Dreams That Get Funded:
Programming Rolls its Own Reality[1]

Mark Halpern

My second resignation from IBM, in December of 1969, was precipitated by an offer from Kai Magleby, who had been selected by Fairchild Camera & Instruments to bring them into the computer business systems market. I gladly accepted his invitation to take charge of the software side of this new enterprise; it was very flattering to be asked, by a complete stranger who knew me only through my work, to take so responsible a position—and like most flattery, it proved hollow. Our financing was to come from the profits Fairchild expected from its main business, semiconductors, but 1970 turned out to be one of the periodic low points in that industry. At first, the hard times in the electronics/computer world were good for us; we were sheltered under Fairchild's wide umbrella, and were still hiring when other companies, particularly small startups, were laying people off or failing altogether. On one memorable occasion, some eight or ten people, the entire professional staff of a Sunnyvale-based software company, marched in to offer themselves as a unit; they had just gotten their last paychecks, and been told to deposit them quickly to make sure there were still funds to cover them. But within a couple of months the industry-wide recession pulled us down too; its profits dried up, and Fairchild aborted our fetal company.

They didn't kill us with a single merciful stroke. First came the waves of layoffs, eventually reducing us from a high of about 40 to a core team of about six. Then we tried—with Fairchild's blessings—to sell ourselves to other companies. We had some promising meetings with RCA, and were much depressed when these negotiations fell through at nearly the last moment. (This disappointment turned out to be a well-disguised blessing; within less than a year, RCA abandoned the computer business altogether. If our negotiations had been successful, we would have been relocated to the east coast, only to be dumped there.) Finally, about September 1970, Fairchild gave the few of us who still survived our severance pay, and we dispersed to find whatever jobs we could, with many promises to stay in touch and get together again when conditions improved.

What turned out to be available for me was the job of Director of Education in the Data Processing & Systems Administration of Crocker National Bank, in San Francisco. The nearly two years I spent there were not entirely without value for me, but have little light to shed on the early years of computing, so they will be passed over quickly here. One of the few events whose memory endures from that period of exile was my meeting with the late British publishing mogul Robert Maxwell, for whose Pergamon Press I was editing the *Annual Review in Automatic Programming* [2]. He was touring the United States, and his office had set up meetings for him with Pergamon authors and series editors in each city he was to visit; it was much like a king making a royal progress through his realm, giving audience to his loyal liegemen across the land. At the appointed hour, I appeared at his hotel suite in San Francisco, was taken in hand by his retinue, given a drink, and,

after a suitable pause to build up the drama, ushered into the presence. I remember little of my brief conversation with the great man, but my surviving correspondence with Pergamon suggests that the topic was my failure to provide them with a volume every year.

This failure was due to a number of things: the standards I tried to impose on my contributors in both style and substance (the native language of most, wherever born, seemed to be Broken English), the difficulty I had in getting working co-editors (as opposed to people who were willing to let their names appear as members of the Editorial Board), and the fact that I had a living to make. My reasons evidently cut little ice with Maxwell; I later heard from the Pergamon staff editor with whom I regularly worked that Maxwell had muttered something to him about finding a younger man to edit the series. Since no one at Pergamon seemed to feel that any action was called for in response to the mutter, I remained as editor until 1974, and got out one more volume. Maxwell was right on one point: a series calling itself *Annual Review* should produce a volume every year. But his diagnosis was wrong; the editing of such a series requires, if it is to be done both well and on schedule, something close to full-time effort, and few outside academia could afford such devotion to the task. It wasn't my age and consequent feebleness that was preventing me from delivering a volume every year, but the fact that I was editing the series in whatever hours were left over from a time-consuming if not otherwise demanding job.

Around the end of 1972 life began returning to the electronics and computing industries, and I left Crocker to rejoin that world as manager of a department ("Tymcom IX/Tymcom X Programming Languages") at Tymshare, a pioneer in public time-shared computer networking. Tymshare was then quartered in a number of small buildings located on Bubb Road in Cupertino, a town that had not yet become widely known as the heart of Silicon Valley. The San Francisco-based recruiter who got the company interested in me had only the vaguest idea where it was, and asked me with some trepidation whether I would be willing to work in what he evidently regarded as Ultima Thule. I was not only willing but eager; I was longing to get back into the real computer world, by which I meant the world of providers rather than users; I was quite happy to work in Cupertino, a 15-minute drive from my Palo Alto home; and I found the novelty of Tymshare's technology and market attractive. It was one of the first companies to build and market time-shared access, across a national network, to computer-based services that would otherwise be unaffordable to most of their customers.

Tymshare was highly successful, and growing at an unmanageable rate. While I was still completing the employment forms on tax deductions and health plans, and not yet formally a Tymshare employee, my new boss interrupted to ask me to put the forms aside so that I could interview a candidate for another position. This was an omen; when I'd had a chance to absorb the development plans that I'd been hired to carry out, I realized that more than just the rate of growth was out of control.

Tymshare's host computers at that time were a bunch of old XDS 940s that had the twin virtues of being stable and fully depreciated, and several PDP-10s from DEC. (The

network based on the 940s put the "Tymcom IX" in my department's name, that based on the PDP-10s explained the other half.) Little or no attention was paid to the 940s; they were just our cash cows. It was the PDP-10s that were to be our machines of the future, and it was for them that we wanted better languages, utilities, and application packages. I had been told while being interviewed for the job that my biggest task would be to direct the development of a "SuperFORTRAN" compiler based on a Fortran compiler being produced by DEC. What I did not learn until I had the job was that although Tymshare had committed to producing SuperFORTRAN by a firm date, DEC had not committed to a firm delivery date for the compiler on which it was to be based. Furthermore, we knew very little about that compiler, had no design documents for it, and did not even have particularly close relations with DEC.

Nevertheless, my group of six programmers had begun producing what they called a design for the structure they were going to build on the largely unknown foundation that DEC would someday supply, and I was handed the design notebook as if it were the Scrolls of the Law. What it contained was a collection of notes written in whatever formats the.programmers individually favored—one contribution was handwritten—with no uniformity of terminology or even of basic assumptions. This did not alarm me as much, though, as seeing that my boss was not alarmed. His unconcern became more understandable when I learned that he had little experience either in management or industry; he had come only recently from academic research, where such insouciance was acceptable, perhaps even desirable.

But I had much to learn myself. The design goals for SuperFORTRAN had been presented to me as givens, and I wince even now to think that I accepted them without question. Marketing had written an agreement with an important customer back East—a component of Bell Telephone whose name I cannot recall—on the properties it would have, and told us on the development side what we were to produce. As we worked, however, the list of goals changed and grew endlessly, and we sometimes found ourselves, at the end of the month, no closer to the objective than at the beginning. At last I grew desperate enough to do what I should have done immediately on taking over: I got upper management's approval to send one of my team back to the customer to learn directly, without Marketing between us, what he needed from us. My emissary took the red-eye flight to the East coast, and after talking to the customer for a few hours, called me to say "You know that list of eighteen features we've been trying to develop? Well, delete items two, four, five...." I have tried, since that experience, to remember always that developers must have direct contact with their end users, and take no one's word for what the users want. Often enough, even the users aren't sure what they want; but no one else even comes close.

I discovered that in order to do my job, I had to set up the same kind of direct contact with our vendor, too. DEC, although well advanced in the development of the compiler that was to be the foundation of ours, claimed to have little or no documentation that they could send to us. Since we desperately needed to know more about that compiler, I got them to agree to let us send one of our people to their main plant in Maynard, Massachusetts, to sit alongside their development team, attend their meetings, and

transmit everything he could learn to us in Cupertino. At first this visit was to be for just a week or two, but it became clear that our man would need to stay there for several weeks, perhaps even a few months, if he was to serve the purpose. He was understandably unwilling to leave his wife and young child alone for so long, so we had to fly them back East, too, and put them up at company expense for the duration. Eventually SuperFORTRAN got built, and may even have turned out to be a profitable product—no one could ever say for sure, since Tymshare was growing so fast, and making so much money, that no one could be bothered to do cost accounting.

For all its technical sophistication, Tymshare was perfectly capable of missing an opportunity for want of vision outside its own narrow area of competence. Sometime in the mid-1970s, Stanford Research Institute (now "SRI International, Inc.") decided to withdraw support from Doug Engelbart, a pioneer in the technology of the graphical user interface, the "windows" concept, and the mouse. He had won much recognition for his work in bringing these concepts to reality, but SRI declined to support his project much longer, and he was looking for a new home for it. Tymshare was offered a chance to acquire that project, and I was part of a three-man team—the other two were from our Marketing division—sent to SRI to investigate the Engelbart opportunity. My visit to SRI confirmed my belief that this was the most advanced work being done anywhere in making the computer usable by non-programmers—that is, 99% of the world—and that it would give its possessor a great advantage in reaching an enormous number of potential users. I said so in my report to Tymshare, but the two Marketing men, with the superior grasp of reality that comes from such a background, reported that it was not a commercially attractive proposition. Their views prevailed; Engelbart's team split up, much of it going to Xerox PARC, where their work was parlayed into the Macintosh GUI and other such developments of no commercial value. As a final touch, Engelbart himself was finally hired by Tymshare sometime in the late 1970's, but only after much of his best work had become the property of Xerox and, later, Apple, and without some of his key staff members.

I do not mean to leave the impression that there were no rewards for being an employee of Tymshare; there were many, in fact. One had the psychic lift that comes from being an innovator, and advancing the state of the art; of watching the value of one's stock grow; and of working directly with advanced technology. It was quite a distinction and adventure, back in the early 1970s, to have in one's study at home a terminal that put you on-line, almost any time of the day, to big computers and huge data collections. The equipment and capabilities were laughable by today's standards (as ours will be by tomorrow's): my home terminal was a Teletype machine capable of communicating at 300 bps, sending and receiving those bits through an acoustic-coupler modem into whose rubber cups one stuck one's telephone handset after hearing the squeal that indicated you were connected to the host modem. It printed connect-the-dots characters on rolls of greasy-feeling paper which was forever turning purple at the edges to warn you that you were nearing the end of the roll; it was noisy; it vibrated so badly that more than once I took its rattling and shaking for one of the seismic tremors that are a near commonplace along the San Andreas fault.

Nevertheless, it worked; it put a user in touch with his files at any hour, and from the convenience of his own desk. One evening in 1973 or thereabouts I was working at my home terminal when a message from the operator interrupted to tell me that my host was being brought down for maintenance in half an hour. I was momentarily anxious, because I had far more than half an hour's work to do—but then I remembered our Paris machine. I transferred my files to Paris, logged off Cupertino and logged in to Paris, and *voilá*! continued working happily away from my study in Palo Alto. We didn't need electronic games to feel like Masters of the Universe! The system even offered a rudimentary electronic mail facility; you could hold a halting conversation with anyone else who was logged in. I recall home-to-home exchanges with Faith Bugely, leader of a project I was part of, that let us practice a primitive form of telecommuting, even though carried on at the plodding pace of the Teletype. That pace had its good side, too—you talked only to people you really wanted to talk to, and wasted no time on the babble that fills most networks today.

I stayed at Tymshare for almost six years (January 1973—November 1978). During that time its revenues grew to about $300 million, and its shares were traded on the New York Stock Exchange. There was one flaw in Tymshare's vision, however, and it proved fatal. Tom O'Rourke, founder, CEO, and board chairman, had been one of the originators of the concept of timesharing mainframe computers via a public network, and had seen his dream become reality, and make him a millionaire. But by the late 1970s, the VAX and other minicomputers had emerged, offering an attractive alternative to timesharing as a way of buying computer power in small-to-medium portions. Quite a few of us urged O'Rourke to recognize this challenger, and to take the appropriate steps to make sure we were not left behind with an obsolete technology as the mini became the standard platform for many of our customers. But O'Rourke had other ideas; he sought to restore the company's fortunes by diversifying into application development for a variety of markets, acquiring a handful of smaller companies to give us the industry-specific expertise we lacked.

One of the application areas chosen was that of hospital information systems, and we acquired a small company called Medical Data Systems (MDS) in Mahwah, New Jersey, that had some experience in developing such products. But the difficulties of getting MDS to regard itself as acquired, aggravated by the 3000-mile distance between us, proved so severe that Tymshare got little benefit from its purchase. So in the event, the design of TyMed, our hospital information system, was done in Cupertino by several of us who had no experience at all with such systems, with the participation of just one of the experienced people from MDS—and he turned out to be less than an asset. Even our successive project leaders had had no experience in the development of products for hospitals, so we winged it; we read all the relevant literature we could get our hands on, studied the pioneering system that had been installed at El Camino Hospital in Mountain View, and set up as close a relationship as possible with our potential Beta site, Sequoia Hospital in San Mateo. Considering our lack of experience, we did a creditable job. I've been told that our design was later used as the basis of a successful product, and I remember our coming up with ideas in record-locking and two-stage commitment that anticipated some of the best work of database-management designers years later. But its

hasty acquisition of a subsidiary at the other end of the country, its failure to assimilate that company so that it became an actual asset, and its assignment of the product design task to a cobbled-together group at headquarters that had to learn its job as it went, were all symptomatic of Tymshare's loss of direction and increasing desperation.

Tymshare profits began to sag in 1982, and in 1984 the board forced O'Rourke to sell out to McDonnell-Douglas, which was expanding and diversifying with all the money it had made building Phantoms. McDonnell had no idea what to do with Tymshare, and was further handicapped by its geographical remoteness in St. Louis. They began selling off various parts of the company, cancelling development projects, and otherwise killing the goose, until, a few years later (I had long since left), there was nothing left of Tymshare except the network, Tymnet. This last asset was finally sold off to British Telecom, and with that sale ended the odyssey of Tymshare, a company whose brief history exhibited all the excitement and all the sorrows of the California high-tech entrepreneurial dream.

When it became apparent to me in 1978 that Tymshare was on the descending slope of the curve, I found another job as a compiler-department manager, this time in Berkeley, with the Western Development Center (WDC) of Datapoint. That company, headquartered in San Antonio, had contracted with a small Berkeley software group to build a Cobol compiler for its line of networked microcomputers. When the product was successfully delivered, Datapoint was sufficiently pleased to buy the little group outright, naming it their Western Development Center, and making its founder, a UC Berkeley computer science professor named Herb Baskin, a Datapoint VP. Baskin and I hit it off personally, and I joined him gladly, even relocating from Palo Alto to the Oakland hills so as to be closer to what I felt sure would be my office for many years. Here, too, I neglected to ask some important questions, and overlooked some important problems. I'd had enough experience by now to know that in so rapidly moving a technology as software, good communications between developers and management are an absolute necessity, and are difficult to achieve under the best of circumstances. With company headquarters separated from us by two time zones (and sometimes another hour due to daylight savings time), it was almost impossible to communicate even in the simple sense of finding the right person at his desk, let alone in the deeper sense of achieving real rapport.

The first assignment Baskin gave me was to fire two or three members of the department I'd just taken over: the Cobol compiler they had built was both slow and buggy, and he was furious with the people who, in his view, had failed him through incompetence or sloth. I asked for time to study the product, and to talk to the people involved, before taking any such drastic steps. Baskin was reluctant to delay what he saw as the necessary house-cleaning, but we were still in the honeymoon stage, so he gave me a little while to familiarize myself with the situation and the people; perhaps he accepted my view that it was bad form to fire someone whose name you weren't yet sure of.

My talks with my victims-to-be, and my inspection of the product and the bug reports submitted against it, convinced me that Baskin's position was mistaken on all counts. The programmers were of mixed quality, but not incompetent or lazy. The faults of their

product were due to inadequate planning and management more than anything else. As for the bug reports, no one had yet subjected them to a systematic analysis, and Baskin had simply taken them at face value. My programmers and I went through the pile, sorting them into categories such as "duplicate," "unreproducible," "user error," "documentation problem," and so on. We found that only 40% of them reported real new software problems. (My experience, and that of others I've discussed the matter with, suggests that this value is quite typical for the percentage of real bugs in a sizable collection of raw bug reports.)

Datapoint was enjoying an *annus mirabilis* when I joined them, its stock often climbing by several points per week. The programmers who'd been with Baskin when his group was acquired by Datapoint had all been given stock options, and many had paper profits well up in five figures by the time I'd been with them for six months. Then a storm broke out of what had been a clear blue sky: it was suddenly discovered that many of Datapoint's reported sales figures were fraudulent. The company had, it turned out, put such intense pressure on its sales force to meet and exceed quota that some salesmen and even high-level sales managers had resorted to reporting sales, and taking delivery of machines, without any real customer on hand. The machines so ordered were stashed in warehouses in various cities, apparently in the hope that real customers could be found for them before the cat got out of the bag. But the scheme rapidly unraveled, and became an ongoing story in the *Wall Street Journal* for weeks[3].

Datapoint stock plunged, and with it the dreams of many of the programmers. Top executives of Datapoint were fired; even the Sr. Vice President for R & D, who so far as I know had had nothing to do with the scam, left. Not only was there no prospect now of original development work at WDC, the very existence of the facility was rumored to be in jeopardy. (The rumor turned out to be correct; within a year or so WDC was trimmed back to a few contract programmers, then abolished altogether). I was therefore quite receptive to an offer in February of 1980 from Mohawk Data Sciences of the managership of the language development department in their Los Gatos development laboratory. Despite the lesson I had just had at Datapoint about the impossibility of doing important original development for any company at an outlying site, I once again accepted that situation; Mohawk was headquartered at Parsippany, New Jersey, about as far as one could get from Los Gatos and still be in the contiguous United States. The story of how MDS, along with their acquisition, Qantel Business Machines, failed is the usual story of dithering, out-of-touch top management, more intent on playing executive-suite games than understanding their market and technology, and does not deserve relating in the few pages available here; the business press for the period 1980-1982 can be consulted by anyone who insists on the sordid details. Suffice it to say that by the end of 1982 I was once again a sailor whose boat was sinking beneath him.

Luckily, the computing industry was in a healthy phase at the time, and the position that was offered to me when the Mohawk/Quantel episode ended was quite different, and its story far more interesting. A recruiter called in December 1982 to ask if I'd consider joining a software startup down in the peninsula. This was something new, and therefore attractive. I'd so far in my career worked only for well-established companies, or, in the

case of the Fairchild startup, for an enterprise that was at least under the wing of a large company. But what was described to me now was a classic Silicon Valley entrepreneurial situation: a tiny group of ambitious techies with a good idea, and financing from venture capitalists.

The company that I went down to Mountain View to see in January of 1983 was called, at the time, Dialogue Systems, but it soon changed its name to Enhansys, and I'll call it that throughout. It consisted then of eight or nine people, about half of them professionals. Their quarters were modest; the woman who served as secretary and receptionist used as her office what was actually a large storage closet. To keep her from claustrophobia and depression, someone had kindly pasted a *trompe l'oeil* poster to the wall she faced, so that she seemed to be looking through a window out onto a lovely bay. All possible nervousness and formality on either side was abolished when my first interviewer had a profuse and sudden nosebleed just as we began talking, and bled freely over my newly-polished shoes. The company was both small and in its infancy; the venture capitalists (Mayfield Fund was the lead investor among these VCs, as they're commonly called) had given it just enough money to produce a complete product description, and promised more if that was delivered in good time. So I was in time to participate in nearly all the steps of bringing a new company into being; the only ones I missed being the development of the business plan, and the trudging around to VCs to get some seed financing (I added this experience to my *curriculum vitae* a few years later).

The software-product idea on which Enhansys was based was for a tool for shrinking a heterogeneous computer network down to a user's local disk. The need for such a product had occured to the two founders, Tim Sherrod and Jake Sredni, while they were with Advanced Micro Devices, Tim as an MIS manager, Jake as an industrial statistician. They had noted how engineers responsible for the performance of a semiconductor fabrication line often needed data from several disparate machines on the company network. Some of that data might reside on an IBM mainframe, some of it on HP machines or VAXs, and so on. Further, each machine would have its own database management system whose proprietary formats were incompatible with that of the others, so that even when all the data objects had been tracked to their host, and downloaded to the engineer's workstation, he still had to do considerable processing just to get them all in some semblance of standard compatible form.

Tim and Jake's idea was a software product that would do all this for the long-suffering engineer. He would simply tell it what items of data he required; it would locate these items, pull them down to the workstation, and put them in a standard tabular form with no further effort on the user's part. Once downloaded and formatted, the data could be analyzed and depicted by Enhansys' statistical and graphics tools. Tim and Jake had tried to persuade AMD to let them develop the system there, and been turned down. They decided it was too good to abandon; they wrote a business plan, got some seed money, and set up shop. I not only thought the product was a good one, but saw in it an opportunity to use some of the ideas that had grown out of the XPOP project[4] many years earlier. I joined Enhansys quite happily.

Anyone who has followed these memoirs will have noticed that many of my jobs began with some incident that was prophetic of their ends; an incident whose significance, however, became clear to me only much later, when it could no longer serve as a warning. At Enhansys the significant incident was a report delivered to us by a marketing consultant who told us that the semiconductor industry was indeed a good market for the product we were developing, but that the petrochemical industry was even better. He showed us some impressive data to support his conclusion: number of installations, amount spent on capital equipment per professional employee, type of computing platform used, and so on. The reaction of the Enhansys principals was to shelve this information, and continue to pursue the semiconductor industry as our sole market. They felt comfortable with that industry; they knew the jargon, the players, and the game, and they didn't want to start off by switching marketing targets to one they were less familiar with, or diluting their efforts to chase two at once. This seemed reasonable enough, and we rolled up our sleeves to do a complete external specification.

Although I was a contributor to the external specification effort, I was principally responsible for an implementation plan. I had convinced Tim that the XPOP approach would give our product a degree of flexibility that we would later be glad to have, and that the existence of a running XPOP would give us a head-start toward getting the product up and running. There was an initial false start here; I attempted to get XPOP itself, which had been implemented in full only on the long-obsolete IBM 7094, up and running again. This involved locating an operational 7094 in Seattle[5], flying up there with a tape, and trying to remember procedures that I hadn't performed in fifteen years. It was a fascinating experience to work with the zealot who had bought a 7094 at scrap prices from Boeing just before it was to be junked, and nursed it along with the traditional baling wire and chewing gum long after IBM itself had stopped making parts for it, and forgotten how to maintain it. It was magnificent, but it was not computing. After some weeks of alternating elation and frustration, we stopped trying to breathe life into old tapes and old machines, and set about developing a new processor that would have the chief properties of XPOP, but transplanted into IBM's then modern VM/CMS environment.

We succeeded. It took about twenty of us two years to make the product (which, like the company, was called Enhansys) sufficiently stable and functional to put into the hands of users. As soon as the first customers laid their hands on it, of course, a torrent of bug reports descended on us, together with complaints about performance. This caused the usual conflict between the developers, who wanted to spend most of their time on new functionality, and marketing, which wanted bugs fixed and performance improved—and new functionality as well. These problems, of course, are the marks of an immature but essentially successful product; no one bothers to document the bugs or complain about the performance of a product unless it's doing a useful job, and giving promise of doing an even better one. (It's another tribute to the product's technical quality that, years later, some users continued to work with it even after Enhansys had dropped it; a few of the developers set up a consulting firm that supported the installations that Enhansys had abandoned.)

The language-processor that was the core of the product was essentially my conception; it was built mostly by Dan Lofgren, a clever young programmer who did practically all the implementation, as well as much of the detailed design. Internally, it was a combination of interpreter and library of compiled code that gave us, I believe, an unusual combination of flexibility and efficiency. The interpretive part was a collection of constructs called boxes—these were advanced macros, consisting exclusively of compiler directives—together with the processor that interpreted them. (This technology was a direct outgrowth of XPOP, but Tim dubbed it 'Phoenix' to avoid tipping our hand to any competitor who might remember XPOP). Each box corresponded to an Enhansys command, and specified to the interpreter how to parse the command, what values were to be supplied as defaults if the commands failed to offer certain arguments, and so on, thereby enabling the interpreter to build fully parameterized calls to the underlying library procedures. (Space considerations have forbidden the inclusion of examples here; readers wishing to see some can get them direct from me—see the address information provided in the biographical material at the end.)

Obviously, the user would not want to have to create boxes himself in order to realize such fundamental operators as "+" and "-", and of course he did not have to; these and many others were supplied as part of the product. He could have done so had he chosen to, however, and the significance of this fact lies in what it implies for the extendability and flexibility of the language in syntax, semantics, and even in pragmatics—the ability, for example, to specify such things as the trimming or padding of the emitted object so as to match some other structure in whatever shape it might have at the moment this operator was invoked. There were some 34 types of directives, which together constituted a full programming language, with the ability to create variables of a number of structures, assign values to them, branch on a variety of conditions, and so on—but tailored very closely to the single application of mapping a very wide variety of statements into correctly parameterized calls on a library of pre-compiled routines.

The process of interpreting a box generated the calls to the appropriate compiled code routines, which then did what was called for by the command. The great accomplishment of Phoenix was to decouple the user-visible part of the total processor from the great chunks of code that did the dogwork. All the features that were important to the user were controllable from within the boxes, uncluttered by any irrelevant matters. By the same token, all the routines that did the heavy lifting were in tight compiled code, out of the user's sight.

One useful consequence of this semi-interpretive design was that a great deal of control could be exercised over Enhansys just through knowledge of the Phoenix language, the body of directives for building boxes. We expected that sophisticated users would be able to augment and customize Enhansys simply through the creation and modification of boxes, but we never got the customer base that might have done this. We did benefit from Phoenix in other ways, though. At least a few of our marketing and sales people were able to manipulate the product by means of that technology (unfortunately, they had plenty of time to do this). When they wanted to have the product changed in some way, they found that instead of trying to describe to the developers what they wanted, and then

get the proposed work scheduled, they could often use Phoenix to build a version of the product that incorporated the desired feature. They could then *show* the developers what they wanted for the product instead of talking about it; we could run the released version against the proposed one to evaluate them; and if a customer were the source of the request, we could show him unusually quick response to his needs.

Despite its success, Phoenix suffered from the bad name attached to all interpretive systems; most of the company's developers had learned in Computer Science 101 that interpretation was an expensive, indeed prohibitively expensive, process. Every time a fresh batch of features was added to Enhansys, and a villain was needed to account for the inevitable performance degradation, Phoenix was always the first of the usual suspects to be rounded up. Because it required that every Enhansys command be partially interpreted every time it was used, Phoenix was seen by some at the company as the cycle-hog that was making our product unacceptably slow. Over the years, it was investigated several times (I was excluded from the investigations) to determine if it was to blame for our less-than-satisfactory performance. The cost of Phoenix was computed mathematically, and measured empirically; it was exonerated every time. The Phoenix approach would no doubt have been unacceptable in, say, a small utility routine, but the total Enhansys product was so huge and cycle-hungry that the cost of Phoenix was barely perceptible, like a billion dollars in the Federal budget; its flexibility was purchased at a cost that was so small a percentage of the total cost of running the product that it was negligible as a consumer of cycles.

So the Enhansys product was a technical success; but technical success and marketing success are of course two very different things. It was released in 1985: just in time to confront one of the worst of the semiconductor industry's bad times. The industry was again laying people off, shutting down chip fabrication facilities, watching the "book-to-bill" (the ratio of new orders booked to current sales) drop below unity and continue sinking. This recession had not occurred overnight, of course, and we had had plenty of warning that such a market might greet us when we released Enhansys, but somehow we were not prepared for it. As I noted earlier, there was strong rapport between our founders and the semiconductor industry; it had earlier made them ignore the marketing research (for which we had paid handsomely) that told them of a better primary market for our product. Now this bond proved so strong that our management evidently preferred to go down to defeat with that industry rather than redirect our marketing efforts to an alternative target.

Much lip service had been paid, during the months in which we watched our targetted customers fall on bad times, to the idea of redirecting our marketing plans, and there had been plenty of time to do so had there been the will. In an effort to jumpstart the process of shifting our marketing toward the petroleum industry, another appropriate target, I suggested that we approach a few recently retired oil industry vice-presidents, men who knew the petroleum industry as our management knew the semiconductor industry, and offer them commissions to open industry doors for us, teach us industry jargon and folkways, and generally put us into position to market our product to that industry; petroleum, if not enjoying boom times, was at least not in the pit that Silicon Valley was

in. My suggestion was ignored; I was a techie, not a practical business person. Talk about diversifying our marketing efforts continued, but in reality we did nothing but chase semiconductor companies that were more concerned about avoiding Chapter 11 than about acquiring expensive ($40,000-$100,000) new systems for controlling their fabrication processes.

In more than one way, then, the reality principle took a terrible beating at Enhansys, especially at the hands of those who who prided themselves on being practical men, and who, unlike the propellor-heads who were building the product, enjoyed the respect of the "financial community." We made good progress on another front, though, that of high-mindedness. Our founders were not content to build just another profitable company; they wanted one with a "culture" also. It was to be a Brook Farm populated by the pure of heart, as well as a business that was someday to go public and let its owners cash out big. There were several company meetings at which we discussed the ethical standards a potential customer would have to meet before we would sell our product to him; we developed a blacklist of "politically incorrect" companies and industries that was longer than our list of actual sales prospects. In the event, we need not have worried; we were not beseiged by evil weapons-makers or environment-pollutors demanding our product—nor by many others, either.

Sometime in 1985 I was called in by the CEO—not one of the founders, but a financial man installed by the investors—to be told that Tim and Jake were out of the company. There had been a trial of strength between them and the CEO, and to what should have been no one's surprise, the moneymen's representative had won. Not long afterwards, the victorious CEO himself left "to pursue other interests." He was replaced by another financially-savvy but technically ignorant type selected by the financiers—no matter how often this type fails to rescue a high technology company in trouble, the VCs send in another—and this man's formula for success was to abandon the flagship product althogether—it wasn't "world class," he told us—and concentrate on what had been an adjunct to it, the software that imported data from wherever on the net it was located. The new Enhansys lasted a year or two, then disappeared entirely. I will spare both the reader and myself a recital of the painful details of Enhansys' decline and fall; successful companies are successful in various ways, but unsuccessful companies are all unsuccessful in the same way: poor sales. As it died, it was my miserable duty as a manager to lay off some people I liked and respected, and finally to be laid off myself.

I did something unusual for me on the occasion of this layoff: I stopped to think. Whenever previously I had found myself unemployed, I automatically sought another job like the one I had just lost—a software-development management position. But this time I stepped back to ask myself what I really wanted to do, and came up with an answer that rather surprised me. For some years I had wanted to write a book—I had actually drafted a couple of hundred pages, and gotten some comments from friends on them—but had never had the time or energy to bring the draft to completion while working as a software manager. I'd also, while developing Enhansys, gotten my first PC (one of IBM's early two-floppy, no-hard-drive Portable PCs), and while using it, conceived a novel kind of PC software product. I very much wanted to develop this product, but realized that this

was impossible while working the normal 50-hour week for someone else (the figure named is an average, of course, not the maximum by any means). So I had two reasons to want a job that could be handled in the 40 hours per week that is the norm in many industries, but is thought of as part-time work in the frantic world of software development.

The clear alternative for me was technical writing. I had always done a good deal of it, plus editing, plus management of these activities: I'd co-edited three volumes of Pergamon's *Annual Review in Automatic Programming*, contributed articles to Van Nostrand's *Encyclopedia of Computer Science,* and published several papers in the refereed literature, such as *Communications of the ACM*. So, with some hesitation at first, but then with growing certainty of the rightness of what I was doing, I got in touch with recruiters, told them I wanted to be a technical writer, and in July of 1987 got hired in that capacity by Tandem Computers, Inc.

This story changes character here: it is no longer what I was doing for my employer that occupies the foreground, but the projects that I had chosen for myself. The book I wanted to write was to be a pulling-together and summing up of my views on such topics as artificial intelligence, the evolution of software-development technology, the relationship of mathematics to programming (with special attention to the idea of formal correctness proofs), the development of bug-free software, and a few others. Much of the text already existed in one form or another, since I'd published papers on several of these themes over the years, but a great deal of work remained to be done in updating and relating this materiel so that it read more like a book than a collection of miscellaneous papers. I was convinced that such relationships existed; I felt, although I could not then have articulated my reasons for feeling so, that there was a real underlying unity to be found in these papers. In the event, I found so close a set of relationships among the topics and papers out of which the book was composed that, as I said in its preface,

> I have had to discipline myself, in fact, to hold to a reasonable number the cross references that I inserted whenever I found that a point I try to make in one chapter is supported or amplified by an argument offered in another. The problem has been not to find their common theme, but to retain some autonomy for the several essays the book comprises.

The book got published[6], although the publication was of a type that didn't much involve the public. Ablex Publishing Corporation, I discovered only after the book appeared, does not stoop to anything so vulgar as the purveying of books to the riff-raff who frequent bookstores. Instead, like gentlemen, they acquire a list of standing orders from institutions—libraries, universities, etc.—and simply distribute the books they publish to these institutions, along with invoices. They price their books high enough to ensure that the revenue from this captive market suffices to recover their costs and make a reasonable profit; any copies sold beyond that are icing on the cake. Of course they do not advertise, but sometimes word leaks out, despite their discretion, that such-and-such a book has been published, and anyone enterprising enough to track down the publisher, and initiate correspondence with him, is rewarded by being allowed to purchase a copy. One such

seeker tracked me down via a national electronic mail system to which we were both subscribers, and asked me piteously, if I were the author, to tell him how to get a copy. I have had the gratification of knowing that there are no wimps or dilettantes among those who have personal copies of my book; anyone who owns one has paid his dues, in every sense.

So the first objective I sought in changing jobs in mid-career, that of getting my book out, has been attained, if in a slightly left-handed way, Whether the second objective, that of getting my software conception turned into a marketable product, will also be reached remains to be seen; I hope to be able to report success some day in a fourth installment of this memoir.

REFERENCES

[HAL90] Halpern, Mark, *Binding Time: Six Studies in Programming Technology & Milieu*. Norwood, New Jersey: Ablex Publishing Corp., 1990.

[KEN91] Kenner, Hugh, "Poets and Sleepwalkers," *BYTE* (April 1991), pp. 392-393.

[STR85] Strehlo, Kevin, "The Day of Reckoning is Coming," *InfoWorld* (July 1, 1985), p. 8

[WIN82] Winans, R. Foster, "Datapoint's Plunge Leaves Big Holders Queasy But Some Others See an Opportunity, Move In," *Wall Street Journal* (May 13, 1982), "Heard on the Street" column.

NOTES

Note 1: This is the third episode of a memoir of a career in computing that began in 1957. The first episode appeared in this journal, Volume 13. No. 1 (1991); the second in Volume 14, No. 1 (1992). The title is taken from a remark made by my friend Bob Brill when I objected to some proposal of his as not showing much sense of reality: "What is reality but dreams that got funded?"

Note 2: I was the primary editor or co-editor for three volumes of the *Annual Review*: volume 5, for 1969; volume 6, for 1971; and volume 7, for 1974. The story of how I became editor, the names of my fellow-editors, and other details can be found in the second installment of this memoir: "Turning into Silicon: Further Episodes from Programming's Early Days," *IEEE Annals of the History of Computing*, Vol. 14, No. 1 (1992), pp. 61-69.

Note 3: The Datapoint debacle was the subject of almost daily stories in the *Wall Street Journal* between March and May, 1982; for an example, see [WIN82].

Note 4: A summary description of XPOP is given in the second installment of this memoir, along with references.to more detailed treatments.

Note 5: This was the Puget Sound Computer Service (Herbert Burke, Owner), the funkiest computer installation I've ever seen. Its two-room storefront quarters were located amid launderettes, Mom-and-Pop all-night groceries, and other such unpretentious necessities of urban life. The front room was the business-like, for-show part of the layout; it actually had a few desks and chairs, and gave the impression that an unsuccessful garage sale was in the process of winding down. The 7094 was in the back room; there the ambience was part Victorian lumber room, part mad scientist's laboratory, part Batcave. Herb's business card read "Serving the Data Processing Community Since 1966"; it could have added "... and with the Same Equipment!" For more amusing details on Herb and his business, see [STR85].

Note 6: I have some reason to think that if the book is ever called to the attention of the civilized world, and made reasonably easy of access to anyone wishing to read it, it may enjoy some success. Hugh Kenner discussed it at some length in one of his columns (see [KEN91]); Richard Hamming wrote to say that "[your] book arrived Tuesday morning. Early that afternoon...I began to read it. The best remark that I can make is that I set everything else aside and read it straight through!" (personal letter, November 8, 1990). Stan Kelly-Bootle, author of *The Devil's DP Dictionary* and the column "Devil's Advocate" for the *UNIX Review*, said some flattering things about it in his column.