conducting research, writing papers and running an international journal is many times greater than what it could have been without computers.

My computer usage changes rapidly with innovations in the field. When the Web started, I created one of the first personal websites, which now has thousands of pages. I also started websites for several organizations. I maintain a blog and a wiki related to my journal. I use websites, wikis, Blackboard and VMT technologies for my classes. I coordinate four laptops, an iPod and an iPhone to maintain access to my email and my digital music collection wherever I am. I am continually integrating new applications into my computer-supported life.

For at least the past 15 years, it has been clear to me that the interesting issues in informatics have less to do with the design of computer technology and software as such than with the ways in which groups of people can interact with each other through the technology. My research has therefore increasingly focused on analyzing how small groups of people communicate and work using networked computers. This raises the age-old issues of philosophy, but within an interesting new context that sheds new light. Details of language usage in communication and shared understanding become explicit. Questions about knowledge, reality, cognition and society come to the fore.

In my career, I have been involved in many software development projects. It seems like each one required a whole new approach, new expertise and even a new language. Change in the

informatics field has been increasing at a doubly exponential rate, which means that people starting careers in the field now will have to deal with even a drastically increased rate of change. Technical skills that freshmen learn now—even assuming they are leading-edge—will be out-of-date in a year or two and obsolete before they graduate. To get by after college, they will need to be constantly learning.

In contrast to technological details, what I learned long ago about how to think, learn, read, write and compute still serves me well. The philosophy I read as a young man guides my research now, and I often return to the musty volumes on my bookshelf to reflect on what was said there by Plato, Hegel, Marx or Heidegger. I was always a self-motivated learner, and followed my interests, learning how to learn what I wanted to know about. I have spent many years in schooling, but I am equally engaged in learning outside of school. Always a good reader, I find myself writing more than I read now. Writing is hard work, but there is no way to learn how to do it well other than doing it. Even my high school penchant for algebra comes in handy now as I interpret the work of students in the VMT project and try to solve their problems myself. Perhaps it is such a mix of persistent themes with the churn of innovation that allows a career in informatics to be a rewarding and ceaseless adventure.

**References**


Stahl, G. (Ed.). (2009). *Studying virtual math teams*. New York, NY: Springer. Available at http://GerryStahl.net/vmt/book.

Stahl, G. (2006). *Group cognition: Computer support for building collaborative knowledge*. Cambridge, MA: MIT Press. Available at http://GerryStahl.net/mit/.

Stahl, G. (1993). *Interpretation in design: The problem of tacit and explicit understanding in computer support of cooperative design.* Unpublished Ph.D. Dissertation, University of Colorado. Available at http://GerryStahl.net/publications/dissertations/computer.

Stahl, G. (1975). *Marxian hermeneutics and Heideggerian social theory: Interpreting and transforming our world.* Unpublished Ph.D. Dissertation, Northwestern University, Evanston, IL. Available at http://GerryStahl.net/publications/dissertations/philosophy.