Defense Software Development in Evolution©

Capers Jones
Software Productivity Research, Inc.

The author and his colleagues have been measuring software quality and productivity rates since 1985. They classify the projects that they examine into six major groupings: information technology software, outsource software, commercial software, systems software, defense software, and end-user development software. Applications are placed in the defense software group if they followed U.S. military or Department of Defense (DoD) standards. Overall, defense software projects have ranked near the top in software quality. However, defense software projects have ranked last in terms of software productivity, mainly because DoD standards created a number of extra tasks for defense software that do not occur in the civilian sector. In addition, the volume of defense software specifications and other paper documents has been about three times larger than civilian norms. As the DoD moves toward adopting the best civilian practices and standards for software, it is possible to see some improvement in productivity while keeping software quality levels high.

The phrase *defense software* refers to software produced for a uniformed military service such as the Air Force, Army, Navy, Marines, or Coast Guard. The term can also include software produced for the Department of Defense (DoD), or the equivalent in other countries, by civilian companies such as Boeing, Lockheed Martin, Raytheon, and a host of others.

Uniformed military personnel produce some defense software. In the United States, however, civilian contractors develop the bulk of defense software. Oversight and project management roles are normally the responsibility of military program management officers.

The broad definition of defense software includes a number of subclasses such as software associated with weapons systems; with command, control, and communication systems (usually shortened to C3 or C cubed); with logistical applications; and also with software virtually identical to civilian counterparts such as payroll applications, benefits tracking applications, and the like. The main attribute that distinguishes defense software from other types of software is adherence to military or DoD standards.

DoD Software Industry Overview

Since about 1990, the U.S. defense contracting community has been undergoing some significant changes. Waves of mergers and acquisitions have led to a reduction in overall numbers of defense contractors. While the remaining contractors are growing in size, "downsizing" or elimination of redundant personnel also accompanied the mergers so the overall defense sector has not grown in terms of demographics for several years. However, in the aftermath of the Sept. 11 tragedy, new importance has been placed on the defense community so demographics may climb again in the future.

The United States is far and away the major producer and consumer of military and defense software in the world. The volume and sophistication of U.S. military software is actually a major factor of U.S. military capabilities. All those pictures of cruise missiles and smart bombs that filled television news during the Gulf War and the Afghanistan action have an invisible background: It is the software and computers onboard that make such weapons possible.

In addition, the NATO countries tend to use many weapons systems, communication systems, logistics systems, and other software systems produced in the United States. This means that the volume of U.S. defense and military software appears to be larger than the next five countries put together (Russia, China, Germany, United Kingdom, and France). Many other countries produce military and defense software for weapons and communications systems that they use or market, including Israel, Brazil, South and North Korea, India, Pakistan, Sweden, and Japan.

Differences Between Civilian and Defense Software Practices

To an outside observer, military software and hardware projects are noticeably different from civilian norms. The first noticeable difference is the procurement process itself. The bulk of military projects are acquired by means of competitive bids, with lowest cost as a primary consideration. The bidding process is quite formal and includes rather massive sets of deliverable items from the prospective contractors. Thus, responding to a military request for proposal can be an expensive proposition in its own right.

This form of acquisition by competitive bids also leads to another difference between civilian and military norms. Military procurement is often accompanied by litigation that challenges the successful bidder. Based on discussions with DoD officials, almost half of the initial contract awards are challenged by losing vendors. An entire body of military contract law, special courts, and arbitrators deal with these challenges. In contrast, less than 10 percent of civilian contracts go to litigation to challenge the winning bid.

As a result of frequent litigation challenging the initial contract awards, there is often a six- to 18-month delay in reaching a final decision on military software contracts and starting the work. This means that many large military software and hardware projects are under immediate schedule pressure. Since schedule pressure is one of the major root causes of software failures, some projects that are rushed tend to run late, have poor quality, or end up being canceled since they cannot meet operational requirements.

Another difference between military and civilian practice is readily apparent once the contract work begins. The relationship between the DoD and its contractors had tended to be somewhat adversarial. As a result, the oversight and control requirements of military projects have been more extensive and burdensome than civilian norms. This has had a direct and tangible impact on defense software productivity. On the other hand, the number of very large software projects successfully concluded in the defense domain appears to be better than civilian norms for the same sized applications. In other words, defense software may have more front-end litigation than civilian software, but fewer instances of litigation for non-performance at the end.

Due to elaborate oversight requirements, the volume of planning and tracking paperwork required for a typical military software project has been about three times larger than for civilian software projects of the same size, based on our comparisons. Indeed, software requirements, software specifications, and almost all forms of text-based documents were several times larger for military projects than for equivalent civilian projects.

The large volume of paper documents is one of the main reasons why military software productivity rates lagged behind all other domains. About 400 English words were produced for every source code statement in the Ada95 programming language on typical military software projects in the 1980s and early 1990s. These words cost at least twice as much as the code itself. It is not unusual for large defense projects to accumulate roughly 50 percent of total costs in the area of producing and reviewing paper documents. This is far more than for any other kind of software.

While every software project needs requirements, specifications, plans, and deficiency reports, about half of the words created for military software projects seemed to be due to the very elaborate oversight and status reporting criteria associated with military contract work. Basically, some of the documents are produced to demonstrate contract compliance rather than to add technical content to the project itself.

Dealing with the DoD and the military services for business and contract purposes is so complex and specialized that companies actually doing significant amounts of military business usually have specialist military proposal and contract personnel who often are retired military officers. It is very difficult for amateurs to bid successfully on a military contract.

Being the world's largest producer and consumer of military software, the United States' software production methods are of global importance. In the United States in 1994, the Secretary of Defense William Perry issued a major policy statement [1] saying in effect that DoD standards no longer needed to be utilized. Instead, the armed services and the DoD were urged to adopt current civilian best practices.

Immediately, several task forces and study groups were created to explore leading civilian software practices. However, the military community has a conservative bent. Many military and DoD standards, such as MIL-STD-2167A or MIL-STD-498 [2], have continued to be the de facto standards of the military world, if for no other reason than because military contractors have used them for so long they are comfortable with the nomenclature and requirements.

Since civilian software fails, too, (witness the protracted delays associated with the luggage handling system of the Denver Airport [3]) another challenge for the defense community is to select practices from the civilian sector that truly do work, as opposed to practices that are merely fads. James Johnson and his colleagues at the Standish Group publish an annual report on software failures [4].

Further, some of the civilian standards, such as the ISO 9001-9004 quality standards, create document volumes that are just as large, or larger, than military standards such as MIL-STD-2167. Overall, selecting the *best* standards in either civilian or defense sectors is not necessarily an easy task.

"Organizations at or above CMM Level 3 are more likely to be successful on large systems [larger than 10,000 function points or 1,000,000 source code statements] than those at Levels 1 or 2."

The World Wide Web has many interesting sites dealing with the evolution of *military standards* toward civilian equivalents such as those published by well-known standards organizations, including the Institute of Electrical and Electronic Engineers (IEEE). Using a search engine with the key words "military standards" will bring up more than 20 relevant sites. One of the relevant documents is an interesting report on "Systems Engineering Standards and Models Compared" by Sarah Sheard and Dr. Jerome Lake. This document is available via the Software Productivity Consortium at <www.software.org/pub/externalpapers/98042.html>. On the whole, the best models for the military domain would be the large civilian systems software producers such as AT&T, IBM, etc.

The phrase *systems software* refers to software applications that control complex physical devices. Examples include digital computers, modern telephone switching systems, aircraft flight controls, and robotic manufacturing tools. The larger systems software applications, such as IBM's multiple virtual storage (MVS) operating system, are about 100,000 function points in size, which is equivalent to roughly 10,000,000 source code statements in common procedural languages such as Fortran, PL/I, or Ada.

Systems software is of similar size and complexity levels to many large-scale military applications. However the civilian systems software domain manages to build large applications with smaller specifications, shorter schedules, lower costs, and equal or higher quality than normally found on defense projects. The companies that build systems software tend to utilize sophisticated internal standards augmented by major international standards, such as those published by the IEEE and ISO.

Both the civilian systems software domain and the defense software domain excel in software quality control. Since both domains are concerned with complex hardware platforms that are controlled by software, it is imperative to have state-of-the-art quality control or the hardware devices may fail or perform in hazardous ways.

In contrast, the companies that produce information systems, commercial software packages, and software not controlling physical devices often lag in software quality control. There are both social and technical reasons for this. For example the systems and defense software domains almost always have formal quality assurance departments, while the information systems and commercial vendors are not as likely to have quality assurance groups.

Historically, the systems software and defense domains evolved from older engineering groups that had quality assurance support even before computers were utilized. The information systems domains evolved from accounting, finance, and business operations that seldom utilized quality assurance before computers were common.

Another aspect of the DoD attempt to move in a civilian direction is increased usage of commercial off-the-shelf software (COTS). Obviously the use of COTS packages refers to ordinary business and personal software packages such as databases, payroll programs, spreadsheets, and the like. The COTS concept is clearly not aimed at sophisticated weapons systems where no civilian packages exist.

Unfortunately, the military and defense domain have no strong incentive for adopting civilian best practices other than innate professionalism. The DoD itself and the military services are not profit-making organizations. If they tend to overspend or develop software in a way that is more costly than the civilian sector, so long as the fundamental mission requirements are not compromised, there is no overwhelming reason to improve.

For contractors, there are actually business reasons for staying somewhat inefficient compared to civilian norms. For time and materials contracts, there would be a negative incentive for adopting civilian best practices since increased productivity and shorter schedules would reduce the revenues and the profits from major contracts.

For fixed-price contracts, a case might be made that adopting civilian best practices would lower costs and raise the probability of gaining the contract. However, artificially low bids are common enough that this strategy might not be effective. The whole process of military procurement and contracting is in need of very careful analysis and possible re-work.

Defense Software Technologies

The military services and the DoD have been quite active in software technology research. Many initiatives have been funded and several prominent organizations, such as the Software Engineering Institute (SEI) and the Software Productivity Consortium (SPC), have focused much of their research on defense and military software.

Historically, the major programming languages used for military software included assembly language, Fortran, and some specialized languages that were seldom used outside of the military domain: Jovial and CMS2. The Ada83 programming language and the newer Ada95 programming language continue the tradition of developing specialized languages for defense software. However, the Ada languages have also attracted some civilian users, especially so in Europe.

Because of the diversity of software applications under the overall military umbrella, almost all programming languages are used. For example COBOL is used for more business-oriented military software such as payroll applications. The C and C++ programming languages are also used. Of the total of about 600 programming languages in current use, we have noted at least 75 languages on various military applications including JAVA, which is expanding in use among all software classes.

It is interesting that when productivity comparisons are restricted to coding, and exclude production of paper documents, the defense community and the civilian systems software community are roughly equal in terms of productivity. In other words, the defense programming community is as good as other software domains in coding.

Military projects often share common features with civilian systems software projects. One of these features is a need for high quality and reliability coupled with rather sophisticated software quality assurance groups. The military software domain has the second highest levels of defect removal of any type of software that Software Productivity Research has studied. Many military software projects top 95 percent in defect removal efficiency, and some have approached 99 percent. Since the U.S. national average is only about 85 percent in terms of pre-deployment defects removed, the defense community has had better than average quality control.

This is true for weapons systems and communications systems, but not necessarily true for ordinary defense applications such as payrolls and accounting that do not follow military standards. The domains that lag in software quality control include information systems, commercial software vendors, and some but not all outsource vendors. Of course in every domain there are broad ranges of performance, just as there are broad ranges in every human activity.

The military domain also ranks as number two in the use of software quality assurance departments. On many defense projects, more than 30 percent of the total work force is involved with testing and quality assurance tasks. Quality control in the military domain for weapons systems is quite sophisticated for obvious reasons. The main reason is because military software controls complicate physical devices such as radar sets and aircraft flight controls. If these do not work as intended, lives and battles could be lost.

The importance of quality control and formal processes within the military software domain explains why more defense software producers can be found at or higher than Level 3 on the Software Engineering Institute's (SEI) Capability Maturity Model® (CMM®) than other domains. Since many companies that are SEI CMM Level 3 produce both military and civilian software, there is some overlap between the systems and military software companies.

The defense software community deserves credit for pioneering software process assessments and process improvement technologies. The impact of the SEI's CMM has benefited many major defense applications and is spreading rapidly among civilian software producers as well.

In our studies since 1994, large applications of the same nominal size, such as 10,000 function points, appear to have better productivity and quality levels when produced by organizations at or above CMM Level 3. For smaller applications of around 1,000 function points in size, the data is less definitive but still favors the higher CMM levels.

A number of fairly sophisticated software quality approaches are quite common in both the military and systems software domains. The quality assurance and control methods used by both systems and military software include the following:

- Formal design and code inspections.
- Quality estimation tools.
- Quality and defect removal targets for key projects.
- Quality Function Deployment.
- *Six sigma* quality targets.
- Complexity analysis tools.
- Automated defect tracking systems.
- Test library automation support.
- Automated change control tools.
- Trained testing specialists.
- Formal regression test suites.
- Full life-cycle quality measurements.
- The SEI's CMM.

Both the systems and the defense software domains also strive for excellence in project management disciplines. Software cost estimation and software milestone tracking are very detailed activities in the defense domain. Software project management is an area where the defense community may be superior to most civilian sectors.

The military software domain utilizes the following two techniques that are seldom encountered on civilian software projects:

- Independent verification and validation (IV&V).
- Independent testing by a third party.

The phrase *IV&V* implies using a third party or an external company to investigate whether all requirements have been met and whether the design and other documents meet all relevant military standards. The phrase *independent testing* refers to hiring a company other than the prime contractor on a military software project to conduct late stage testing after internal testing.

Both IV&V and independent testing add costs to military software projects that are not encountered on normal civilian projects. Whether or not these stages actually improved quality is ambiguous. It is true that military software defect removal is among the best of any kind of software project. However, it is no better than the defect removal found on civilian systems' software projects where IV&V and independent testing are not performed. Yet the military results are still better than those usually noted on information systems and commercial software applications, and on some outsource projects.

The overall results of the military software quality approaches have been generally successful. Indeed, only systems software and military software have approached or exceeded 99 percent in cumulative defect removal efficiency levels.

Large system development is inherently difficult and complicated. The defense software community often has a need for very large software systems that can approach or exceed 100,000 function points or 10,000,000 source code statements. At this large end of the spectrum, the defense community achieves better quality levels and more successful outcomes than any other domain except the best of the systems software producers. Productivity rates are fairly low, but failures and cancelled projects are low, too. Thus, the overall economic picture for building very large applications is not too bad in the defense sector. Indeed, for the largest applications beyond 100,000 function points, military software is an overall leader in terms of success and failure ratios.

It can be said that the strong emphasis in the military world on rigorous processes, complete specifications, and formal quality assurance controls produce projects that are fairly successful above 10,000 function points in size and even above 100,000 function points. These large software projects are expensive of course, but being able to complete such projects and have them work is a very difficult task. The success of the military software community on very large software applications is commendable.

Conclusions

Overall, the defense move toward civilian best practices is encouraging. However because this initiative only started in 1994 and required several years of research, the results may not be fully visible until sometime around 2005 or later. The reason for this is because applications in the 10,000 to 100,000 function point size ranges normally have development cycles approaching five calendar years. Thus, major defense applications using civilian best practices and standards are still under development and hence not yet studied and measured in terms of overall productivity and quality.

The following are some of the conclusions that we have reached from studying both civilian and defense software projects:

- Large applications above 10,000 function points or 1,000,000 source code statements require rigorous quality control and capable project management to be successful. Large applications that skimp on quality control and are careless with plans and estimates usually fail. If they do not fail, they will run late and exceed their budgets by notable amounts.
- The strong emphasis on quality control and project management disciplines associated with the SEI's CMM leads to a greater probability of successful completion than less formal processes for applications larger than 10,000 function points or 1,000,000 source code statements. Organizations at or above CMM Level 3 are more likely to be successful on large systems than those at Levels 1 or 2.
- For small applications below 1,000 function points or 100,000 source code statements, formal processes are not as significant as the experience of the development team. This is because teams are small so competence – or incompetence – of even one person tends to be visible and significant.
- For small applications below 1,000 function points or 100,000 source code statements, the level achieved on the CMM by the development team does not lead to major differences in successes or failure rates.

Overall, there are hundreds of ways to cause software projects to fail, and only a few ways to make them succeed. The highest odds of success will be found where capable teams use formal quality control and formal project management methods.

The military and defense community has been a pioneer in both quality control and project management methods. This appears to have paid off when building large software packages. Capable software personnel are in great demand everywhere so all domains are striving to select and keep good personnel.

The DoD's move to civilian best practices is encouraging and indicates a desire to improve software performance. Of course, quite a few civilian practices are of marginal value so one of the problems facing the defense community is to select practices that are truly best in terms of achieving high levels of quality, reliability, productivity, or other tangible factors.◆

References

1. Perry, William J. "DoD Policy on the Future of MILSPEC." CrossTalk Sept. 1994.
2. Sorensen, Reed. "Software Standards: Their Evolution and Current State." CrossTalk Dec. 1999.
3. Dempsey, Paul Stephen, et. al. Denver International Airport: Lessons Learned. McGraw-Hill. Mar. 1997.
4. Johnson, James, et. al. The Chaos Report. West Yarmouth, Mass.: The Standish Group, 2001.

About the Author

Capers Jones is chief scientist emeritus of Artemis Management Systems and Software Productivity Research Inc., Burlington, Mass. Jones is an international consultant on software management topics, a speaker, a seminar leader, and an author. He is also well known for his company's research programs into the following critical software issues: Software Quality: Survey of the State of the Art; Software Process Improvement: Survey of the State of the Art; Software Project Management: Survey of the State of the Art. Formerly, Jones was assistant director of programming technology at the ITT Programming Technology Center in Stratford, Conn. Before that he was at IBM for 12 years. He received the IBM General Product Division's outstanding contribution award for his work in software quality and productivity improvement methods.

Software Productivity Research Inc.
6 Lincoln Knoll Drive
Burlington, MA 01803
Phone: (781) 273-0140
Fax: (781) 273-5176
E-mail: cjones@spr.com