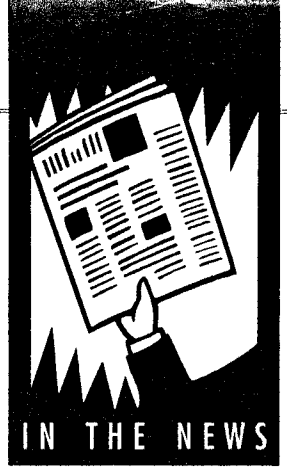

How software affects the world. How the world affects software.

Galen Gruman, Editor

LESSONS FROM PATRIOT MISSILE SOFTWARE EFFORT

WHEN NEWS FLASHED THAT IRAQI Scud missiles were streaking towards Israel in the recent Persian Gulf war, many people wondered whether the software-controlled Patriot antimissile system would actually work. The Patriot system was one of many technology-oriented systems deployed in this war that had never before been used in a war. Others included the M-1A tank, night-vision goggles, Tomahawk cruise missile, and the software system used to plot and schedule the roughly 1,000 attack-plane sorties daily.

For several years, the US military's reliance on computer-controlled high-technology weapons came under question by lawmakers, scientists, and activists worried by failures in high-profile programs like the Bradley fighting vehicle and apparent embellishments of results early in the Strategic Defense Initiative antimissile research program.

But the Patriot systems deployed in Israel and Saudi Arabia worked. Of the 43 Scud missiles launched against coalition forces in Saudi Arabia and Kuwait, Patriot systems intercepted 29, while operators let another 11 land in the desert or ocean because the trajectories computed by the Patriot system showed they would not fall near civilian or military areas, according to Jan Walker, a US Defense Dept. spokeswoman. No intact missile reached its target.

Ironically, although praised by military and political figures as an example of American high-tech abilities, the software controlling the Patriot missile system is rooted in 25-year-old technology. In fact, proof-of-concept work on the original system began in 1965, two years before even structure programming had been proposed as a development methodology. Engineering development began in 1972.

RESULTS NOT TRANSFERABLE. Although successful, the system components and the experience from building the Patriot are generally not transferrable to other projects.

Other missile programs — like the Eris system for intercepting long-range missiles now being tested and the Erint, Thaad, Arrow, and Hedi medium- to long-range systems under development — are not likely to be able to directly use the code or experience from the Patriot system, said Larry Moore, chief of software engineering at the Defense Dept.'s Patriot project office in Huntsville, Ala.

While some of the algorithms might be reusable, the code is not because it is written in Jovial, not the now-required Ada, and because the Patriot's radar is a unique one not adopted by succeeding missile systems, he said. "Much of the sophisticated logic in the [software] system is tied to the [hardware] system," Moore said.

Furthermore, the knowledge gained from the effort resides mainly in the developers' heads, Moore said. "We still have people who were working in 1972. What we lacked in tools and methodology in those days, we've overcome with people who understand the Patriot software and the Patriot system," he said.

ENGINEERING PRACTICES. The Patriot software development followed traditional systems-engineering principles rather than any software-specific methodologies. "I have a hard time separating myself as a software developer from a systems developer," Moore said.

Although praised as state-of-the-art, the software controlling the Patriot missile system is rooted in 25-year-old technology.

The developers relied extensively on simulation and hardware-in-the-loop testing in designing the system. They simulated the hardware components and their interactions to optimally model the project's requirements. The intent was to demonstrate "this is what we need," said A.Q. Oldachre, the Patriot's deputy project manager, "Then the question was 'Can we build it?' " As components were created, they were added to the simulation system, letting the designers test the hardware in the loop early and incrementally, as more components were developed. "We progressed from unit testing to testing with basic tactical components" and finally to a full system with the full environment simulated," Moore said.

Based on the results of this interactive, simultaneous prototyping of both the hardware and soft-

Continued on page 108

Editor: Galen Gruman
IEEE Software
10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Internet g.gruman@compmail.com

PATRIOT MISSILE SOFTWARE LESSONS

Continued from page 105

ware, "we had to modify the design and simulation, then go back the other way: 'Can I make some trade-offs and still achieve the same objectives?' " Oldachre said. "When modeling a complex environment and the predicted response to that environment, you end up with a certain amount of requirements changes," Moore said.

Eventually, the simulated hardware had been replaced by the actual hardware and tested at the White Sands Missile Proving Grounds in New Mexico. "The result was a modular configuration that was controllable by operational software," Oldachre said.

Because systems integration had proceeded incrementally, the hardware and software were known to work together. Software design and implementation were likewise intertwined, avoiding any surprises at integration and testing. "It's absolutely essential that you have a tight coupling between the system aspects — not to put your coders off over here and your software designers off over there," Moore said, "You're going to have a disaster if you don't have good communication there." The communications was "not perfect," he said, "but we sure recognized [its] importance."

"We're not in the state of the art," Moore said, "But the things we do we do very smartly." Because the project started before modern software-engineering approaches, "we didn't have the ability to take advantage of the newer methodologies," he said. "We certainly have a lot of people who stay abreast of modern practices, and we have been able to use some of these and the tools that go with them," he said. For example, the developers adopted structured programming practices "very early on," Moore said, and used some debugging tools — including some developed internally. "But we can't afford the luxury of going back and redoing it" with these new methods, he said.

RECONFIGURED MISSION. At the time it was proposed, the Patriot system embodied a new, risky concept: control based in software, rather than hardware, so the weapon's mission could be changed. "We did start out with the premise that it was to be a software-driven system, with all calculations made on the ground, to give it more flexibility," Oldachre said. The Patriot was originally designed as an antiaircraft — not antimissile — system. But by 1977, concerns about short- and medium-range missile attacks led to the Defense Dept. requesting that the missile's developer, Raytheon Corp., reprogram the system to attack such tactical ballistic missiles, in addition to aircraft.

In 1980, after the control software had been rewritten to take missiles into account, the Army tested the reprogrammed missile system. Although the hardware had a mean time to failure of 2,000 hours — twice as good as the specification's requirements — when coupled with the new software, the mean time to failure dropped to minutes, said Tony Shumskas, a military staff analyst for testing and engineering at the Defense Dept.

> The Patriot system embodied a new, risky concept: control based in software rather than hardware, so the weapon's mission could be changed.

The failure appeared to be worse than it was, Oldachre said, because the failures were related to diagnostics and maintenance software used by soldiers to uncover and fix hardware problems in the field. That software had not been worked on during the several years of the flight-software redesign so developers could focus on the flight software. "We had lost four years of development work in the maintenance side and had to get into a recovery mode. By 1980, it had just not caught up," he said. It took three more years to bring the maintenance and diagnostics software up to the level of the flight software.

Such an approach "cost us some time," Oldachre said, but was required by the Air Force, which did not want to wait for a complete system to make its decision on whether to acuire the Patriot. "When you do that, you ought to do it with the recognition that you'll have to come back" and complete the ignored components, he said.

The reconfiguration effort cost less than $160 million — a "fairly inexpensive" price, Oldachre said. All told, the development effort cost from $500 million to $600 million.

PATRIOT CAPABILITIES. While the current software size is classified, the original specs called for a system of about 2 million lines to handle the system modules: higher echelon command and control, control for various firing platforms, information coordination among the system elements, radar-waveform processing, radar engagement control, discrimination system, real-time tracking, simulation, training, and diagnostics (including real-time fault detection). The support software — for simulation, training, and diagnostics — takes "a big part, but not more than 50 percent" of the total system, Moore said. The code is written mostly in Jovial.

The system itself involves more than guiding a missile. The Patriot system is controlled by a ground-based computer that coordinates a phased-array radar and the missile battery. The computer aims the radar, interprets the data, and presents the results to an operator, who then decides whether to fire a missile. The system directs the Patriot missile against the incoming missile or airplane. The system can track and engage multiple missiles and fighter planes simultaneously. Although the exact range is classified, the radar's radius appears to be a few miles, which is consistent with its performance in the war and with its mission of protecting military targets. The missile itself is "just a peripheral," Moore said.

Like other antimissile systems, the Patriot is designed to intercept a missile in flight. But a successful intercept does not mean an enemy missile is rendered harmless. In one case, an Iraqi missile intercepted by a Patriot system stationed in Israel merely knocked the missile off target. In several other cases, debris from the intercepted Scuds damaged homes and injured people.

Tougher for any interceptor are missiles that fall apart in flight, since the Patriot at best will hit only one piece, which may not be the warhead. For example, of the Scuds launched against Saudi Arabia and Kuwait, seven (including some that were allowed to fall) broke up during flight and could not be targeted — including one missile whose debris fell on a military barracks near Dhahran, Saudi Arabia, killing 28 soldiers. The Scuds fell apart because they were apparently old and poorly maintained, but both the US and Soviet Union have multiple-warhead missiles designed just to avoid total destruction by antimissile systems.