

# What Does the Future Hold?

*Howard Shrobe*

I was asked to give a visionary talk about the future applications of Artificial Intelligence technology; but I should warn you that I'm actually not very good as a visionary. Most of my predictions about what will happen in the industry don't come true even though they ought to. So I'm not going to tell you what the future holds; what I will do is to point out some of the technological trends that are at work.

The outline of the talk is as follows: I'll start off by looking at the previous IAAI conferences and reflect on what we've learned from them. Then I'll look at what's changing in the hardware base that sets the context for all the computer applications we do. I think that will lead to interesting new viewpoints. Next I'll sketch what applications might arise from this new viewpoint. Finally, I'll discuss how the development of practical applications ought to interact with the scientific enterprise of trying to understand intelligence, in particular, human intelligence.

As I go on, there will be a lot of points where this talk contacts with other talks that will be presented at this conference, and I'll give you forward pointers to all of those, at least the ones I've thought of.

Let's begin by looking back over the previous 11 years (counting this year) of AI applications presented at this conference. I want to ask a few questions about these applications. The first is, "What were the applications like, what did they actually do?" The second question is, "What was the economic case made for those applications; why were these successful; what were their key attributes?" Finally, I'll repeat something I did a few years ago when I was chairman of this conference, which is to look at how the program chairs of previous conferences conceived of the conference.

Even before the Innovative Applications of Artificial Intelligence Conference was started, at the first AAAI in 1980, John McDermott

(then of CMU) presented a paper called "R1: An Expert in the Computer System Domain." As many of you know, the name R1 came because of the joke, "I didn't used to know how to spell 'engineer,' and now I are one." When Digital actually deployed R1, they changed the name to XCON so that they didn't have to keep repeating the dumb joke. What was notable about the R1 paper was that it was a knowledge-based system, a very large and complex one, with technical sophistication, significant coverage of the domain, and it performed very well.

There was another notable aspect of R1: It wasn't sponsored by the government. It was sponsored by a corporation, Digital Equipment Company, at that time the second largest computer company. They sponsored the project not for the love of pure science, although I'm sure they had that in mind, too, but because they had a real problem that they wanted solved. This program solved the problem, and it became critical to corporate operations. What the program did was to take an order for a computer system and figure out what components were needed, how they packed into cabinets, what cables were needed, and so on. This was a major problem in the era of large minicomputer systems. We're giving an award to John for that work, as we well should, which will be presented Wednesday morning.

In 1989, we decided to launch the IAAI Conference. Raj Reddy, who was the president of AAAI, decided we should start the conference. I was conference chair at the time, but I found out about it only after he had convened the Program Committee; I asked if I could be on the program committee, and I've been on every one since then. The first conference was held separately from the national conference on AI. It presented 30 application papers selected from over 100 submissions. If you look at the authors and sponsoring organizations of these applications, it's a sampling of

## In 1989, The First IAAI

- 30 Application papers selected
- Corporate Giants spanning the Economy
  - Manufacturers Hanover, IBM, Alcoa, GM, Arthur D. Little, Pac Bell, duPont, Ford, DEC, NASA, Boeing, Bellcore, US Navy, MCI, Met Life, American Express
- Proceedings Later Republished in Book Form
  - 27 Applications published
  - American Express Authorizers Assistant one of the missing
  - MCI another MIA (funds transfer application)

Figure 1. Overview of the First IAAI Conference.

## Mix of Applications: Ten Years

Manufacturing and Design	30
Business Operations	30
Finance	25
Diagnostics and Troubleshooting	12
Telephony	11
Claims Processing and Auditing	12
Computers and Software Engineering	7
Military	8
Information Retrieval and Classification	6
Space	5

*Others:* Customer Service, Sales Support, Personnel, Crisis Management, Agriculture, Music, Environmental Monitoring

Figure 2. A 10-Year Mix of AI Applications.

the largest corporations and government organizations. You can see that list in figure 1.

The proceedings were rebound later in a book form. Although there were 30 applications presented at the conference (and in the original proceedings), there were only 27 applications in the book. You might guess that a couple of people got embarrassed and pulled their papers because the applications failed, but you'd be wrong. These were actually all

successful applications. Two of the three applications that weren't in the book (I've forgotten the third) were the American Express AUTHORIZERS ASSISTANT and a FUNDS TRANSFER application done by MCI for a large commercial bank. The reason these applications weren't in the book was not that they didn't work, but they worked too well. They were extremely successful. As far as I know, the AMEX application is still running; it has been deployed for well over 10 years, and it is still critical to the company's operations. The reason AMEX wanted the paper pulled was that the application was so successful that talking about it might divulge things that they considered corporate secrets. In particular, the number of transactions and some aspects of the business practices embedded in the application were very sensitive.

This has been an ongoing issue. At breakfast this morning, somebody quoted to me a statement about another conference that has the same problem: "We don't want to tell you about our successful applications because that would be giving things away, and we don't want to tell you about our failed applications because that would be embarrassing." Thus, it's hard to get papers submitted; what else are there besides successes and failures?

Over the 10 years that we've been running this conference, there has been a relatively constant mix of the kinds of applications presented (figure 2). The categories are, of course, relatively fuzzy, but the bulk of these applications are decision aids, situated in the general flow of the business process of an organization, and they are aimed at boosting productivity. This is critical to the argument I'm about to make. This year, as Samy said in his opening remarks, the mix is somewhat different: there are some agent-based applications that operate in real time, some real-time planning and scheduling applications, and some personal assistant application. I think that's a harbinger of new directions.

Now let's look at what previous program chairs thought the point of the IAAI conference was. At the first conference, we said that the emergence of scientific achievements had triggered opportunities to tackle new problems. Although we had not yet achieved the goal of automating cognition—and that's still true a decade later—the technology is nevertheless making a major impact in the everyday operations of large organizations. The point of the conference was to exchange information about what really works and what the real problems are. The goal was to lead to better technology, to find and remedy current deficiencies, and to solve real problems.

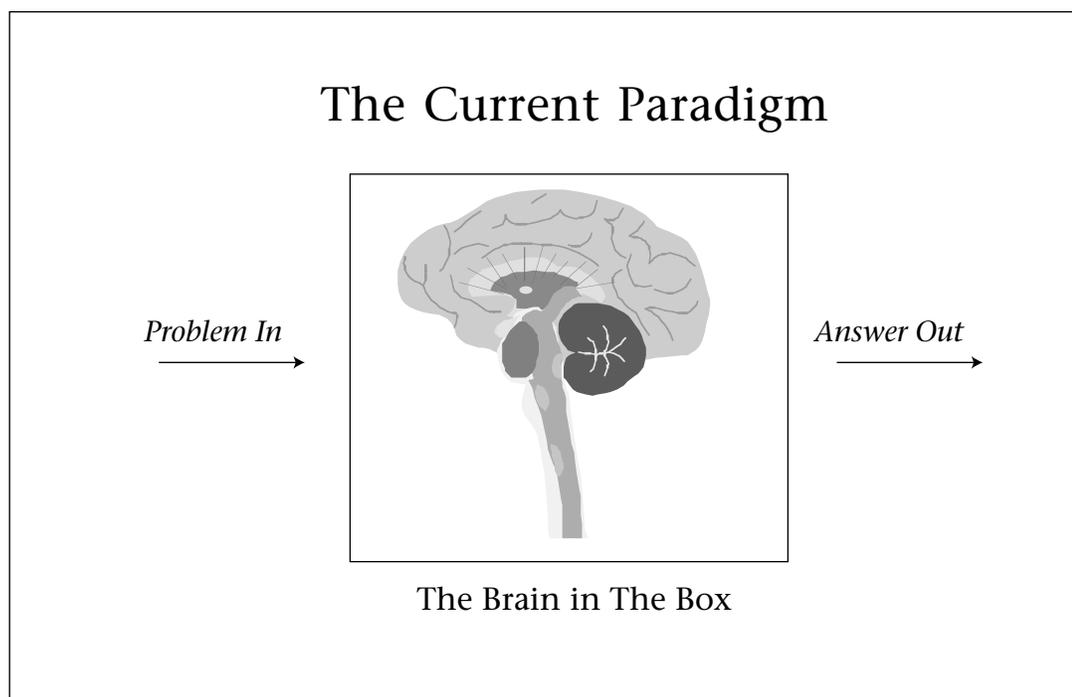


Figure 3. *The Brain-in-a-Box Paradigm.*

The introduction to the proceedings of the second conference says that the point was to demonstrate the utility of deployed applications. Now it is interesting to compare these two perspectives: At the first conference, we said that scientific achievements have motivated and set the stage for applications. At the second conference, we were, in contrast, clearly making a political point: We wanted to demonstrate the utility of the applications and to show that they are well integrated with the existing computer and corporate environments.

The introduction to the proceedings of the third conference says that we are making a case for the business person: We are trying to show what can be done with the current technologies and to solve practical problems. It goes on to say that research is driven by applications and that by highlighting application successes and problems, we hope to suggest areas ripe for research.

The themes of succeeding IAAI conferences have been fairly constant: Show the maturity of AI as a commercially viable set of technologies; in particular, we tended to emphasize the role of AI technologies as a “raisin in the loaf of bread.” Any successful application will employ many information technologies; the AI component may be only a small component of the overall development cost, but it’s the sweet nugget in the loaf that makes the bread taste great.

The reason I’ve gone through this review of

perspectives is that it shows a progression in what we saw as the dominating issues facing AI. In the first few years, the point was to show that AI was a viable technology, to counteract the feeling that there was an impending AI winter (which I personally think never happened). In the next few years, we tried to say that although one could actually make the AI component of these applications deliver value, the hard problem was to make it work in context. All the big problems were integration problems. That was largely a result of changes in the computing base, the dominant programming language changed, and so on. Such issues, when viewed from a 20-year perspective, are of absolutely no consequence. The dominant programming language seems to change every five years (and not always for the better), and you’ve just got to accommodate to that. Once those issues seemed to be dealt with adequately, we then changed our focus to showing that the application of AI technology had become as routine a practice as many other areas of information technology.

I think it’s time that we should declare victory: AI applications are a well-established, perfectly viable technology in practice in industry today. The reason there was a perceived AI winter is that the companies selling the tools went broke. I know. I was there (at Symbolics). Mark Fox used to make an interesting point about this: He said that if you looked around today for

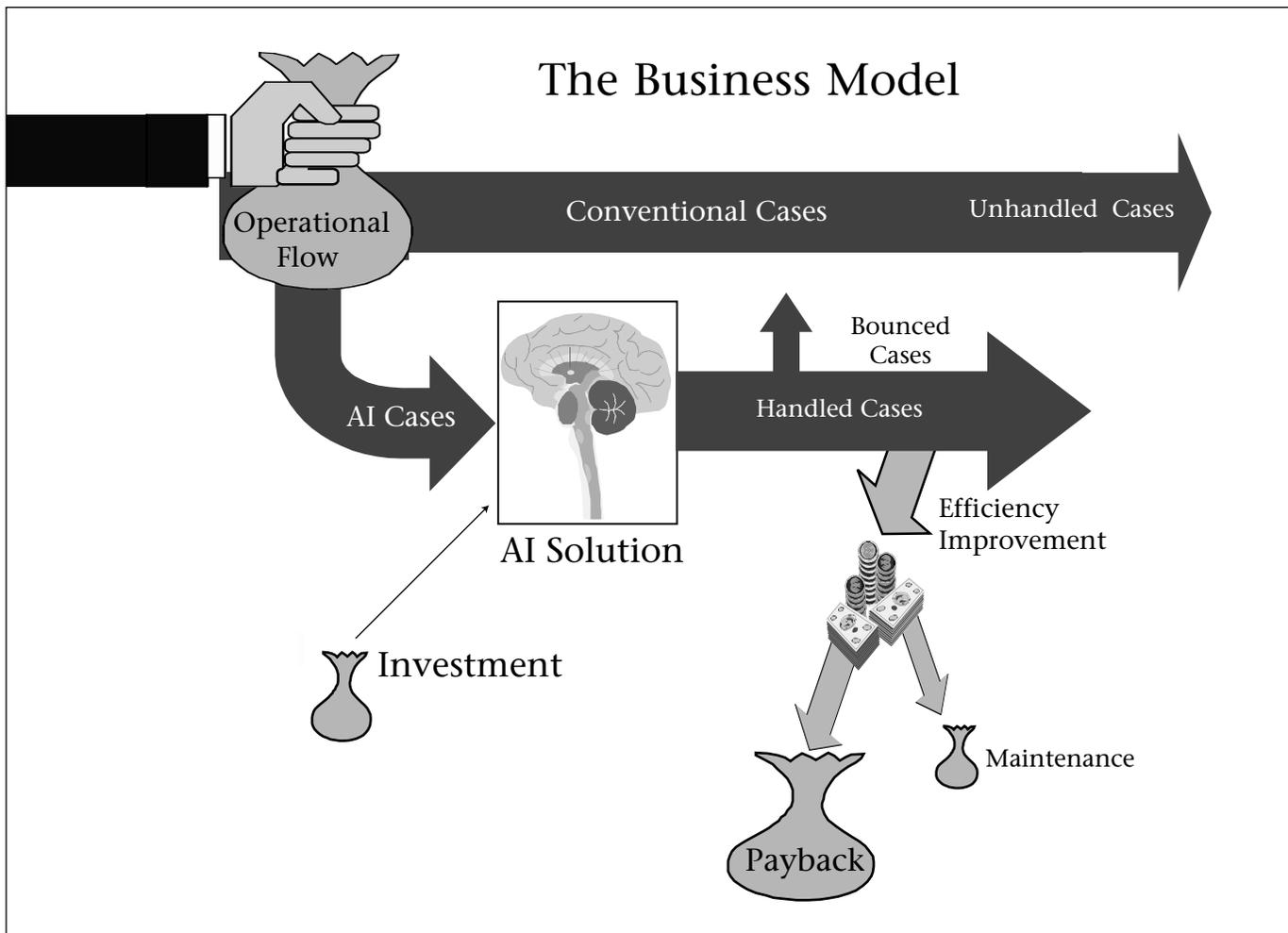


Figure 4. The Business Model for Traditional AI Applications.

a company called "Operations Research, Inc.," you wouldn't find it. That is not because operations research doesn't work; it's because it works too well. Every company has an OR specialist, and every consulting practice has a department full of people who are experts in it. In fact what has happened to AI tools and technology is precisely the same. If you go into large corporations you will find there are several expert systems groups dispersed throughout the company; they are populated mainly by people who are not research computer scientists but experts in an application domain. They are typically building a series of modest-sized expert systems that solve real problems in the corporation. This is considered a routine, although perhaps a slightly adventurous, practice.

If we look at the paradigm behind all these applications, what you will see is something that looks like figure 3: Inside there is a big AI program that does the decision making. A problem arising from the business flow of the

organization is fed into the program as a query and out comes an answer. By and large, these programs are large knowledge-based systems (compared to our research aspirations, the knowledge base may be modest sized, but the KB is certainly not trivial compared to what had been done in the early days of knowledge-based systems research).

These systems operate in a query-processing framework. That is, a query arising from the business flow is formulated and sent to the knowledge-based system in a relatively formal representation. The query may be formulated by an interactive graphic user interface (GUI), or it may come directly out of the organization's enterprise system. Once the system produces its answer, it feeds it back to the source of the query in a formal representation.

The business model now looks like figure 4: The company has an operational flow worth a lot of money. Thus, if you take the case of the AMEX AUTHORIZER'S ASSISTANT, this flow contains

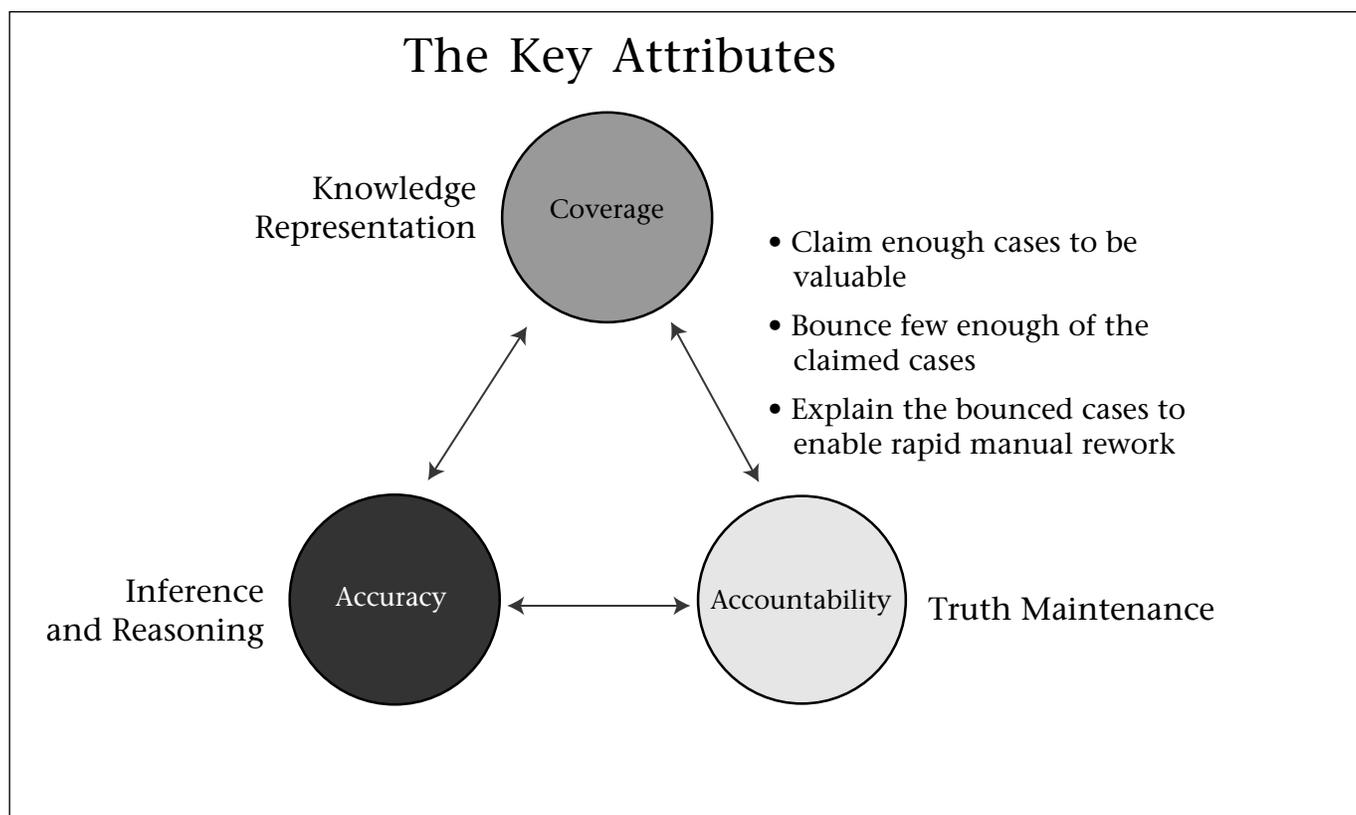


Figure 5. The Key Attributes of Traditional AI Applications.

perhaps billions of transactions per year—I don't know the exact number, but it is a very large number indeed, and it is worth a lot of money. A lot of the transactions never get to the AI system because they are too simple and are handled by more routine technology, but a significant percentage are passed to the AI system. Of course, the system can't handle some of these. However, it can handle most of them. The organization reaps an enormous efficiency improvement on these transactions. There is a monetary value to this efficiency improvement; part of this is used to pay for maintaining the application; the rest shows up as return on the investment. If this return is much bigger than original investment, you declare victory and write a paper for IAAI.

In the typical example, there are many billions of dollars of business flow, and the efficiency improvement is small, but a few percent of efficiency improvement still produces a lot of money compared to the investment. We have seen that this process can pay off in significant monetary terms for an organization, but the monetary assessment sometimes trivializes the benefit. Sometimes it is the ability to do the task at all that is significant.

In this model, there are three attributes that

are key (figure 5): (1) coverage, because if you can't handle enough of the cases you don't get the payoff; (2) accuracy of the coverage, because it is expensive to bounce a lot of the cases; and (3) accountability, because you have to be able to explain what went wrong when you incorrectly process a query. In many cases, you also need to be able to explain your reasoning even when you successfully handle the problem. You often have to justify a decision you have taken and preserve it for future reference; this is particularly true for systems that operate in a design context.

These key attributes lead to an AI research agenda: We focus on inferencing and reasoning for accuracy, on representation for coverage, and on dependency management and truth maintenance for justifiability. There is a large set of perfectly viable and important AI techniques to support these three attributes. I recently spent three years as a program manager at the Defense Advanced Research Projects Agency (DARPA), and when I looked at this agenda I said, "Well that actually lines up pretty well with the kinds of research we sponsored about five years ago in the government, at least out of DARPA." Thus, these capabilities have arisen from a long-term focus on knowledge

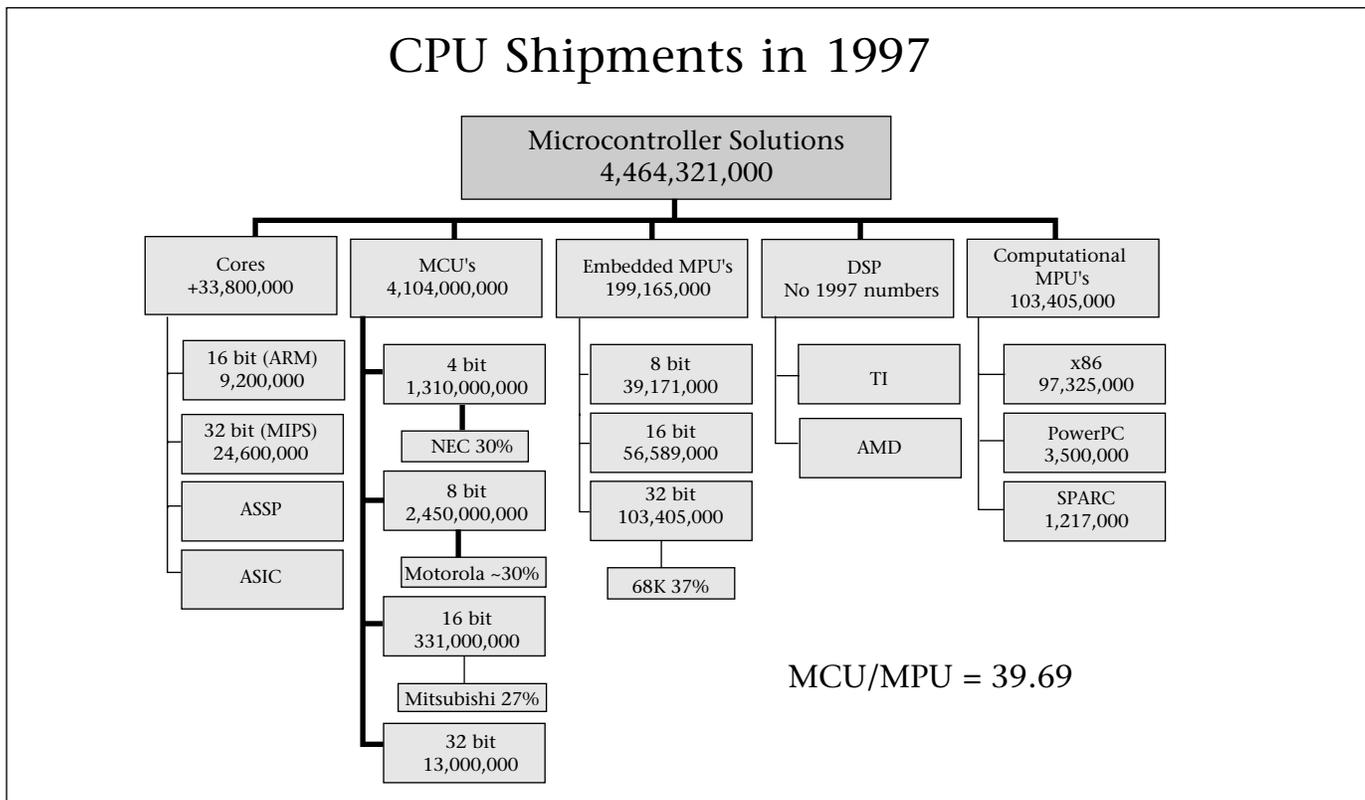


Figure 6. CPU Shipments in 1997.

Source: Data Quest plus additional information.

representation (there is even a DARPA program named High-Performance Knowledge Bases), inference, and planning by the government in the form of sustained research funding.

This is where we are today, and it has worked exceedingly well. Even if there was an AI winter, we should declare that it is over (but, as I said, personally, I think it never happened).

What does the future hold? Let's begin by looking at the underlying base of computational elements that the VLSI (very large-scale integrated) chip industry is delivering. I believe there is a sea change afoot. I have to thank my friends at DARPA's ITO office for the data on these slides; it is part of their pitch and I stole it from them (figure 6). The chart shows the number of microprocessors sold into desktop and server applications, that is, conventional uses of processors. The data are from 1997; at that time, there were about 100 million of these conventional microprocessors sold in a year. Also shown are the number of microcontrollers, processors that are used in embedded applications, a very broad spectrum of uses. There were 1.3 billion 4-bit microcontrollers compared to the 103 million microprocessors and 2.4 billion 8-bit microcontrollers. The numbers are enormous even when you look at

the higher end of this family, that is, at the components that we would recognize as computers, such as the 16-bit microcontrollers. Remember that when the first PC came out, it was a 16-bit micro. There were 331 million of those sold in 1997 compared to the approximately 100 million processors in desktop applications. Even in the 32-bit category of microcontrollers, there were 13 million sold in 1997. In aggregate, the microcontroller market is huge. I was talking to Samy last night and he said a low-end car from GM (which is where he works) has 16 computers in it today; that number will go up. A high-end car has 36; so, if you ask, "How many processors do you own?" the answer is that the ones you think of as computers are a statistically irrelevant sampling. The ones that really add up are in your cars and other places, even today.

Now this pie chart (figure 7) makes that clear. The huge wedge is microcontrollers, and the tiny little slice is processors used in computers as we think of them. Here are the projections for the year 2000 (figure 8): Desktop processors will grow to 150 million, which is a healthy growth rate of 1.5, but if you look at the number of microcontrollers, it is predicted to grow to 7.2 billion, a growth rate of 1.75; it's

## 1997 Summary MCU + MPU

	4 Bit	8 Bit	16 Bit	32 Bit	TOTAL 1997
Computational MPUs	N/A	N/A	N/A	103,156,000	103,156,000
Embedded MPUs	N/A	39,171,000	56,589,000	103,405,000	199,165,000
MCUs	1,310,000,000	2,450,000,000	331,000,000	13,000,000	4,104,000,000
Cores	Unknown	Unknown	10,000,000	48,000,000	58,000,000
DSP	N/A	N/A	N/A	Unknown	Unknown
TOTAL	1,310,000,000	2,489,171,000	397,589,000	267,561,000	4,464,321,000

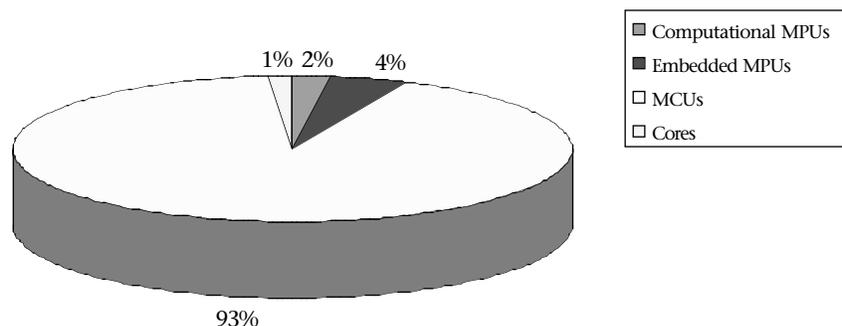


Figure 7. Controllers' and Processor's Share of the Market.

*Summary:* Even in 1997, computational microprocessors are a "statistical fluke." Almost all computation is embedded. (Source: Data Quest.)

growing somewhat more rapidly. However, most significant of all is the growth rate of high-end microcontrollers. The 16-bit ones are projected to grow by a factor of 2.3 during that period, and the 32-bit microcontrollers are projected to grow by a factor of 3 during that time. The crossover point between the microprocessor category and the high-end embedded microcontroller category is in the near future.

What that says is that most computation will be embedded in physical systems that interact with the world. The emerging paradigm is: Get physical; give our brain in the box eyes, ears, and hands (metaphorically of course), so that it can perceive and interact and effect the environment.

Here are a few examples of existing systems that are in this category. One of these is actually a deployed application, the IMAGE-GUIDED NEUROSURGERY SYSTEM that Eric Grimson will talk about Wednesday at 2:30. This is now in routine practice at Brigham and Women's Hospital. I am pretty sure he has videos of this that he is going to show. Eric gave a presentation on this work a few years ago at IAAI; it was an invited talk, and at that time, it was an emerging application. It was about to go into experimental practice, but it is now routine.

This system uses cameras and CT scans to form three-dimensional models of a brain that are used in tumor removal surgery. During

surgery, the camera is used to observe the precise orientation and position of the patient's head, and the 3-D image of the brain is superimposed onto the real one. This is used by the surgeons to help guide their surgical probes as they dig through the brain to get to the tumor that needs to be removed. This allows them to perform surgery that was previously considered too risky because the surgical path would have come too close to vital brain tissues. Thus, this is a perceptually enabled application that is already in routine, although certainly not widespread, practice.

Another application is the INTELLIGENT ROOM project at the Massachusetts Institute of Technology. I should apologize for my MIT-centric focus, but I am from MIT and it is traditional that we are myopic, and I wouldn't want to disappoint any of you by breaking with tradition. The INTELLIGENT ROOM is a smart space; there are now probably 15 or 20 similar projects going on around the country. This was one of the first, and it is still in many ways one of the most mature ones. In this system, we combined speech recognition and natural language understanding, vision and gesture recognition, and information management and retrieval into a coherent system that you can interact with in ways that are different from those of conventional computation. I have a brief video that shows some of the interactions with this.

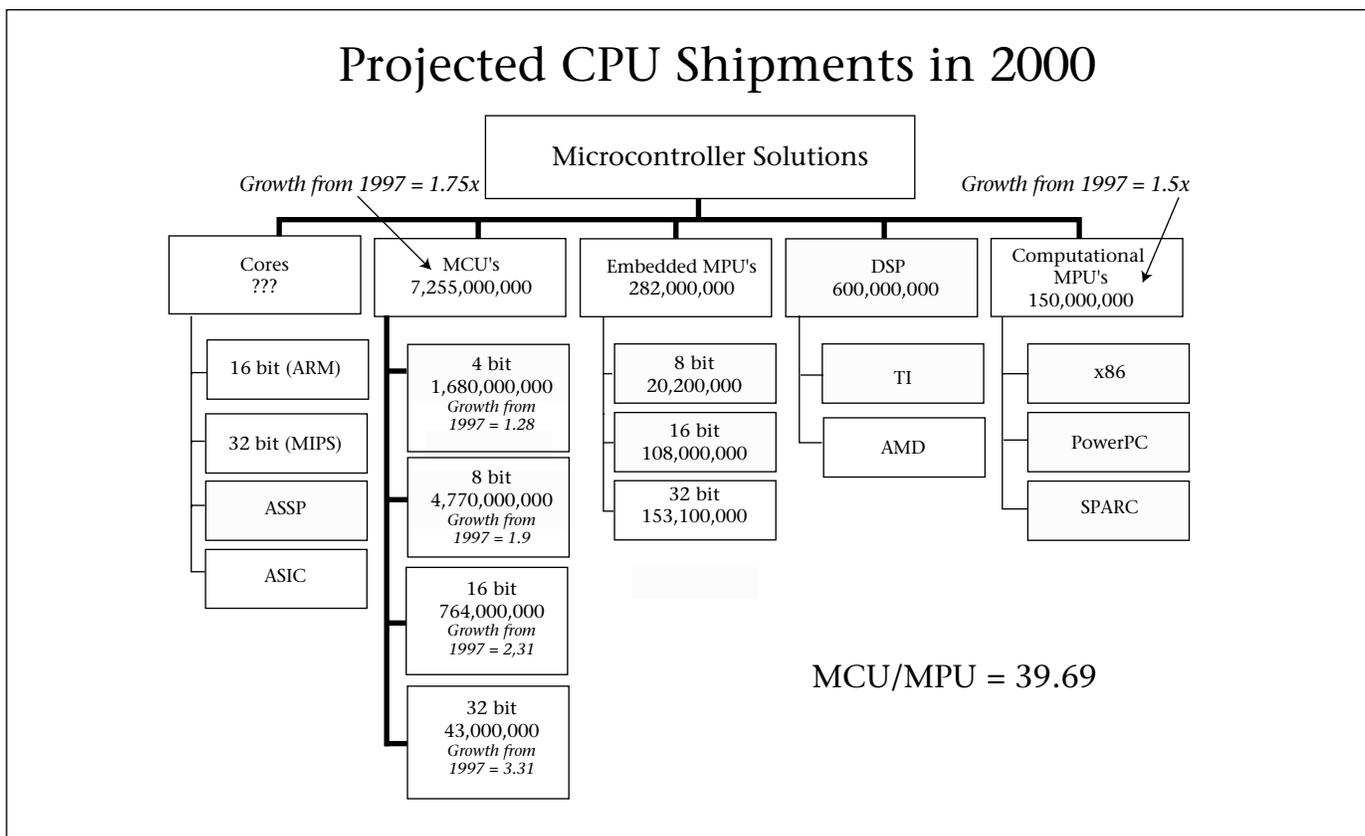


Figure 8. Processor Shipments Projected for 2000.

Source: Data Quest plus additional information.

I should mention that this was made for last year's AAAI conference; the system is significantly better now.

In the video, there is a point where Michael Coen lies down on the couch in his office (which is a very atypical MIT office). There is a camera mounted above the couch; when he lay down, the camera could see that he had changed body posture and decided he must be going to sleep because his posture was now prone. The system then closed the window blinds, killed the room lights, asked him when he wanted to wake up, and put on Mozart. All these appliances are under computer control; they are the "hands" of the INTELLIGENT ROOM. The only bug in all of that is I would not personally have gone to sleep to that particular Mozart selection. These tests are done in real time; in fact, it works a little better this year both because of better software and because Moore's law has worked one more time.

This is a system that you interact with by talking to it, gesturing, drawing, and sketching, and it responds with images and voice. There is no keyboard in evidence except in the rare occasions where typing happens to be the best way to interact with the system. By and large,

there is no gadgetry; there are no badges you wear to identify yourself. In the video, Coen is wearing a microphone, but we intend to replace that with a microphone array at some point so that the environment will be totally gadget free.

The video showed snippets of two different kinds of applications. One, which one day will be in your home, helps to manage your daily life: "Oh, you are about to go to sleep; when do you want to wake up?" The other is sort of "a command post" (in the very generic sense) that focuses on information management and retrieval. These are examples of what you can do in a system enabled by perception and effectors.

I'd like to share my vision of what one of these environments would be like 10 years from now when deployed in a work environment. I'd expect that we would hold our meetings in these perceptually enabled spaces and do the things we normally do at meetings: We might consider some issue, somebody might make an argument for the way we ought to approach that issue, somebody else might make a counterargument. I might offer to send him a market forecast that would be relevant to the discussion. At any point in this process, the

system might present relevant information of its own.

There are a family of representations that are used for capturing these kinds of deliberations (going back to a system called GIBIS), and there have been many attempts to make systems that actually use these representations to capture design rationale and design deliberation. The problem has always been that you have to enter this information manually, but that's very tedious, so no one ever does it, and the information always gets lost. Even information as simple as "What did I promise to do in yesterday's meeting?" often gets lost, particularly for those of us whose brains are starting to deteriorate. Thus, I often walk out of a meeting with a pad saying "Call Randy," "Call Samy." I look at it the next day and say, "Samy who?" and "Why?" Even if I could remember it was about issue X, what I'd really like to know is what was the context around that issue?

My vision is that we will interact with one another inside perceptually enabled spaces that listen to the discussion and that extract the essence of what's going on. It doesn't need to be perfect; it doesn't need to understand everything. In point of fact, neither do we, but what you would like to be able to retrieve is the contextual stuff about who interacted with whom and for what purpose. I might come in to work the next day and ask, "What did I promise to do yesterday?" and get the answer, "You promised to ship Jane some forecasts this week." If I asked why, the system might tell me, "Because they supported Tom's claim that he would have enough cycles to use a straightforward implementation." I could ask, "Was anybody arguing against that?" and be told that indeed Sam did.

All this is perhaps a forward pointer to Reid Smith's talk on corporate knowledge management. I don't know if this is Reid's version of the future, but it is mine. I think that the trick of capturing corporate knowledge over the long term is to capture it in context through natural interactions with the environment. Let the rooms, the environments become natural players in the scene. It would be nice if this happened around the coffee machine as well as in the meeting room, because after all, half of the good ideas in the world happen around the coffee pot, not in the meeting room. The meeting rooms are the rites of passage where you ratify decisions; they are very important places because they are more formal and, therefore, easier places to capture information. However, they are by no means the only ones. I have lots of good ideas in rather inconvenient places like the shower and my bed. I'd like to be able to

capture those ideas before they disappear, so these capabilities must be ubiquitous, and our entire set of ideas about where the work place is located will need to change. If I have a good idea in the middle of the night, I'd like to explain it to my electronic personal assistant; by the time I arrive at work, I'd like the system to have worked on the problem by finding me papers I should read or people I need to talk to.

I think 10 years from now we might be able to do some of this. It is not going to be easy: There are obviously issues of dialog tracking here that are nontrivial, but people have worked on them a long time. There are language-understanding issues that are nontrivial, but people have worked on those for a long time too. Then there are representation and reasoning issues and all the rest of it, but all of those have also been worked on for a long time. I think the time is getting ripe to attempt to do this because as I showed you before, we can build environments that are good enough today to support some limited kinds of interactions of exactly this kind. It will be harder when there are many people in the room because you have to keep track of who is talking and to whom and why. There are all kinds of issues, like how do you know that John was responding to Sam and not Jane, but in an environment with lots of perceptual sensors, there are lots of clues: who were you looking at, who spoke last, and so on. The task of engineering systems that can use such clues well is a very difficult but very exciting challenge.

One of the technologies that was used in the system illustrates that natural language technology is getting good enough to support the kind of information management tasks I've just described. Given fairly unrestricted queries about a domain the system happens to know about, the system can produce information for you, direct you to web pages that answer your question, and so on. This is one of many very good natural language systems that are currently around. This one again is an MIT one, done by my colleague Boris Katz. It is going to be shown in the exhibit hall as part of a larger demonstration, but you can also get to this system on the MIT AI Lab web page at any time. Incidentally, there is a very interesting story about what happened to this system when we exposed it to the public on the web. Boris had probably been working on this system for 15 years or so when John Mallery and I came to him and said, "Oh, we're going to spend about 1 days effort and put your system on the web for you. Won't that be nice? Thousands of people will come and ask it questions that you can't control." You can imagine that he was

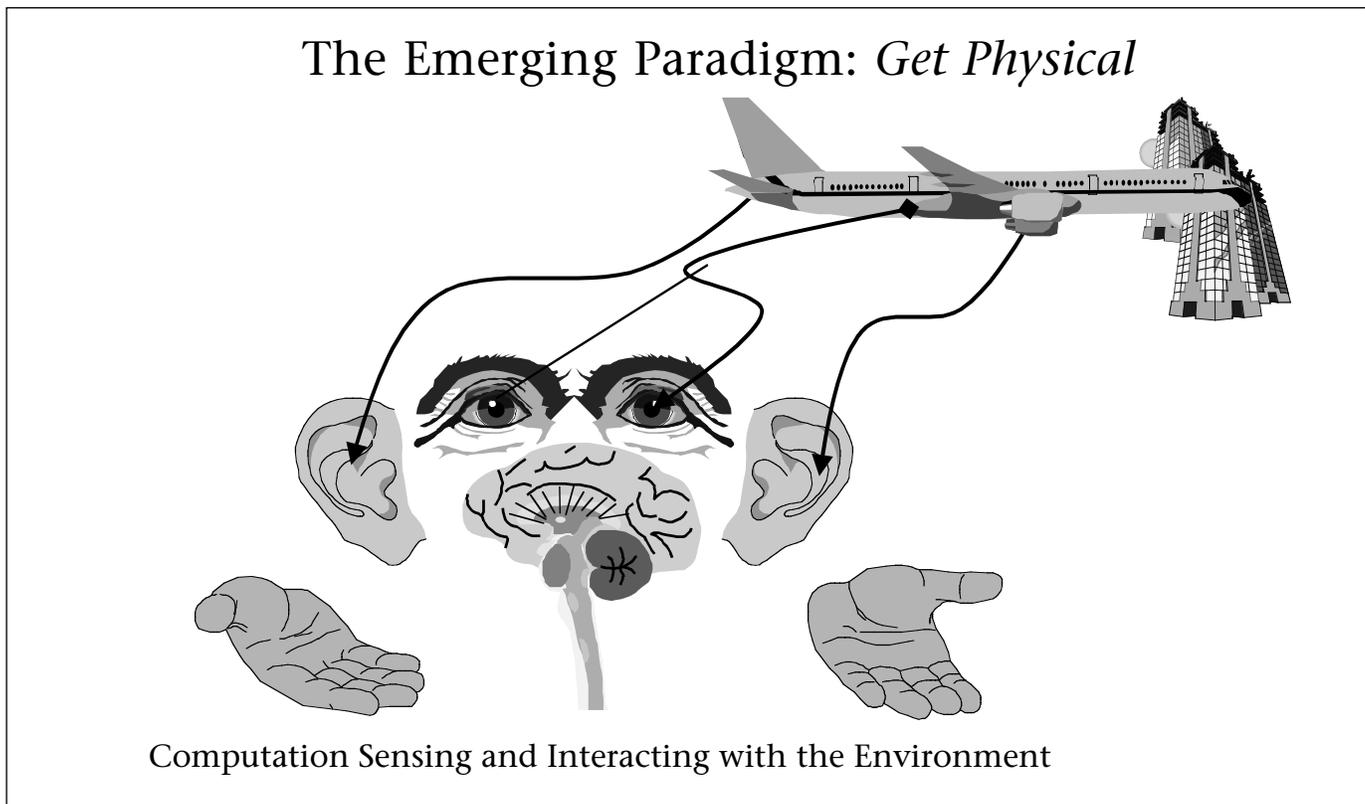


Figure 9. *The Emerging Paradigm: Embedded Intelligent Systems.*

not enthusiastic about that, but in point of fact, this system has gotten better in the last 5 years, much, much faster than it did during the first 15, because now real people, not linguists, ask it questions. We automatically journal all of those questions and go back and figure out why the system failed and then improve it. This is a very limited example of being embedded in the real world, but it has made an enormous difference.

This suggests turning to the second aspect of what is changing in the underlying computational base. Fortuitously, as I was working on this talk, *The Boston Globe* published just the right chart for me. What it shows is the supply of communication bandwidth versus current demand, extrapolated for a few years. Roughly speaking, what this says is that bandwidth is free. There is a lot of it, and there is going to be a lot more. This is the second big surprise. The first was to realize that we are entering an age of abundance in embedded computation. It was a little bit of a surprise to me to realize that the processors that we tend to think of as computers are a statistically insignificant sample of the population of processors and to realize that the embedded processors are becoming quite powerful. However, that surprise is nothing

compared to the surprise of realizing that bandwidth is free. In point of fact, bandwidth is growing so much more rapidly than Moore's law that it is probably going to be the driving factor. There are some simple reasons for this. Some of them are technological things like wave division multiplexing in optical fibers, but the other one is relatively mundane. When you lay down fibers, you dig holes in the ground; it turns out that digging the holes is a lot more expensive than the fiber, so as long as you've opened a hole in the ground, you may as well put in as many optical fibers as you can. Therefore any time somebody goes out to create new bandwidth, they create an awful lot of it. The same thing is true for satellites. If you are going to the bother of shooting something into orbit, you probably want to make sure it can handle a lot of channels because it is really expensive to get it up there; so every increment of bandwidth is enormous.

Whether demand catches up to supply or not is actually not the important debate. The real point is that there is enormous supply to meet any new demand. We've entered an era of abundance for communications as well as for embedded computation. This will enable systems like the INTELLIGENT ROOM to be dispersed in

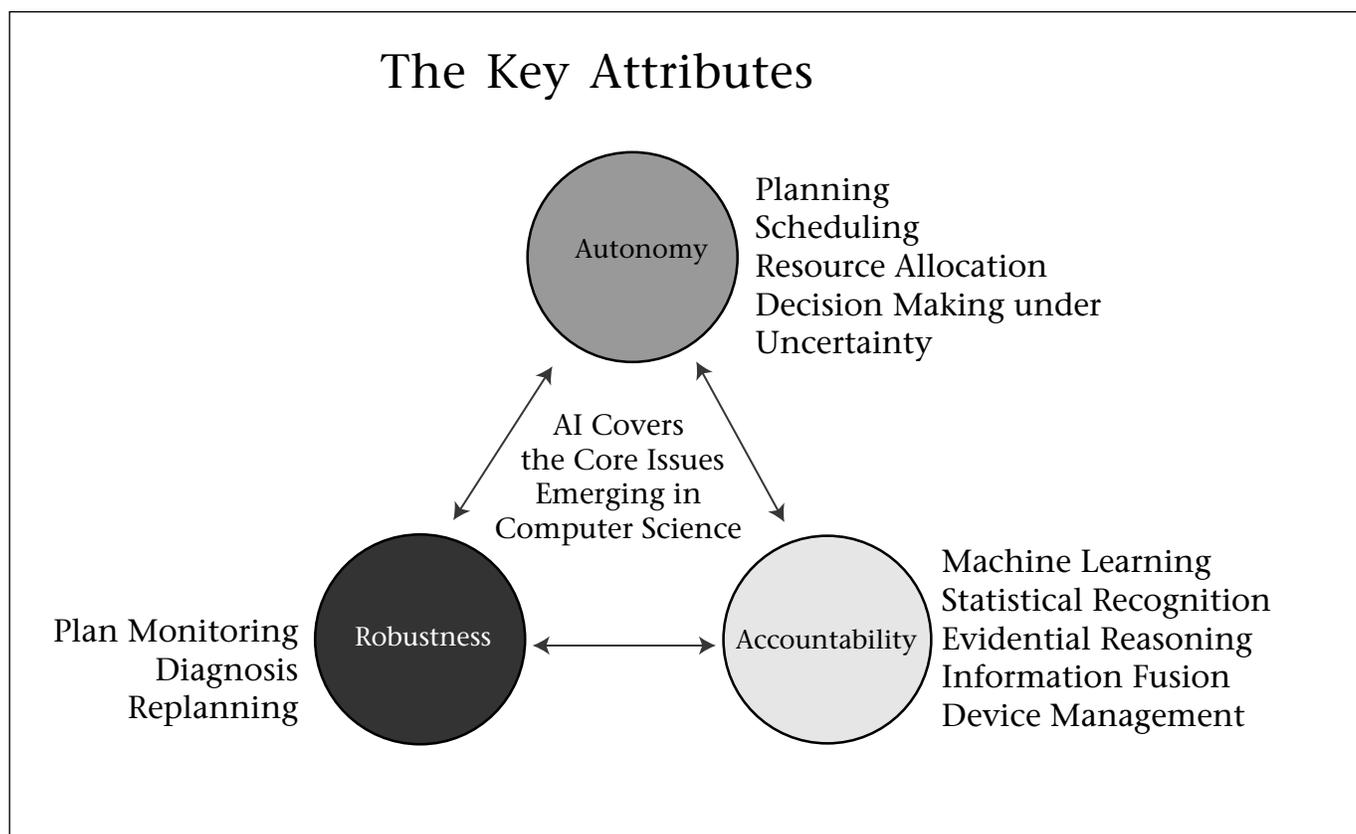


Figure 10. Key Attributes of the Emerging Paradigm.

space. If I want to have a teleconference with a colleague across the world, the bandwidth will be there to do that. What has actually been holding it up more than anything is that the underlying applications that provide a teleconference are terrible. You have to sit in front of a camera and become a talking head. If you decide you want to get up and point at something, the camera won't follow you. It's just terrible. But now we've got the bandwidth to carry a lot more, and we have the smarts to build systems that can move cameras to follow you and the language-understanding capabilities to make sense of what you've said. We can use all of this to capture the flow of information.

What this suggests is that the common mode of computation will soon consist of a highly distributed ensemble of embedded communications processors with effectors and sensors (figure 9). The last jump in the communications network may well be wireless to allow mobility, but the wireless infrastructure is getting better very quickly as well.

If this is indeed the emerging computational world, it is a very different one from that which we are used to. Sensors and effectors will not typically be regularly placed. You will put them where you can and that is not usually on a

nically spaced grid. Even if it were, they fail. Failure is an unavoidable fact of this world. The more stuff you have, the more opportunity there is for some of it to break. This is particularly true when communications is part of the process; it breaks faster than processors do. There are lots of sensors, but they are cheap and imperfect. They are imperfect not only in the sense that they break, but they are also imperfect in the sense that they are noisy. Any time you attempt to sense something, noise is in the way. The challenges then are to form accurate perceptions of the world using an ensemble of distributed, fallible components; to recover from failures of the components; to compensate for noise whether it is from the sensors or the environment itself; and to use all this infrastructure that is growing rapidly to best advantage.

Earlier I talked about the three attributes of accuracy, coverage, and accountability and how they were key to the traditional AI agenda. Those attributes and the research agenda they inspired will not go away, but there is a new triad of attributes that is more relevant to the emerging world of abundant, embedded computing that I've just been describing. These are (1) autonomy, because the systems are out

there in the environment and they have to plan for themselves; (2) robustness, in the sense of being able to recover from failure; and (3) coherence, in the sense of being able to form a coherent picture from lots of partial, noisy sensing of the world (figure 10).

These attributes also lead to a research agenda that is very exciting. Although there is certainly overlap with the previous agenda, there is a very significant shift of emphasis. The goal of autonomy leads a focus on real-time planning, scheduling, and resource allocation. Decision making will be in an environment that requires reasoning under uncertainty because absolutely nothing is noise free. The goal of robustness means that there will need to be a huge focus on monitoring whatever plans we have made, diagnosing what went wrong, and recovering and replanning. The goal of coherence leads to a focus on machine learning, statistical recognition techniques, evidential reasoning, information fusion, and device management: deciding which devices I should use, where I should put them, and so on.

Like before (and conveniently enough), this tends to map onto a government-sponsored research agenda that is going on right now. I see Jim Hendler who is currently a DARPA program manager sitting in the audience. He could spell out a road map for how this agenda maps onto current and planned government programs.

I want to spend a little bit of time discussing how we might attack these issues and what results we already have to draw on. The goal of autonomy means that these systems will need to be able to plan and schedule their own actions. They will have to do that within real-time demands; this, in turn, raises the question of how good a plan they can develop given that they have to develop it in time to be useful. A perfect plan that isn't available until after the need is gone is not very useful. In military applications, this issue comes up all the time because a perfect plan to get a helicopter into an observation site 10 minutes after the battle is over is not a very useful plan. Thus, planning has to be situated in a context of time and computational constraints, and optimality is no longer the key issue; the goal is to get a good enough answer within the time constraints. Of course, people in the planning community have been looking at these issues for at least the last five years. We don't have perfect answers, but we do have good enough techniques to build some real systems. These systems will also need to dynamically allocate resources; whatever set of resources they started with will not necessarily be the set of

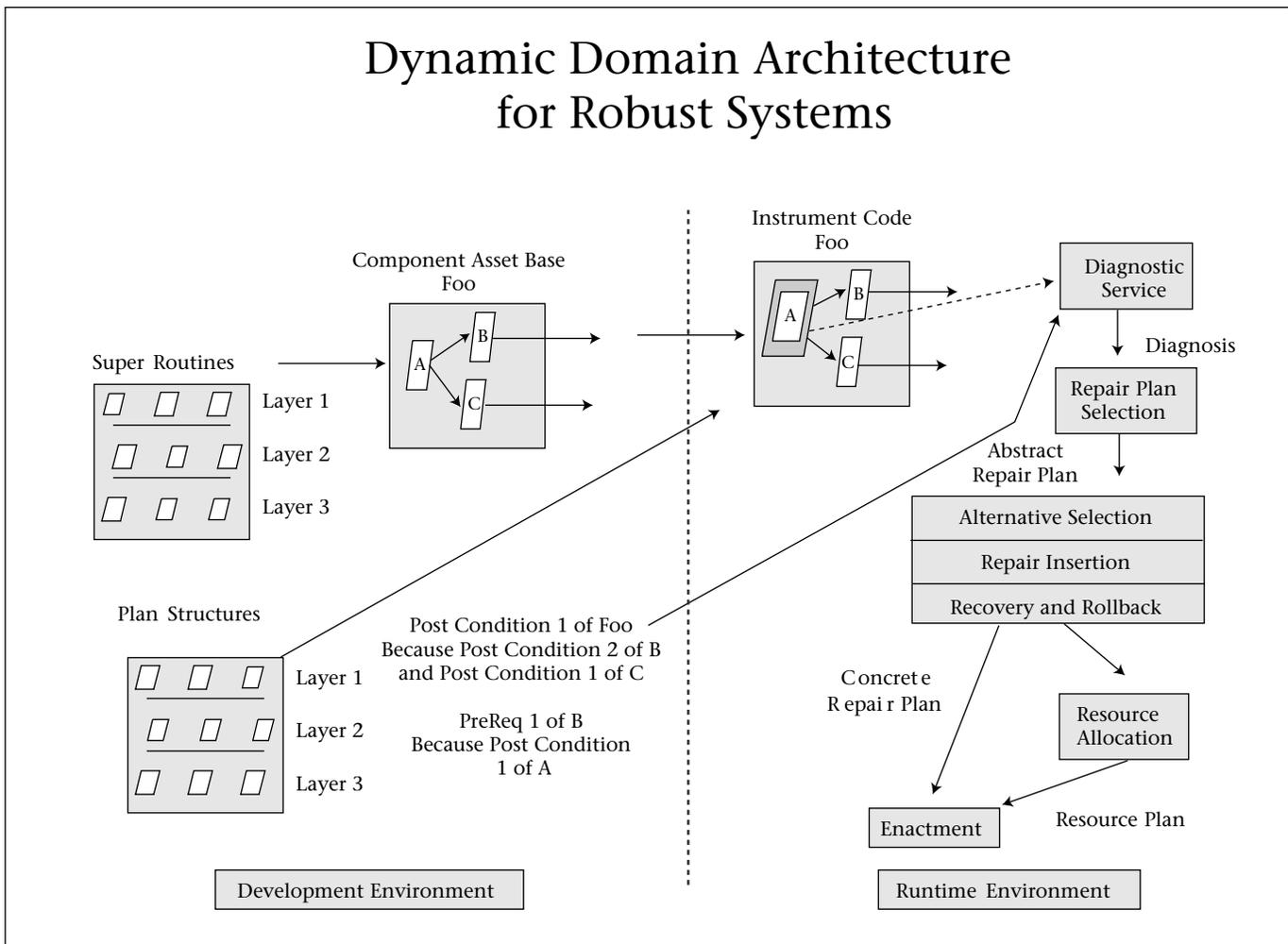
resources they have to work with when a problem arises because things break, and resources get deployed to other tasks. Systems will have to function when there are many tasks executing concurrently and contending for the use of the common resources.

We have some spectacular examples of autonomous embedded systems like the REMOTE AGENT system on Deep Space 1 that have already been deployed. Some of these are a bit peculiar because real time tends to be a little slow on space missions, but it's nice to be able to do an autonomy experiment where real time is not measured in seconds but hours and weeks sometimes. The Deep Space 1 autonomy experiment was just flown this year, and it was declared a success. REMOTE AGENT successfully operated the Deep Space 1 probe for a day. Wednesday at 10:30 Ken Ford and Peter Norvig will talk about this.

The Mars Rover also had a significant autonomy component. It is interesting, though, that autonomous control of the rover was not enabled until the critical and secondary missions were already completed. These were done under manual command control from Earth, but I'm told that the rover worked much better once they let it run itself; that is a lesson that still needs to be learned by the operational community.

Now let's turn to the category of robustness. Robustness means a lot of different things to different people. I'm using the term to mean the ability to recover from failure of software and hardware components. We have done a lot of research into diagnosis and recovering from failure of hardware components. In fact, the NASA work on Deep Space 1 significantly focused on recovering from hardware failures, and it used a rather rich set of results on model-based diagnosis.

However, there is another problem faced by systems that sense the world and rely on their perceptions. It is not just the sensors that are weak; our theory of perception is also weak. We, as system designers, don't necessarily know the right operation to perform in order to get the kind of result we want. Therefore, we need to build software systems that are prepared to recover from their own failures. How can we do this? At least one way of thinking about this problem is to make the system contain representations of its own design rationale; the system should know how its components interact to achieve its overall goal. The plan representation that we used in the PROGRAMMER'S APPRENTICE project 25 years ago is not a bad starting point for this kind of representation. I personally haven't worked on program



representation for 20 years, but I'm starting to think about going back to that agenda. Having the system contain representations of its design plans enables it to monitor the execution of its programs and to engage in diagnostic repair and recovery whenever the system detects that its software has failed to achieve its goals. Again, I can point to the NASA talk on Wednesday as a good example of a system of this kind.

To make a brief plug for my own work, I'd like to suggest that an overall architecture for such systems is the next kind of software infrastructure that we want to create (figure 11). It is an obvious point that our progress is often represented in the formalization of new programming infrastructures: They change the way that people approach the design and the programming of systems. That is the reason why even this many years later, people still walk up to me and say, "I wish I still had my LISP machine"; the infrastructure embedded in

that system represented a better way of programming than what was common then and a much better way of programming than what is common now. Unfortunately, we seem to have progressed backwards for a while. The Lisp Machine system relieved the programmer of certain tasks that were very burdensome; these were primarily information-management tasks. By building solutions to these tasks into the infrastructure, the Lisp Machine system made programming a lot easier for everybody who used it.

Where robustness is a key attribute, one would want the programming environment to include many more things than the actual code. For example, we would want to represent how the program is structured into domain-specific layering of services, in which the super-routines—the services at the higher levels of abstraction—are actually the things that get reused a lot. As we pointed out in the PROGRAMMER'S APPRENTICE a long time ago, most programs

# Coherence

Many faulty sensors,  
randomly arranged in a noisy world.  
*Solution:* Use events in the world to  
mutually calibrate and to learn about one another

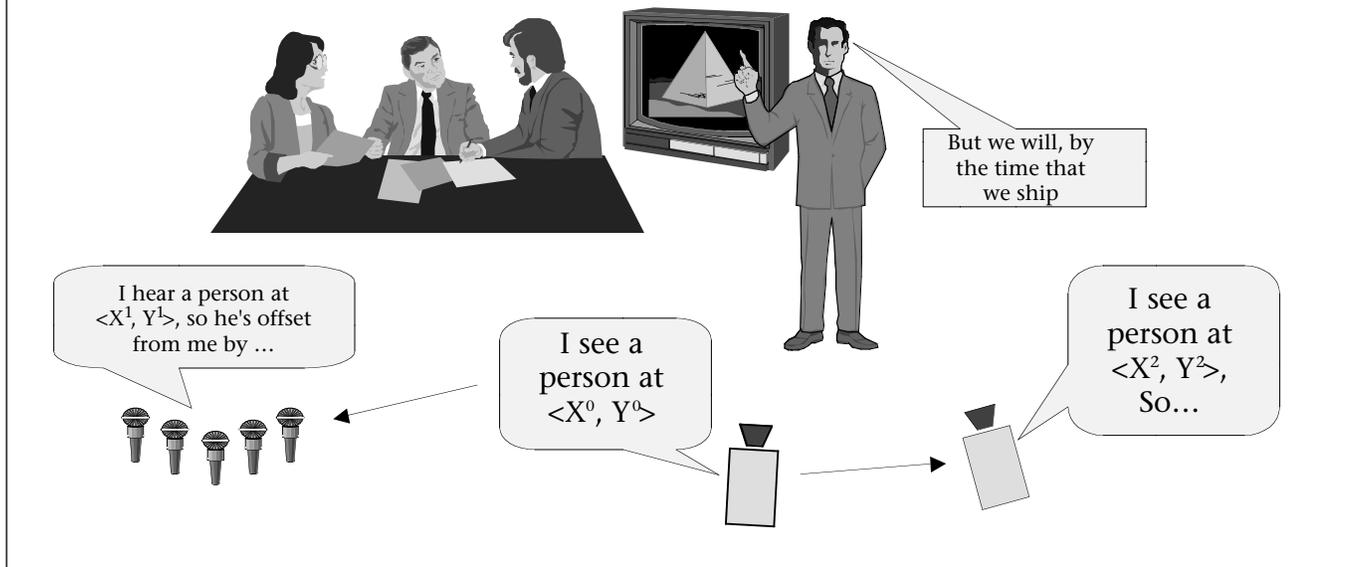


Figure 12. Mutual Calibration Leads to Coherence.

are actually combinations of very cliched patterns, and it is these common patterns that get reused. Each of these super-routines, or patterns, may be implemented in lots of ways. A common super-routine in image processing, for example, might involve a filtering operation. Now, there are loads of ways of doing any particular filter. They all achieve the same thing, but they do so in different ways, and some of them work better under certain conditions than others. The importance of focusing on patterns, or super-routines, is that associated with each super-routine is a plan that says why it works. You can carry this information into the run-time environment by synthesizing monitors around the components of the system to make sure that the invariants of the plan really are invariant; that is, you check at run time that they achieved what was intended. If they don't, then the monitors will signal conditions that cause diagnostic routines to be invoked. These might use model-based troubleshooting techniques to figure out where and how the program failed to achieve what was intended. The failure might be because we

have only weak theories of the domain, or it might be because an underlying component failed, or it might in certain contexts be because your system was hacked. All these are perfectly reasonable motivations for building systems this way.

This leads to a view of the software development environment that is a very information- and knowledge-rich system. The development environment captures not just code but also the rationale for the code, the plans, and so on. It performs a very rich set of services; it extends the system with code that we currently need to write by hand. That is, it inserts code that helps to detect exceptional conditions and to establish points from which recovery operations can be initiated. This leads to an agenda to develop facilities that currently don't exist in any software systems, but I think this is the right agenda; after all, this is the code that we all know we should write but never do. I speak personally, and I think everybody who is honest about it says, "Well error handling is the last thing I worry about." It has to be because until you

have got the functionality running, you can't worry about the exceptions. There a number of new run-time services that will be part of the infrastructure of this new generation of software; these include the run-time services to conduct diagnosis, repair, and recovery. This is a model of a new software environment that is knowledge rich on both the development and execution components.

The final issue to be considered is coherence. As I said, the world will consist of many, many somewhat faulty sensors; they will sometimes induce noise, and some random subset of them will break. The real world that we all function in ourselves day to day is an ugly place; it's one our systems will have to get used to. One way of approaching the issue of coherence is to make the observation that the world is a good representation for itself. Often, you can use events in the world to mutually calibrate the different sensors and effectors that are embedded in the world; if they observe the same events and share their perceptions of that event, they can learn a lot about one another and about the world. Thus, you can imagine a camera, by this I mean the camera and the computer processing its imagery, saying, "I see this guy at this position" and telling that to the microphone array, which says, "Hmm, I hear him at a different position, so I've just learned something about the calibration between his and my coordinate systems" (figure 12). Constant iterative calibration using mutually perceived events in the world may be a key technique to achieve coherent perception.

Here is another example of this idea. In figure 13 are four projectors ganged up into an overall display. They are not positioned very carefully, as you can see (figure 14); they overlap, their imagery does not quite line up, and their color balance is all off. However, looking at these four projectors is a camera; during a calibration pass, it figures out how a pixel in frame buffer space maps into a pixel in screen space. It's a "mere matter of computation" to invert that mapping and to figure out what pixels need to be written into the frame buffer to get a desired pixel on the screen. Similarly, it's just a "mere matter of computation" to compensate for a variety of distortions; so, we can go from incoherent projected imagery to a coherent one by sensing and computing. My colleague Tom Knight is working on this concept. There are a lot of interesting issues to be addressed. Once you say I can computationally correct for distortions in the projector, it becomes possible to use really cheap projectors. Thus, instead of using a projector like the one we use in this hall, which is a true tour-de-

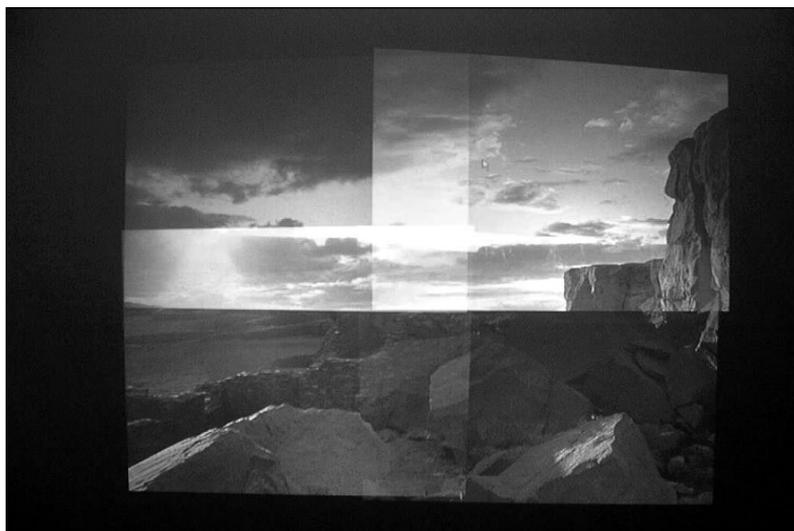


Figure 13. Uncorrected Overlapped Images.



Figure 14. Computational Correction Gives Coherent Image.

force of mechanical and optical engineering, you could buy a piece of junk and computationally compensate for all its problems. Unfortunately, you can't buy projectors that are pieces of junk. They cost about \$5000 a pop. We'd like them to cost about \$500.

I want to close by making an observation about the interaction between two kinds of activities in our field. The first is the kind of activity that the IAAI was established to promote: the development and practical application of new technology. The second is the pure scientific agenda of the AI community, which is to develop an understanding of machine intelligence. Of course, these are not completely unrelated exploits: Corresponding to the scientific goal of understanding machine intelli-

gence, there is an application agenda of engineering intelligent programs. For many of us, the goal of understanding machine intelligence involves the goal of understanding human intelligence because the kind of intelligent machine we'd like to build is one that resembles us. If this is our goal, then the experimental side of it might be the experimental modeling of human cognition in computer programs.

In the part of our work motivated by scientific goals, we tend to ask questions like, "How do people apprehend tasks so as to make them computationally tractable (for the kind of computing engine that sits in heads)?" "What knowledge and problem-solving methods and perceptual techniques, and so on, make all that work?" In the application domain, we ask similar questions: "How do we apprehend some specific task so as to make it computationally tractable for the kinds of machines that are built from VLSI chips?" "What knowledge- and problem-solving methods are appropriate?" There is an obvious interplay between those two exploits, but they are still different exploits. One of them has the goal of understanding intelligence; the other has the goal of making useful systems. These are not in conflict; they are complementary. We should continue to do both vigorously.

The IAAI conference has many goals: It serves to highlight the intimate relationship between research and application. It helps to evaluate the adequacy of our current technology, to focus attention on new enabling technologies. Finally, the conference helps to explore the frontiers of emerging application areas as I have been doing for the past half an hour or so. This may suggest new research issues, by highlighting areas where application technology is weak, and new insights are required.

However, the flow of ideas is not unilateral. Work on applications may also serve to create critical new scientific research insights. Sometimes the interaction consists of people in the practical domains saying "Hey, I have a task that I can't do. Guys go solve that problem." However, often what happens is just the opposite; a team of people trying to build an application say, "You know, the way I just made this work might actually tell us something about the way it works in people."

Here is a possible example of that. The INTELLIGENT ROOM was motivated by the practical goal of making a useful environment. To do that, we had to worry about the question of how to coordinate the cameras and microphones and how to get coherence out of an

amorphous ensemble of noisy sensors. It turned out that the easiest way to do that was to rely on machine-learning techniques; the software agents that control the different sensors and effectors learn how to calibrate against one another and how to usefully interpret one another's input.

Michael Coen began to develop the hypothesis that the human brain might do exactly the same thing. His suspicion is that individual functional parts of the human brain use machine-learning techniques to develop models of one another. In other words, he believes that the way his software agents work might be psychologically real. Not only do we use machine learning to learn about the world through our sensations, but the various parts of our brains initially view one another as foreigners and learn how to work together using much the same techniques as they used to learn how to perceive the world. That is one of the claims. I have no idea whether it is true or false, but it is certainly an intriguing one, and it has the ring of something that just has to be true—at least to me.

AI researchers who are pursuing the scientific goal of understanding human intelligence worry about psychologically realistic representations. Does this suggest how the mind, and in particular the human mind, might do the task? Just by coincidence, there was an article that appeared in *Science* magazine recently reporting experiments by a group of neuroscientists that show that visually centered coordinates are the coordinates used by at least a significant component of our motor system, in particular the motor cortex. That is a striking claim because if I'm trying to plan how to grab the microphone with my hand, the obvious coordinate systems are hand- (or arm joint-) coordinate systems. I am moving my hand after all, and what I need to do is figure out the configuration of my arm joints. However, the observation the scientists made is that you always move your hands under the guidance of your visual system. None of our motion is independent of other sensations. In primates, vision is the dominant perceptual modality; therefore, as the various brain components learned to work together, visual coordinates are pushed into other parts of our brains. It turns out that this is also true of our auditory systems; there are significant parts of the auditory cortex that use visually centered coordinate systems.

I want to make several observations about this. The first is that neuroscientists can now provide us with interesting results from the point of view of AI research; this is new and

important. Notice that what the neuroscientists reported was not that the firing rate of neuron  $x$  in region  $\gamma$  of the brain is 23 blips per millisecond, which is what they used to be able to tell us when I first joined the AI community. Today neuroscientists are providing functional descriptions of the representations used in different parts of the brain; that is a qualitatively different kind of information that should be enormously valuable to us. There is huge progress that has been made in neuroscience, and much more will be forthcoming. The language being used to communicate these results is a language that we who are interested in building cognitively motivated systems can understand.

It might be that a hypothesis that arose in the AI practice of building a smart environment was supported by evidence from neuroscience. That would be enormously exciting.

It would be very interesting if we could build a computational model of cognitive development that leads to the same results. Suppose we built a model of development that used machine learning to support the mutual calibration of motor and visual parts of the model. Would such a model wind up with the motor component working in visual coordinates? Such an experiment would be a wonderful interaction among a variety of communities. This, quite possibly, is where our field is headed. I point you, in particular, to Patrick Winston's keynote talk tomorrow that will explore this theme more.

Let me close by summarizing: Technology changes will offer us great opportunities for building complex systems that are embedded in their environment. We are entering an era of abundance, and we should begin to think that computation and communication in this domain is every bit as free as it has become at the desktop. The building of these systems will also highlight new attributes and new technical agendas that are crucial to realizing the attributes. Also, these systems will be much more like us humans than are the knowledge-based systems that have driven AI practice in the past; therefore, the building of these embedded intelligent systems will be a unifying force for our discipline, unifying application and science, AI and related research disciplines. This future holds challenges, opportunities, and excitement.

**Howard Shrobe** is associate director and a principal research scientist at the Massachusetts Institute of Technology (MIT) Artificial Intelligence Laboratory. From 1994 to 1997, he served as chief scientist in the Information Technology Office at the Defense Advanced Research Projects Agency. Among other

## Book Reviewers Wanted!

The following books were recently received by AAAI for possible review in *AI Magazine*. If you would like to be considered as a reviewer of any of these titles, please send a message (in which you briefly mention your background and qualifications) to Chandra (chandra@cis.ohio-state.edu)

Brands, Stefan A. 2000. *Rethinking Public Key Infrastructures and Digital Certificates*. Cambridge, Mass.: The MIT Press. (314 pages, ISBN 0-262-02491-8. \$35.00)

Dumitrescu, D.; Lazzarini, B. Jain, L.C.; Dumitrescu, A. 2000. *Evolutionary Computation*. Boca Raton, Fla.: CRC Press. (384 pages, ISBN 0-8493-0588-8)

Foder, Jerry, 2000. *The Mind Doesn't Work That Way: The Scope and Limits of Computational Psychology*. Cambridge, Mass.: The MIT Press. (126 pages, ISBN 0-262-06212-7 \$22.95)

Harel, David 2000. *Computers LTD.: What They Really Can't Do*. New York: Oxford University Press. (214 pages, ISBN 0-19-850555-8)

Katagiri, Shigeru 2000. *Handbook of Neural Networks for Speech Processing*. Boston, Mass.: Artech House, (522 pages, ISBN 0-89006-954-9, \$119.00)

Reggiani, A., editor. 2000. *Spatial Economic Science: New Frontiers in Theory and Methodology*. Berlin: Springer-Verlag (453 pages, ISBN 3-54067-4934, \$88.00.)

Solla, Sara A.; Leen, Todd K.; and Muller, Klaus-Robert, editors. 2000. *Advances in Neural Information Processing Systems 12*. Cambridge, Mass.: The MIT Press. (1080 pages, \$65.00).

Subrahmanian, V. S.; Bonatti, Piero; Dix, Jurgen; Eiter, Thomas; Kraus, Sarit; Ozcan, Fatma; Ross, Robert, editors. 2000. *Heterogeneous Agent Systems*. Cambridge, Mass.: The MIT Press. (580 pp.\$60.00).

Wrathall, Mark; and Malpas, Jeff, editors. 2000. *Heidegger, Authenticity, and Modernity: Essays in Honor of Hubert L. Dreyfus, Volume 1*. Cambridge, Mass.: The MIT Press. (413 pages, ISBN 0-262-73127-4, \$25.00.)

Wrathall, Mark; and Malpas, Jeff, editors. 2000. *Heidegger, Coping, and Cognitive Science: Essays in Honor of Hubert L. Dreyfus, Volume 2*. Cambridge, Mass.: The MIT Press. (406 pages, ISBN 0-262-73128-2. \$25.00)



activities, he helped to initiate the EDCS and Information Survivability programs. His work has spanned the areas of AI, software engineering, advanced computer languages, VLSI design, and computer architectures. Shrobe also served as chief technical officer at Symbolics Inc. In that capacity, he

played a significant role in the development of the Lisp Machine's GENERA programming environment, still regarded as the most powerful software development environment in existence. He received his B.S. (1968) from Yale College and his M.S. (1975) and Ph.D. (1978) from the Artificial Intelligence Laboratory at MIT, where he was a cofounder of the PROGRAMMER'S APPRENTICE project. In 1978, he joined the staff of the MIT Artificial Intelligence Lab as a principal research scientist and in that role was one of the main designers of the SCHEME-81 microprocessor (a Lisp interpreter on a chip) and the DPL/DAEDALUS integrated circuit design. Shrobe also cofounded the Hardware Troubleshooting Project at the MIT Artificial Intelligence Lab and has conducted research on designing and understanding mechanical devices. His e-mail address is hes@ai.mit.edu.