Academy of Sciences toxicology section, where he met Spencer. "We got talking," says Kurland, "he about his interest in lathyrism and me about my obsession with getting to the bottom of the Guam mystery." That chance meeting stimulated Spencer to attack the two problems—lathyrism and the Guam disease—with the same approach.

"The first step," which began in 1981, says Spencer, "was to define the neurology of human lathyrism and to produce a satisfactory primate model in which the action of BOAA could be examined." This took 4 years and served as a guide to an attack on the Guam disease, the result of which is reported in the current paper. "It is not a total animal model of the human disease," admits Spencer, "but it is very close. I'm not trying to say that BMAA is the cause of this disease. I am trying to reawaken an interest in cycads." Spencer believes that there are probably other agents in the seeds that work in combination with BMAA.

"I expect our search will lead to a class of environmental chemicals that act as triggers for neuronal death."

Spencer speculates that the three diseases that make up the neurological complex on Guam might be elicted by different "doses" of the cycad toxin. "A high level of intoxication leads to motoneuron disease, while Parkinson's and Alzheimer's develop after lower exposures."

So, perhaps the mystery of Guam is solved. But the story does not stop there. "There are wider lessons to be learned," says Spencer. For one thing, the fact that motoneuron disease, Parkinson's disease, and Alzheimer's-like dementia can each be triggered by the same neurotoxin implies that the three diseases might be linked at some fundamental level. Another is that just because a disease might occur at high frequency and affect individual families throughout generations does not necessarily mean it is genetically caused, as is often inferred. "This should influence our thinking about Alzheimer's disease, which has recently been linked with genetic causes," says Spencer.

But the key inference is the notion of early exposure to a neurotoxin whose effects are expressed clinically only many years later. For instance, many Guamanians who left the island at the age of 20 to live in the United States have developed the disease 30 years later: hence Spencer's term, slow toxin. The chemical assault on the brain, even if it is transient, is compounded by a steady loss of brain cells with advancing age. This is the core of the environmental-toxin model that Calne, Spencer and others have been developing for this group of neurological diseases.

For instance, Calne and William Langston, of the Institute for Medical Research, San Jose, suggested a little over 3 years ago that "in most cases of Parkinson's disease the cause may be an environmental factor, possibly toxic, superimposed on a background of slow, sustained neuronal loss due to advancing age." This suggestion was inspired by the discovery that a chemical that goes by the shorthand name MPTP causes parkinsonian-like symptoms in both humans and animals.

The notion is further strengthened by the results, soon to be published by Calne and his colleagues, of a survey of six families in which several members have Parkinson's disease. The patients often developed symptoms at more or less the same time, irrespective of their ages. "We construe this pattern of age separation within families as suggestive of an environmental rather than a genetic cause," they conclude.

Spreading the environmental hypothesis net yet wider, Calne and Spencer speculated at the end of last year that "Alzheimer's disease, Parkinson's disease, and motoneuron disease are due to environmental damage to specific regions of the central nervous system and that the damage remains subclinical for several decades but makes those affected especially prone to the consequences of age-related neuronal attrition."

Spencer's results on the Guam disease clearly support this position. "I'd be very disappointed if the link between toxin and motoneuron disease related only to the western Pacific form of the disease," adds Kurland. "Clinically, the motoneuron disease you see on Guam is identical to what you see in the United States. I'm optimistic that Spencer's results will set off a search for similar toxic agents to which people are exposed in the West." The culprits are not necessarily to be sought in food, says Spencer. "I expect our search will lead to a class of environmental chemicals that act as triggers for neuronal death. But at the moment we don't know what they are." ■

ROGER LEWIN

Artificial Intelligence Moves into Mainstream

For software developers, the most important result of this research may not be the AI programs at all, but the AI programming style

SITTING in one of the largest and busiest commercial exhibition booths at the 1987 annual meeting of the American Association for Artificial Intelligence (AAAI),* which was held recently in Seattle, and speaking in his capacity as a vice president of one of the most active corporate AI development groups in the country, Texas Instruments' W. Joe Watson made a very disconcerting statement: "Most of us think that AI per se will lose its identity within about 5 years."

Watson did not mean by this that the recent surge of interest in commercial AI applications is beginning to wane; if anything the AI industry is maturing and becoming better established. At this year's AAAI meeting, for example, Texas Instruments was sharing the exhibition hall with nearly 100 other vendors of AI software and hardware, up from 85 vendors last year. Some 5000 meeting participants were thronging the aisles and display booths. Half the companies in the hall seemed to be selling some kind of expert system software to run on ordinary personal computers. (Expert systems are programs that give expert-level advice in fields such as medical diagnosis or tax planning.) And there were so many high-powered, graphics-based workstations being offered for advanced AI development work that the exhibition floor looked like an upscale video arcade.

* The Sixth National Conference on Artificial Intelligence, Seattle, Washington, 13–17 July 1987.

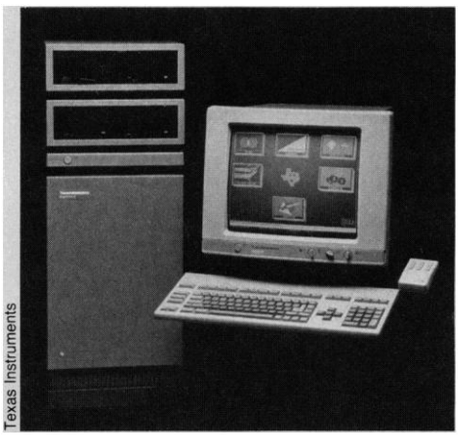
Nor was Watson suggesting that all the problems of machine intelligence have been solved. Quite the opposite. Over in the technical sessions the researchers were still grappling with profound difficulties in such areas as computer image understanding, natural language understanding, and the surprisingly intricate workings of everyday common sense.

Instead, Watson told *Science*, he meant that AI programming techniques are rapidly merging into the mainstream of computer science. "By the 1990's," he said, "the AI programming style will be standard, and the hardware support for that kind of programming will be standard." He recalled a recent talk given to the Texas Instruments AI group in Austin by a visiting computer sciences department chairman: "He said, 'We don't talk about AI in our lab. We talk about classical computer science versus modern computer science.' "

Watson was hardly alone in his assessment. *Science* heard variations on the same theme from a number of companies at the conference: from a software developer's point of view, the most important single idea coming out of AI research is not a program per se, but a new approach to programming in general.

A new approach is certainly needed. With conventional software engineering it simply costs too much and takes too long to get the programs written and debugged, especially when it comes to a massive project such as the space shuttle, or a new military weapons system. "In conventional software development the thrust is to get the algorithms in place to guarantee a correct answer," explained Watson's colleague William Petersen, marketing manager for knowledge engineering services at Texas Instruments. "So you spend a year trying to nail down the detailed specifications for the problem. You generate reams of paper. You get all kinds of systems engineers involved. And then by the time you get it done, the client has changed program managers three times and is giving you a whole new set of specifications." Indeed, those last-minute changes are one of the main reasons for the high cost and chronically late delivery of software: a seemingly simple alteration in the specifications can force a wholesale revision of the programming code, which in turn leads to a whole new round of debugging.

What makes the AI style of programming so different is not that the AI researchers write perfect code the first time; in fact they revise their programs constantly. The difference lies instead in their philosophy of what the software is supposed to do. From the beginning their goal has been to write programs that can cope with an uncertain world, in situations that cannot be predicted ahead of time. (In fact, that ability is one definition of intelligence.) So instead of telling the computer precisely what to do at every point, as in a conventional program, an AI researcher will typically tell it what to *know*. The computer then uses that knowledge to reason about what to do as new situations arise. Indeed, much of the advanced research in AI revolves around how one actually represents the knowledge in a computer. Expert systems, which are the most commercially profitable applications of AI to date, were an early spin-off of this work; knowledge obtained from human experts is represented as rules of the form, "If *this* is the situation, then do *that*."


Texas Instruments

An environment for programming. *AI-inspired programming tools, as implemented on high-powered, graphics-based workstations such as this one, could have a major impact on the way programming is done in general.*

In keeping with this commitment to flexibility in the programs themselves, AI researchers have pioneered a variety of techniques for flexibility in the creation and revision of those programs. One of the prime examples is called object-oriented programming, which is perhaps best known from its implementation in the Smalltalk programming language. Basically the idea is to abandon such familiar programming concepts as data, procedures, and subroutines, and instead build programs that function as a community of near-autonomous individuals—the "objects." Each object is actually a piece of software that behaves like an expert in its own particular domain. The number 3, for example, is a very simple object that knows how to add itself to another number, how to subtract itself from that number, and so on through all the basic arithmetic operations. A database is a more complex object that knows (among other things) how to take statistical information about itself and display it on the computer screen as a graph, as a bar chart, or even as a set of gauges with needles to represent various numbers.

More generally, objects can store information, process information, and create new information; they can create new objects and delete old ones; and they can send messages to one another, with object X requesting that such-and-such an action be taken by object Y. Even the overall program itself is an object, a kind of corporate executive that gets things done by organizing its subordinate objects into specialized teams and task forces.

The payoff of all this for the software developer is the ability to practice "rapid prototyping." By calling up objects from a predefined library and then linking them in appropriate ways, he or she can stitch together a rough, but functional program very quickly. "You sit down with the user and say, 'Tell me what you want,' " said Petersen. Usually, of course, the customer has only a vague idea of what he wants. But no matter. "Soon—like in a day—you can get *something* running," he said. "Then you bring the user back and say, 'Is this what you want?' " And so it goes. "You're constantly working on something, showing it to the user, and getting feedback," said Petersen. "Furthermore, if you need to change something it's not hard." The objects are deliberately set up so that the programmer only has to alter the way one object or set of objects responds to a given message; the other objects proceed as before. Thus, changes can easily be incorporated even late in the development process.

Obviously, as Watson pointed out, this kind of programming style goes well beyond AI per se. "[Texas Instruments] has already won several military software contracts this way," he said, "And in the commercial area we've done applications with this methodology that had no AI at all, but that were completed four to six times faster than with traditional techniques."

Of course, rapid prototyping is no panacea. For one thing, AI-style programming methods—and especially object-oriented programming—demand the resources of high-performance, high-priced workstations. Moreover, programming is still a lot of work. And the testing and verification process is still as critical as ever. "One of our greatest tasks is management of customers' expectations," laughed Watson. "When you can give them a prototype in a week, they start expecting the full job in 3 weeks." Nonetheless, the company's experience to date has made him a believer. "The vision is that modern computer science will absorb all this, and that in time, what will then be seen as 'conventional' personal computers will have these capabilities too." ■

M. MITCHELL WALDROP