

Software safety focus of new British standard

Galen Gruman, Soft News Editor

The British Defence Ministry expects to issue a new software-safety standard this spring that will require the use of formal methods and mathematical verification on all safety-critical software. Only developers who prove that their software is not safety-critical will be exempt from the requirements.

The standard, MoD-Std-0055, will ban the use of assembly language, limit the use of high-level languages like Ada to safe subsets, and require the use of static analysis. It also sets standards for project engineers. It will require that an engineer sign off on the software's safety compliance, that the engineer have taken accredited formal-methods instruction within the past two years, and that an independent engineer with similar accreditation also sign off on the system. This is similar to the responsibility and requirements enforced on systems-safety engineers for the overall project.

The 0055 standard will be in effect for two years, during which time the Defence Ministry will revise it on the basis of industry's experience. The intent is to develop a long-term standard, said Kevin Geary, a software consultant for the British navy's procurement department who is working on the 0055 standard. The ministry is also working on MoD-Std-0056, a hazard-analysis standard that will help software developers determine where to apply formal methods and mathematical verification, Geary said. "Both mathematical verification and hazard analysis must be performed to provide software with acceptable risk. Neither is adequate alone," said Nancy Leveson, a software-safety expert and a computer-science professor at the University of California at Irvine.

Pros of formal methods. The 0055 standard has been called a "landmark" by those in the software-safety and formal-methods communities, who argue that assigning responsibility to software engi-

neers, as has been tradition in hardware engineering, will help encourage changes in development methods that will help assure safe systems. Safety is increasingly important because software is becoming a greater part of critical systems like aircraft controls, medical devices, nuclear-power plants, early-warning defense systems, and missile controls, they said.

Most software-engineering standards depend on testing, which is not always reliable, Geary said. "The problem with software is that you must test against specifications. If you didn't get the specifications right, you might not get the software right," he said. However, mathe-

***The forthcoming
UK Defence Ministry
standard will require the
use of formal methods
and mathematical
verification for
safety-critical software.***

matical analysis of formal specifications notations can be used to find errors in the specifications, Leveson said.

The increasing number of tools like Zed, Vienna Development Method, Spade, and Malpas will help make the implementation of formal methods possible because these tools can perform static analyses of information flow and semantics quickly, rather than in the years required with manual techniques, Geary said.

Formal methods and mathematical verification are often considered too difficult to apply, Geary conceded. "There is a lot of unease, but it's quite surprising that there are a lot of key people who've

come around after looking at it," he said. Geary cited IBM's British development center, which decided for commercial reasons — not for government or other outside requirements — to use the Zed formal method on CICS development. "People's resistance is based on ignorance," Geary said.

Another source of resistance is the confusion between formal, mathematical methods and mathematical correctness. "Correctness is a meaningless goal for real systems. For example, do you have a 'correct' airplane?" Leveson said. "A more realistic and useful goal is to build a system that satisfies a given set of functional and mission requirements while at the same time trying to satisfy constraints of safety, security, and cost," she said. Many of these goals involve trade-offs in setting priorities, she said.

Leveson compared formal methods to traditional hardware engineering: "Engineers build formal mathematical models and then use analysis methods to determine whether the model has certain desired properties," she said, "which should be the role of formal methods in software engineering." (Leveson's "Safety as a Software Quality" essay in this issue's QualityTime, on pp. 88-89, gives more details about this process.)

"Both software engineers and hardware engineers specify design," Geary said. "The only difference is how tangle [the product] is," he said.

Still, software engineers do face a burden that their hardware counterparts generally do not: the complexity of their product, said Martyn Thomas, chairman of Praxis Systems, a software-engineering consulting firm in Bath, England, that does much work in safety engineering. Traditional engineers like bridge builders "never had techniques for design, which is more important for software because that's where the complexity comes in. It's not a software problem but a design-complexity problem," he said. Whether overtly or covertly, the profes-

sion must move from certifying systems based on their probability of failure to certifying them based on how they were designed, Thomas said.

More than formal methods. The International Electrical Commission is also working on safety-engineering standards for software. But unlike the Defence Ministry's 0055 standard, the IEC's software-safety efforts, which are also due this spring, will not prescribe development and verification methods. Instead, they will tell developers how to determine both their levels of criticality and the techniques that could be applied to verify safety, said Phil Bennett, chairman of IEC/SC65A/WG10, one of the IEC safety-engineering efforts. Bennett is also a principal at the Center for Software En-

gineering in Flixborough, England, a private systems-verification firm set up by the British government in 1984 and later privatized.

Bennett considers formal methods to be only one of several effective techniques. "You can get away with formal methods in defense, but I don't know many steel-plant managers who can do this," he said. (Bennett is a former control engineer for British Steel.) He recommended that such managers use petri nets and modeling — techniques they can use with their existing engineering knowledge. Formal methods may be widely applicable in 10 to 15 years, Bennett argued, but "in the meantime, we need to use something."

The other IEC effort, IEC/SC65A/WG9, is working on a framework for dif-

ferent levels of safety that would depend on the integrity of each component, said Ron Bell, the group's chairman. If you have only a software system to ensure safety, "the level of software integrity has to be very high," Bell said, but if there is a hardware backup or secondary software system, the level for each component would be lower, he said. The intent is to apply risk analysis to the total package, he said, not to components in isolation.

Like Bennett, Bell said formal methods would not be a requirement in his working group's standard. "We want a degree of formality applied, depending on the nature of risks. But we will not mandate that all software be developed with formal methods," he said. At the highest level of safety criticality, the set of required techniques might include those specified by the Defence Ministry's 0055 standard, but it could also include other techniques so developers could choose one they are familiar with or believe is best suited for the task.

Bell is working on a similar framework for use by Britain's Health and Safety Executive, a quasigovernmental agency that is the major regulator for workplace standards.

Pressure to change. Using standards to force changes in development approaches is not unique to software engineering for safety-critical systems, said Peter Neumann, a computer scientist at SRI International in Menlo Park, Calif., who also runs the Risks Bulletin Board on several research networks. He cited the security-validation requirements of the US Defense Dept.'s *Trusted Computer System Evaluation Criteria*, popularly known as the Orange Book. "They've jawboned a lot of vendors into doing things better," he said. For example, about a half dozen vendors are working on AI-level systems, which require formal proof that the specifications are consistent with the requirements, Neumann said. A year ago, only one company (Honeywell) was working on an AI-level system, which has since been completed.

Those involved with the 0055 standard expect some industry opposition at first, but "the key people in these industries — via the papers they submit at conferences — are effectively moving in the same direction" as the standard, Geary said.

Simply having the requirement should result in progress, Thomas said: "People train damn fast under pressure." The British Defence Ministry's 0055 standard for defense contractors should also persuade the nuclear and aerospace industries to adopt similar standards, Thomas said.

Britain leads in several safety efforts

The British Defence Ministry's software-safety effort is one of several software-safety efforts in Britain. Part of the pressure for safety in Britain results from a law, the Consumer Protection Act of 1987, that makes chief designers liable for damage and injury caused by design errors, said Kevin Geary, a software consultant for the British navy's procurement department. While not explicitly written to include software designers, "the UK legal system goes to the spirit of what is meant," not just the letter of the law, he said.

In February, the British Computer Society's Safety-Critical Systems Group issued a policy statement calling for "a system of registration [of safety-related computer systems], with mandatory fault reporting, so that minimum standards can be enforced and data can be gathered [that] will allow the success of different approaches to be assessed."

Ironically, the statement was criticized, mostly by American scientists, for being too timid in concluding "that there is no evidence that current [safety-related computer systems] pose a serious threat to the public," said Martyn Thomas, the group's chairman and a chairman of Praxis Systems, a software-engineering consultant in Bath, England. Abashed at the criticism of the statement's equivocal conclusion, Thomas said the equivocation was meant to prevent people from panicking about safety issues, although he acknowledged critics' fears that the equivocation might undermine the statement's call for action.

Thomas said the lack of public awareness about the potential for software failures is worrisome: "There is some threshold of risk that we cannot safely cross with our technology. The problem is that we don't know where it is or how to measure it."

The International Electrical Commission is working on two safety-engineering standards that incorporate software safety; both working groups are chaired by British scientists. The proposals are due for comment this spring as well. In the US, the Instrumentation Society of America is advising the IEC effort.

While Britain has moved forcefully on software-safety issues, the US is starting to follow suit. An invited workshop on formal methods for software safety with participants from Britain, Canada, and the US is planned for July in Halifax, Nova Scotia, Canada. Participants will include representatives from government agencies looking into whether such methods should be strengthened in regulated or government-purchased software. Several firms and government agencies are pursuing similar research.

—Galen Gruman, *Soft News Editor*