Improving Software Engineering Practice

Patricia Sanders
Office of the Undersecretary of Defense for Acquisition and Technology

The complexity and size of the Department of Defense (DoD) necessitates an extensive, software-dependent computer network; however, past experience has shown that software is rarely defect free. In an organization that requires pinpoint accuracy to save lives, functional software is an area in which excellence cannot be compromised. The only *way to improve software's performance is to improve the way in which software is developed.*

Increasingly, we live in a complex world where software is indispensable to everyone's life. And increasingly, we are frustrated by the software with which we are forced to live.

Consider if a carpenter were compelled to work with tools as unreliable, complex, and generally inaccessible as most of our computers. Can you imagine turning on a sander and getting a message that says, "general protection fault," at which point the disk flies off and the whole thing self-destructs? Why do we put up with this from tools which, for many of us, are as indispensable as a sander is to someone who makes a living by woodworking?

If we want quality software, we must accept responsibility for how it is developed. Improving *what* you build means improving *how* you build.

The DoD's Environment

The DoD is a large and complex organization. There are 1.4 million active duty men and women in the uniformed services and about 800,000 civilians. Every year, we recruit around 200,000 new people to join the armed forces and separate about 220,000. So approximately 30 percent of our organization is either coming or going every year. We have about 250 major installations worldwide. We operate 550 public utility systems—gas, water, electricity, and natural gas distribution.

We support one of the larger school systems in the world comprising 126 high schools and elementary schools. We are the world's largest day-care provider—300,000 children are enrolled in DoD day-care centers.

This article is based on a speech given at the 1998 Software Engineering Symposium in Pittsburgh, Pa. Sept. 16, 1998.

We have 28,000 separate computer systems that we are tracking for the year 2000; 2,800 of them are mission critical. We disperse 5 million paychecks and about 400,000 bonds and about 600,000 travel vouchers and 800,000 contract actions every month. In Columbus, Ohio, where we do our large contract management administration, there are about 390,000 contracts under administration. We disburse the staggering amount of about $43 million an hour.

We sustain operations in every time zone. Today, there are about 120,000 military personnel deployed around the world, in addition to the 200,000 who are permanently stationed overseas. We operate over 400,000 vehicles—everything from sedans and buses to the street sweepers used to clean runways to combat vehicles to tanks to armored vehicles.

As an organization, one of our challenges is to concurrently manage about 70 years of technology. We operate, on a daily basis, aircraft that were designed back in the early 1950s, and we still have to maintain them, buy spare parts for them, and keep them updated. At the same time, we are working on research and development programs for systems that will not be fielded until between 2015 and 2020. To manage that spectrum of technology is a constant challenge.

In information technology, we operate some of the world's most advanced computers, and yet, just last year, we moved a number of Burroughs punch card readers to a new megacenter because we are still operating punch cards for some business applications. We manage an astounding spectrum of technology. The DoD is not only the largest but also is probably the most complex organization in the world.

Yet, this is an organization that has had its budget cut for 15 consecutive years, has undergone significant reductions, is operating at 46 percent of the budget resources it had only 12 and 13 years ago, and has a third of its personnel coming and going in any one year. And still, it is an organization that is able, within a month, to send 60,000 people to the Persian Gulf along with 400 combat aircraft and 500 cruise missiles and could carry out war tomorrow if necessary.

War-Fighting Changes

We have been engaged in an unprecedented change in the way we think about warfare. It has been going on for some time, and it is moving into a highly sophisticated dimension. The change accelerated in the late 1970s and the early 1980s when we were starting to bring microprocessors into weapons systems.

We are on the edge of breaking through in what we call *network-centric warfare*. To put it bluntly, the DoD is in the business of destroying things. In warfare, we try to do that in a focused way without doing a great deal of damage to things we do not want to destroy. We have done that before by putting extremely lethal and highly accurate capabilities in the hands of whoever was doing the shooting at the time. We are now moving into a more interesting and highly leveraged dimension where the person who launches the missile does not have to see the target. We are going to be sharing information across a network and still be able to attack and destroy an opponent. This dramatically improves the survivability of our own forces, of

course. It is going to be revolutionary. The situational awareness that will be on our side of the battlefield will be three or four orders of magnitude better than our opponent's. We call this *information dominance*.

In the past, the dilemma of warfare was always how to bring mass together for its effect over your opponent without giving your opponent lots of targets at which to shoot. It is the classic dilemma. One of the reasons there were so many casualties during the Civil War was that firepower technology had progressed so much farther than communications technology. We were still massing people close to each other, side by side, so the soldiers could hear shouted orders. Firepower technology had advanced, however, so that cannon could mow people down. We are now going to be in a wholly different world where people do not have to see each other and yet, they can operate together as a combined arms team. What we expect to be able to do is quite dramatic.

Importance of Software

So a new breed of "knowledge warriors" has begun to emerge who recognize that knowledge can win or prevent wars. And this is causing fundamental changes in what is important to our war-fighting capability.

Today, about 10 percent of the weight and one-third of the cost of modern combat aircraft are composed of electronics and related components. Principal among the latter is software—a substance that weighs nothing but costs inordinately.

There has been nothing like the headlong rush to software since the similar rush to electronics after World War I. The average automobile of today has more software in it than the first Apollo spacecraft to arrive at the moon 30 years ago.

In the Gulf War, television cameras, ravenous for dramatic visuals, focused on F-14 aircraft roaring off the decks of carriers, Apache helicopters swooping over the desert, M-1A1 Abrams tanks growling over the sands, and Tomahawk missiles singling out their targets. Pieces of hardware became overnight stars. But the real star was the invisible software that processed, analyzed, and distributed data, though no television watcher ever saw those who produced and maintained it—America's software soldiers.

Software is changing military balances in the world. Today, weapons systems are mounted on or delivered by what we call "platforms"—a missile, an airplane, a ship, or even a truck. What we are learning is that cheap, low-technology platforms that are operated by poor, small nations can now deliver high-technology, smart firepower if the weapons are equipped with smart software. Stupid bombs can have their "IQ" raised by the addition of retrofitted components dependent on software for their manufacture or operation.

In previous eras, military spies paid special attention to an adversary's machine tools because they were needed to make other tools needed to produce arms. Today, the "machine tool" that counts most is the software used to manufacture the software that manufactures software, because much of the processing of data into practical information and knowledge depends on it. The sophistication, flexibility, and security of the military software base is crucial.

Software Costs

The DoD does not track software spending independently of other expenses. But, Federal Sources, Inc., a Virginia-based marketing firm that tracks government spending, completed a survey around September 1997, which concluded that by 2002, the DoD will spend more than $20 billion annually on software used for weapons systems, information technologies, and command, control, communications, and intelligence systems (not including personnel, management, and non-tactical systems). The Federal Sources review estimates military aircraft require by far the largest software expenditures, roughly $5 billion in 1998. Ships sail in at a distant second with barely more than $1 billion. Ordinance and weapons, lumped together in one category, tie for last with vehicles at less than $1 billion.

A study by the Electronics Industries Association estimated in 1995 that the DoD would spend $42.5 billion on computer systems, of which $35.7 billion would be on software—about two-thirds of that on maintenance. These analyses are important in that they illustrate the increasing reliance on software for warfare in the information age. Some, in fact, predict a future in which military hardware procurement becomes secondary to software purchases.

Costs of Software Failure

Information or knowledge superiority may win wars, but that superiority is exceedingly fragile. In the past, when you had 5,000 tanks and your enemy had only 1,000, you may have had a ratio of 5-to-1 superiority. In information war, you can have a ratio of 100-to-1 superiority, but it can all turn on a fuse or a lie or on your ability to protect your advantage from those who want to steal it.

The key reason for this fragility is that knowledge, as a resource, differs from all the others. It is inexhaustible. It can be used by both sides simultaneously, and it is nonlinear, which means that small inputs can cause disproportionate consequences. A small bit of the right information can provide an immense and strategic or tactical advantage, whereas the denial of a small bit of information can have catastrophic effects.

Pentagon leaders have been increasingly stunned upon learning that some of our computer systems can be and have been tampered with by hackers and by military exercises that demonstrate how easy it is for hackers to cripple U.S. military and civilian computer networks.

But my issue is not so much one of information assurance—although that is decidedly a top priority for the DoD, one with which the Software Engineering Institute (SEI) is providing major assistance—rather, I want you to focus on the implication that you succeed or fail on the software. It does not matter how much speed, or how much stealth, or how much armor plating you have; you will not succeed if the software does not work.

The cost of software failures can be high. In the commercial world, a system error in American Airlines scheduling software that incorrectly showed flights full resulted in a $50 million loss. System downtime for American Express costs $167,000 per minute; for Charles Schwab, the penalty is $1 million per minute.

The DoD's damages can be more expensive. Under the START II treaty (Strategic Arms Reduction Talks), three-quarters of our nuclear deterrent is in our fleet ballistic missiles, the effectiveness of which is in the hands of their fire-control software.

So, I contend that software that does not work is self-inflicted information warfare. The policies, processes, and practices that guide the development and use of information technology in general and software in particular are a crucial component of our strategy.

Expectations

Unfortunately, our overall track record for producing quality software is underwhelming. There is a perception that the DoD has a perfect record on software development—we never get it right.

According to the results of a study on U.S. software development reported by the Standish Group in 1996,

- In 1995, only 16 percent of software projects were expected to finish on time and within budget.
- In larger companies, only 9 percent of the software projects will be completed on time and within budget.
- An estimated 53 percent of projects will cost nearly 190 percent of their original estimates.
- Projects completed by the largest American organizations have only 42 percent of the originally proposed features and functions.

These findings show slightly better performance than an earlier DoD study. In that analysis of DoD software development projects that were originally estimated to take between two and three years to complete, there was, on average, a 36-month schedule slip, and one-third of all software programs were canceled before completion.

Despite the real and potential benefits software holds for us, expectations of software performance differ in interesting ways from expectations for hardware performance.

A story going around has it that at a recent computer exposition, Bill Gates reportedly compared the computer industry to the automobile industry and stated, "If GM had kept up with technology like the computer industry has, we would all be driving $25 cars that got 1,000 miles per gallon." General Motors reportedly addressed this comment by releasing the statement, "Yes, but would you want your car to crash twice a day?"

A similar story has it that if software engineers made automobiles, your car would sometimes die on the freeway for no reason, and you would accept this, restart, and drive on. Occasionally, executing a maneuver would cause your car to stop and fail, and you would have to reinstall the engine. For some strange reason, unlike a carpenter's tools, you would accept this, too.

This sort of reliability might be adequate in a word processor, but it hardly seems acceptable in a weapons system or where safety is a major consideration. After all, a soldier without a weapon is at best a tourist and more likely, a target.

Systems Engineering Process

To get good software, we need to build it right. When we track successful software developments, almost invariably, the accomplishment can be linked to the existence of good systems engineering processes because it is the application of the disciplined systems engineering process that makes the difference in achieving the functionality we seek—in both hardware and software.

As Reuel Alder observed (*CROSSTALK*, September 1998), "Discipline is no fun—I consider day planners self-inflicted torture. My idea of a good day is to wake up with no plan and accomplish more than humanly thought possible. The work would be intuitively discovered as the day progressed. Creativity and spontaneity would be enhanced, and routine, repetitive activities would be minimized. Each day would be a fresh and exhilarating experience filled with learning, personal growth, and development. The variations would be unlimited, and the success would be phenomenal.

"But if you believe the last 40 years of development data, this dream is not achievable for most software projects. Yet, we are still largely living in a dream world where we think software can be built by pure 'artists' who arrive at river's edge with no plans, and through sheer talent can turn a pile of scrap iron into a decent bridge.

"However, I have learned from unfortunate personal experience that almost all significant human achievements require more than just talent and creativity. Decades of data prove it: Even the best software artists do better work when they start with a foundation of planning, preparation, and discipline."

Consider requirements management. A 100-company survey by Standish Group International found that 45 percent of a software application's features are never used, 19 percent rarely used, 16 percent sometimes used, 13 percent often used, and 7 percent always used. Yet, in spite of the fact that most of an application is seldom used, software gets bigger all the time.

I have a cartoon in my office that shows two individuals—presumably software engineers—and one of them says to the other as he is running out, "You start coding; I'll go find out what they want." Unfortunately, there is all too much truth in this picture. Because what is being developed is "only software"—and everyone knows software is easy to change—a disciplined requirements management process is all too frequently lacking. Without requirements analysis upfront, however, the results are unsatisfied needs, wasted effort, and rework.

Software may be easy to change—at least relative to bent metal—but it can still be costly in both time and dollars. It is estimated that rework is 40 percent of the cost of development. Metrics collected by Capers Jones indicate that the cost and schedule impact of defects in requirements are the most expensive of all defects, followed by defects in top-

level design (architecture), and finally by defects in code.

We also do not develop software with its lifecycle in mind. Much of the software that is operational today will still be in service several years from now. Over the service life of software-intensive aircraft and smart munitions, there is a need for continuous improvement, correction, and addition of new capability via software modification. Embedded software in weapons system platforms has evolved in operational and technical impact to the point where upgrades must be seen as major subsystems. The effectiveness and efficiency of the process for upgrading and otherwise modifying embedded software has a major impact on readiness. Each year, upgrades to the B-1, F-15, and F-16 aircraft programs cost nearly $200 million. When the planned expenditures for the B-2, F-22, and F-117 aircraft and the advanced weapons are added in, that figure doubles.

One definition I have seen for software upgrade is you take old bugs out to put new ones in. As I previously noted, approximately 66 percent of the DoD's software costs are associated with maintenance. Almost all of the systems engineering practices that have high leverage for lowering the cost of maintenance are practices that need to be implemented during development. These include

- Development practices that reduce the density of defects in the software delivered into operation.
- Effective software test.
- A strong configuration management program.
- Taking account early in the program of the engineering environment and processes that need to be in place for sustainment.

SEI's Contributions

SEI has successfully influenced commercial technology for the DoD's benefit.

SEI's function, as defined in the DoD Management Plan, is to develop and transfer important technology to the private sector so that the government can benefit from a broader base of expertise. Their work benefits both the DoD and industry by helping to define, analyze, and improve operational processes from the level of the individual engineer to practices that apply across the entire organization. They have achieved measurable success.

SEI's mission is to reduce the cost, schedule, and technical and performance risk associated with acquiring and building software. Simply put, SEI exists to help us build software "better, faster, and cheaper."

But it must be predictably better, faster, and cheaper—erratically better, faster, and cheaper is not helpful to achieve the DoD's goals for information superiority. Discipline in process and product management is essential.

For fiscal 1999, we have worked with SEI to define some focus areas for initiatives.

- Commercial-off-the-shelf-based systems.
- Survivable systems.
- Architecture trade-off analysis and product-line practices.
- Continuing process improvement.

Conclusion and Summary

In closing, I will tell you a story about the brass lamp that Secretary of Defense William S. Cohen found in his office when he first moved into the Pentagon. When he rubbed the lamp, a genie popped out and offered him one wish (in a downsizing environment, you no longer get three wishes). Cohen first pointed to the large map covering the wall and the numerous pins in the map that indicated trouble spots around the world and asked the genie to provide stability to all those locations. The genie, however, said that this was perhaps too much even for a genie to accomplish. So the secretary thought some more and asked instead that the genie provide a guarantee of error-free DoD software. The genie, upon hearing this wish, said, "Let's look at that map again."

I am more optimistic.

The information technology revolution is having a profound effect on all of us. But never lose sight of the fact that all this progress depends on one fundamental: No matter how technologically sophisticated we are, it is people who make knowledge and knowledge sharing possible.

Real process improvement is not easy, and anyone who believes otherwise has never tried it or has never helped make an improvement of lasting significance. Learning better techniques and technologies is only the beginning—there are many human aspects through which to work.

Process improvement pays big dividends for those with the discipline to do it right. With it, we can improve what we build because we will have improved how we build. ◆

About the Author

Patricia Sanders is the director of test, systems engineering, and evaluation for the DoD, where she is responsible for ensuring the effective integration of all engineering disciplines into the system acquisition process. These include design, production, manufacturing and quality, acquisition logistics, modeling and simulation, and software engineering, with emphasis on test and evaluation as the feedback loop. She is also responsible for oversight of the DoD's Major Range and Test Facility Base and the development of test resources such as instrumentation, targets, and other threat simulators. She has over 24 years experience in the DoD. She holds a doctorate in mathematics from Wayne State University and is a graduate of the Senior Executive Fellow Program, John F. Kennedy School of Government, Harvard University.

POC: Brenda Zettervall
E-mail: zetterbt@acq.osd.mil
Voice: 703-695-2300