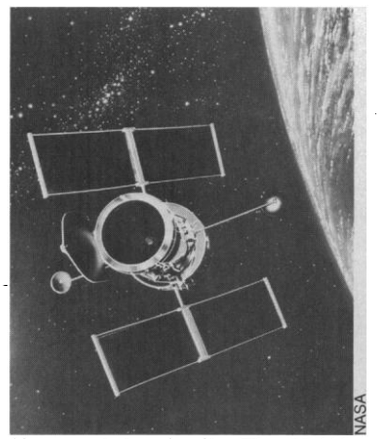
Congress Finds Bugs in the Software

Dig into any of the government's chronically over-budget and behind-schedule development programs—the Hubble Space Telescope or the B1-B Bomber, for example—and you'll find that a good fraction of what gets labeled as "waste, fraud, and abuse" actually stems from crummy software. Not only do the development agencies habitually spend millions of dollars on operations software that is buggy, inadequate, and late, they then have to spend millions of dollars more to fix it.

So says a just released report* from the House Science, Space, and Technology Committee's Subcommittee on Investigations and Oversight. Written by subcommittee staffer James Paul, who spent 2 years working on it, the report also names a culprit: the government itself. "Government policies on everything from budgeting to intellectual property rights have congealed over time in a manner almost perfectly designed to thwart the development of quality software," it says.



How not to do it. *The Hubble Space Telescope was a software nightmare.*

Paul's brief, but strongly worded report echoes the complaints that computer scientists have been muttering for years. "The [federal government's] procurement process is as much or more to blame for poor software as any other single thing," says Peter Freeman of George Mason University, who just finished up a stint as head of the computer research division at the National Science Foundation. "There are huge numbers of people involved [in the agencies] who fundamentally don't have the knowledge or experience to make good decisions with respect to procurements."

The report says that reform must begin on the conceptual level, because purchasers still regard software as an afterthought. Project managers, agency heads, and congressmen alike tend to focus on sexy hardware such as radars, airframes, and engines, assuming that the software to control all this gadgetry can be taken care of later. Yet that assumption can be costly. In 1979, the Nuclear Regulatory Commission shut down five nuclear reactors for upgrading; a software flaw in the computer-aided system used to design them had left them vulnerable to earthquake damage. In the mid-1980s, a Canadian-built radiation therapy machine, the Therac-25, killed at least four people when a software error irradiated patients with massive overdoses.

"Software," says the report, "is now the choke point in large systems."

On the other hand, it will be difficult to make the software development process more flexible because the federal procurement system effectively demands that contractors write the software using bad methodology. "Out on the technical frontiers," the report says, "the government requires a legally binding contract specifying in excruciating detail exactly how the system will look at delivery some years hence." The tacit assumption is that programmers will then use these specifications to write, test, and debug the software. In the programming community this approach is known as the waterfall model because the work supposedly cascades from one stage to the next in systematic progression.

But modern programmers consider the waterfall approach an abysmal way of doing things. Especially when it comes to high-tech systems such as the Space Telescope's scheduling software or the B1-B's defensive electronics countermeasures system—both of which were software fiascos (*Science*, 17 March 1989, p. 1437)—it is essentially impossible for anyone to write detailed specifications in advance. Not only does the hardware evolve during development, thereby forcing the specifications to change, but the hardware engineers themselves have no way of knowing what they really want from the software until they have had a chance to try it out.

This is why the modern "evolutionary" approach to software development looks less like a waterfall than like a spiral. Starting with general requirements, the programmers quickly cobble together one or more prototype systems. The engineers then try out the prototypes on the evolving hardware. Their suggestions guide the programmers in refining new prototypes. And the process repeats as long as is necessary. The payoff is that the programmers have a much better chance of catching errors in the design phase, when the bugs can be fixed at an estimated one-tenth to one–one hundredth the cost of fixing them after deployment, and the software as a whole has a much better chance of doing what it really needs to do.

But flexibility is precisely what the spell-it-out-up-front procurement culture lacks, says the report. In addition, the bureaucracy balks at the big up-front investment in problem definition that must be made when the evolutionary approach is used. Furthermore, as the report notes, "no program manager relishes the thought of defending a request for funds when the major activity seems to be endless arguments over abstruse technical points by large numbers of well-paid engineers."

So, what can be done? In the near term, very little, says the report.

The Department of Defense and a few other agencies have begun to move away from the waterfall model, and Paul thinks those steps should be encouraged. Better methods should be developed for evaluating the quality, reliability, and safety of software, the report says. And perhaps the software community should be encouraged to establish some form of professional certification standards.

But nothing fundamental is going to change without changes in the procurement culture, says the report. And that is going to take awhile, even though chronic cost overruns and scandals are creating mounting pressures to do so. "It's not something you can jump right in and legislate," Paul told *Science*. "The federal procurement system is like a software system with bugs. Every time it's broken down, somebody has patched it. But keeping it together is getting harder and harder and costing more money. And at that point, an experienced software engineer would throw up his hands and say, 'Hey! Let's toss this out and start over.' "

■ M. MITCHELL WALDROP

*"Bugs in the system: Problems in federal government computer software development and regulation" (Subcommittee on Investigations and Oversight of the House Committee on Science, Space, and Technology, Government Printing Office, Washington, D.C., September 1989).