Reconsidering the Risk of COTS Computing

Eugene H. Spafford

DSSG-V Think Piece
Fall 1997

Abstract

In recent years, the military has made a concerted effort to use COTS (commercial off-the-shelf) computing solutions instead of customized approaches. The general rationale for this is to reduce costs, shorten the interval of time-to-design to time-to-deployment, and increase standardization.

However, observation of the application of these principles during the DSSG site visits, coupled with background research in the area, suggests that some underlying problems with use of COTS products may actually result in increased cost and decreased reliability — neither of which should be goals of military procurement.

This essay describes, in an informal manner, why excessive emphasis on the procurement and use of COTS computing solutions should be reconsidered and reevaluated. The references starting on page 11 can be consulted for additional background information, including some of the specific citations presented in this paper.

1 Introduction

There is continuing pressure upon the military to use prepackaged components when building solutions to technological needs. Prepackaged, or COTS — commercial off-the-shelf, components tend to be available more quickly than specialized solutions can be designed and coded. Often, COTS solutions can be obtained at less initial cost, and they may adhere to standards that enable them to better interface to other standardized components.

For these same reasons, this pressure to use commercial off-the-shelf components is no less acute in the area of hardware procurement. The computer industry has been growing and changing at an incredible pace, coupled with an incredible growth in computer technology. The result has been swift obsolescence of hardware/software solutions that were considered state-of-the-art only a few years before. As the military of the future (and present) is critically dependent on advanced technology, it is no surprise that DoD agencies seek to have the "latest and greatest" technology at the best price.

There are, however, many hidden problems associated using COTS products, and in particular, software. The most common problem is one of hidden costs. Although COTS software may be cheaper to acquire in the short-term, in the long-term, operations, maintenance, fixing security flaws, and other expenses may actually render the software more expensive than a custom-designed solution. COTS software is primarily developed for the commercial marketplace — a much different environment. As such, it is often designed to have simplistic operation, a very broad feature set, limited customization, and cursory testing. This is to enable the software to be used by a broad and unsophisticated user base, and allow a compressed time-of-development to time-of-market sequence.

Contrast this with typical military needs. Military- oriented software usually has a limited but well-defined feature set, requires great reliability, is not accessible for frequent upgrade or patching, operates in security-critical environments, is procured in small lots, and has heightened security concerns. Almost none of these characteristics are shared with common, off-the-shelf, software packages available in the commercial marketplace.

Historically, such objections have been dismissed as unimportant because COTS software has not been used in mission-critical environments. However, over the last few years, there has been a increasing trend to use COTS software as a base in mission-critical applications. For instance,we have observed the use of Netscape WWW browsers for mission planning packages and access to intelligence data, the use of Windows 95 and Windows NT operating systems as host systems for C^4I, and Microsoft Excel for logistics. Because the COTS products have been used without major incident in secondary roles, the military system designers and specifiers are now calling for their use in more dangerous situations — without adequate understanding of the risks.

2 A Fictional Scenario

Consider a contrived and somewhat melodramatic example to illustrate some of the potential dangers.

> It is the spring 1997. Thousands of college students from around the United States, many of whom are from other countries, are looking forward to the end of another semester. As one of the rites of spring they are sending resumes to computer companies throughout United States, seeking internships and permanent positions.

Many of these are directed to the Microsoft Corporation,[1] and many are hired.

Some of the students and new graduates are actually covert agents of a foreign power. That country does not have a significant indigenous information warfare capability. However, that has not stopped them from sending scores of political and religious zealots to study in the U.S., and then take jobs in sensitive positions. Some of those agents are hired into positions writing code for the Windows 98 operating system, to be released within another year. Working secretly they insert subroutines into the portions of the systems they're working on. These pieces of code are harmless when examined separately and have no visible effect on the operation of the software. Because of the lack of apparent effect of some small pieces of code amidst hundreds of thousands of other lines of undocumented code, no one questions their presence. Furthermore, because of the cursory testing involved in an effort to keep Windows 98 from becoming Windows 99, the production code is shipped in the summer of 1998 with most of these changes in place.

In early 1999, political tensions begin to rise in the Middle East. The United States responds by sending a carrier battle group and deploying additional aircraft and personnel to bases in the region. After carefully making plans, the intelligence service of one of the belligerents instructs an agent to make a posting to an Internet bulletin board. The posting describes a serious security flaw in the Windows 98 operating system. One of the covert agents of the country working at Microsoft is involved in creating a patch for the security flaw. The patch is widely distributed and is installed on systems around the world, including many systems in use by the U.S. military.

In early December many users of Windows 98 report unexplained crashes or odd behavior. These reports are scattered and infrequent, and they are not treated as high priority in the customer support service at Microsoft. Thus, the cause is still undetermined on December 24.

[1]Microsoft has been chosen as the example in this scenario because of its dominant position in the marketplace. This choice is not intended to imply that Microsoft is any worse than other vendors of COTS products with respect to the problems illustrated by this story.

At midnight on Christmas Eve (GMT) the security patch the previously hidden code activates. It interacts with the recent security patch to unleash several new computer viruses and worm programs that propagate over any available network link, and that add themselves to executable code on any attached storage media. The code also randomly scrambles file contents, deletes files, and disables commands. The viruses exploit previously unknown flaws in various versions of Windows and Windows NT, causing similar disruption and damage. Critical military systems based on Windows malfunction or halt. Frantic operators attempt to reload the systems from backup media, only to have them fail as well. Later, it is discovered that the only working backups are from before the installation of the emergency patch: every backup since that time has been corrupted with the changed code, and refuse to load, or immediately activates the destructive code when started.

Meanwhile, the army of the belligerent country moves across neighboring borders and initiates hostilities with the forces of several nations friendly to the U.S. Allied forces are unable to respond in a coordinated manner because significant portions of the command and control infrastructure have been disabled. There are no backup systems. Some of the systems appear to be unaffected but their internal data has been severely corrupted; there are no internal checks to determine if data has been altered. In one notable case, Navy cruise missiles destroy two schools and a retirement home because the map database used to program them has been altered to show the targets as enemy bunkers. In another case, Air Force planes bomb a battalion of friendly forces with devastating effects because their real-time targeting systems had confused representations of friend and foe. U.S. forces are forced to withdraw and turn off most of their computer systems. In addition to the humiliation in the world media as a result of the civilian and friendly force casualties, the U.S. also suffers severe political damage as a result of its inability to assist our allies

Over the course of the next several weeks and months, investigation is severely hampered by a lack of tools and expertise to analyze the infected software, and by a lack of unaffected backup systems. As time goes on, the military discovers that many of their very large and important database systems, such as the TRANSCOM databases, have been thoroughly corrupted and must be re-built from scratch to be accurate. Some data is lost forever. The loss in time, manpower, and dollars is staggering. Personnel at Microsoft disavow any responsibility and point to their shrink-wrap license. Bill Gates offers to provide a discounted update to fix the problems if the government promises to drop its antitrust lawsuit against his company.

Experts estimate that it will take a decade or more to recover from the economic and political losses. The FBI is unable to find evidence sufficient to identify the culprits working at Microsoft. Microsoft employees discover that they have no audit trails or logs going back far enough to identify who had access to the code in question, so they are unable to assist. The cover agents continue in their positions, waiting for the next opportunity.

This sounds like a piece of (bad) fiction – and it is. However, it is not entirely removed from the realm of possibility. Some of the particular reasons why this may be closer to reality than we might like are discussed in the next section.

3 Discussion of the Factors

3.1 Problems

The military is relying heavily upon commercial off-the-shelf computing solutions, and the providers of the solutions are not developing them to military grade standards. The civilian population is driven to have the latest and greatest software, and military is intent on having the cheapest software – and neither sector is sufficiently concerned with quality, safety, or security.

In particular, one of the big concerns with COTS software packages is the overall security of that software. COTS software is produced in commercial settings where security (in a typical military sense) is neither applied nor practical. Thus we have situations where front-line military units are using armaments procured from companies that are required to do criminal background checks on employees, conduct polygraph exams, and only hire only U.S. citizens. However, those same armaments may be targeted and controlled by software packages available mail-order in countries around the world and partially designed and written by foreign nationals serving as student interns at software companies. Those packages are then configured, assembled and integrated by personnel who have little training or who perhaps spend some of their spare time breaking into computer systems as a hobby. The incongruity is both astounding and frightening.

Not too long ago there was a case in the news where Microsoft had sold a large number of their systems to the government in the People's Republic of China. When the software was loaded and started by the Chinese, it displayed a message on the screen that defamed the Chinese government. Upon investigation it was found that the Chinese translation of the standard Microsoft screens had been performed by a company in Taiwan. Unknown to the people in Redmond, unauthorized messages and code had been included in the Microsoft products destined for Peking.

There have also been reports of viruses being distributed with commercial software packages. The very first macro virus for Word, the so-called "concept" virus, was distributed on a Microsoft-prepared CD-ROM of software for developers. The distribution of the CD-ROM was widespread and resulted in a significant global incidence of computer viruses. Personnel at Microsoft were unaware that the virus was present on their CD-ROM and were unable to identify the culprit who placed it there.

Another major area of concern has to do with the robustness of the software. Commercial software is usually built to sell at the lowest reasonable price to the largest number of people. Most people use software in situations that are not safety-critical. As such, commercial software may fail in unexpected circumstances or when operated out of expected bounds. Thus, in the usual case, commercial software may be expected to fail badly but consumers discovering serious problems are usually willing to wait to accept software patches or release of the next version. This conflicts with the need for software to be usable in safety-critical and often desperate circumstances such as might be experienced in a military setting.

There are literally thousands of examples in the literature of software that failed to operate outside of relatively narrow constraints envisioned by the programmers. In many of these cases the software may have even been custom-designed for the particular application at hand. Developing software, especially software for safety critical applications, is not a well understood art. Few COTS products are designed to operate in safety critical environments and the majority of programmers developing them have had no exposure to important software engineering fundamental concepts or safety engineering training.

Quality assurance is a third area where COTS software is lacking. Effective software testing, including stress-testing, bounds testing, and limit testing, is time-consuming and expensive process. In a commercial setting it is critical for businesses to produce and release software on very short cycle times: sometimes companies release software revisions as frequently as every six months. Extensive quality control and testing is incompatible with such schedules. Furthermore intensive software testing requires trained personnel, expensive hardware and software, and application of labor. All three of these lead to increase production costs and they are therefore unlikely to be commonly employed by most software vendors developing for the common consumer marketplace.

Consider the standard business model is to maintain a competitive advantage in the marketplace. To do this requires a constant stream of new features and products. Although these new features and products create new opportunities in the marketplace, they also create new opportunities for software flaws and errors. It is also likely that these new features have not been tested or even designed for the very large and complicated computing environment represented by the military. Commercial development is generally focused on small issues and short to medium-term research with high pay-off. There is little or no market advantage (and possibly a disadvantage) in expending resources researching solutions to problems not present in a significant fraction of the consumer population, e.g. the military.

One of the most common sources of software errors is lack of domain knowledge. This occurs when the individuals who design and program the computer system are insufficiently familiar with its intended application environment. Unless the requirements and specifications of the computing system to be designed are exceedingly specific and detailed, subtle inconsistencies and errors may appear in the finished product. There are many well-known examples of this phenomenon in the software engineering literature. For instance, programmers who designed the fly-by-wire navigation controls for the F-18 inadvertently failed to account for flights that might cross the equator: the result was a system that would cause the plane to flip upside down whenever crossing the equator in either direction. (This was found and corrected in the flight simulator prior to actual deployment.) The year 2000 problem can also be attributed to a lack of understanding about the eventual operational environment of the software under development — operation in the year 2000. The rigorous design and testing procedures necessary to expose the majority of such problems are simply beyond the capability and cost model of most COTS vendors.

In many settings where computer software is needed in a military environment it is only necessary for the operator to make use of a limited number of commands. For instance, in mission planning operations, there is no need for the operator to have access to a general-purpose text editor and formatter, a Java compiler, a financial spreadsheet system, and a screen saver. However machines that support most COTS software are general-purpose in nature, and may come equipped with this broad variety of programs by default. If the systems are no so equipped, it is a great temptation (and little effort) for the operators to obtain and install such software. This encourages a proliferation of (potentially) unnecessary software, perhaps leading to misuse of the systems, hazardous interaction of various subsystems, and spread of computer viruses and other malware. Furthermore, as these many applications proliferate, and as the basic COTS platform increases in complexity, the underlying hardware must be upgraded to support reasonable performance of the base applications. It also lends itself to increased cost of operation as these many additional software products require update, patches and maintenance over time.

Functionality may also be sacrificed in such situations. Special functions that would be of value to the war fighter may well not be present in typical COTS software. Thus, it is necessary either to develop alternatives (that may contain their own bugs and limitations) from existing commands, re-train operators to use less natural functions, and make do without the special functionality. This is especially critical as a consideration when dealing with real-time software. Few standard COTS software packages are appropriate for real-time applications where getting guaranteed response is required.

Use of COTS software also eliminates the advantage of having customized and sometimes classified software. By using a software base that is available to potential enemies the underlying systems are made more vulnerable. This is because potential enemies can obtain, study and develop attacks against that software well in advance of its deployment. Customized software, although not necessarily classified, is unlikely to be widely available and widely studied by all potential adversaries. This is an especially critical concern in the current and future geopolitical arena where we may find ourselves in opposition to almost any nation state.

For examples of this problem one only needs to look at any of the standard security discussions on the Internet or in computer publications. Over the last five years there has been a steady stream of reports of security problems and bugs in commercial operating systems, including Sun's Solaris, Hewlett-Packard's HP/UX, and Microsoft's Windows NT. Coincident with the bug reports have been repeated reports of computer system break-ins and attacks. The widespread availability of these systems has allowed individuals – many with malicious intent – to study the systems extensively and then attack them. Experts in the field have no reason to believe that all of the potential attacks have been found or published. Basing security-critical systems on platforms such as these may allow them to be attacked or exploited in the event of military hostilities.

3.2 Costs

The underlying illusion of cost savings by using COTS software comes about through a failure to perform appropriate cost-benefit analysis. The initial, up-front costs are almost always smaller when buying COTS products. However there are a number of hidden costs that make long-term operation more expensive. For example, to keep current with COTS products requires purchase, maintenance, and on-going support of more complex and complicated equipment than may be required for the software at the heart of the system. The lifetime cost of this equipment and software can be substantial.

Another hidden cost of using COTS products is that of user assistance and training. Although it may seem that using standardized software would result in savings by having uniform training materials and classes, if the standard software chosen is complex, buggy and prone to computer viruses or other failures, it may require more training and more active user support to effectively use the systems. These small incremental costs are often not observed and not well quantified by managers.

One last concern related to COTS occurs when the market is dominated by only one or two major software vendors, such as Microsoft. In such a market there is even less incentive for the vendor to spend extra resources on issues of quality, testing, or customization that are necessary in safety critical systems. Why bother? The consumer population has little effective choice, having been locked into a de-facto standard set by the vendor. It is also the case that the vendor may be less likely to accept requests for fixes, customizations, or enhancements from the military because they comprise such a small segment of the market. We have heard that this has already occurred with regard to Microsoft.

It is not to be denied that using prepackaged solutions can result in significant cost savings. What is important to understand however, is that those solutions must be evaluated for appropriateness and not solely for cost. Sacrificing some accuracy, safety, and precision for an up-front savings on cost may not be an appropriate trade-off. There are applications and environments where COTS software is appropriate. It is also the case that COTS software and hardware solutions are evolving quickly and may in the future present themselves as better alternatives. However, at the current time, it appears foolish to place short-term cost savings at a higher priority that overall quality and appropriateness for the task can.

3.3 Secondary effects

One consequence of the increased pressure to use COTS solutions has been the loss of both capability and infrastructure to produce specialized computing solutions to military needs. In the 1970s and '80s a number of firms developed custom software for command and control applications, information security, navigation, weapons design and targeting, and other military tasks. In the same companies or closely allied to them were firms that produced specialized hardware platforms to accommodate that software. This included embedded processors and equipment hardened against EMF effects. However, over the last dozen years, as the military has moved more towards a policy of buying COTS, the specialized companies have gone out of business. More to the point, the trained personnel who worked in these businesses have moved on to other fields or have retired. As a consequence, there has been almost no new generation of software engineers who have been trained and practiced in the techniques and technology necessary to produce highly reliable, hardened computing solutions to some future military needs.

4 Concluding Remarks

The situation described by the preceding can be addressed, but it must be done with care. Furthermore, it must be done soon, before most systems are converted to a COTS base that may be inappropriate.

To address this overall area set of problems will require careful study and a multi-pronged approach. The first, and perhaps most important part in any such effort will be to construct a realistic cost model for the procurement, installation, upgrade, maintenance, and use of COTS software. This model also needs to be coupled with a detailed and realistic risk assessment of the use of various products in their target environments. It is inappropriate to only considered the initial acquisition costs for a software and hardware combination without considering all of the long-term implications and their associated risks. To do so is to run the risk of incurring a much greater cost in later years, and possibly to find the application committed to a platform that is increasingly risky to use.

A second activity would be to identify areas of technology necessary to support specialized computing solutions. Some of those areas will be well supported in the current marketplace. For instance, there are some commercial capabilities for real-time design for factory automation, flight control, and medical systems. However, capabilities to produce a wider variety of systems that also include support for multi-level security, safety, and environmental hardening are currently in short supply (or nonexistent). It is unlikely that any significant commercial markets will develop for these technologies in the near-term. It is also unlikely that talented scientists and engineers will be drawn to work in these areas when they are also presented with attractive opportunities to work on problems supported by the marketplace. Therefore, it will be necessary to design appropriate incentives to encourage training and retention of at least a core competency in these areas, if for no other reason than to perform maintenance on existing systems.

References

There are literally thousands of references related to issues mentioned in this paper. The following are references that were consulted during one particular phase of research in this effort.

References

[DD97] Dorothy Denning and Peter Denning, editors. *Internet Besieged*. Addison Wesley, 1997.

[Def96] Defense Science Board. Report of the task force on information warfare - defense. Technical report, Defense Science Board, 1996.

[ea95] Jeffrey Z. Johnson et al. Risk management theory and pratice. Technical report, Trident Data Systems, 1995. Prepared for the USAF Information Warfare Center.

[oCIP97] President's Commission on Critical Infrastructure Protection. Critical foundations: Protecting america's infrastructures. Technical report, U.S. Government, 1997.

[Pow98] Richard Power. Current and future danger: A csi primer on computer crime and information warfare. Technical report, Computer Security Institute, 1998. Third edition.

[RDFZed] Virginia Rezmierski, Stephen Deering, Amy Fazio, and Scott Ziobro. Incident cost analysis and modeling. Technical report, Committee on Institutional Cooperation, 1998 (expected). (in preparation).

[Rho97] Keith Alan Rhodes. How i learned to stop worrying and love what the vendor told me. *Computer Security Journal*, XIII(2):63–68, 1997.

[Sch94] Winn Schwartau. *Information Warfare*. Thunder's Mouth Press, 1994.

[SSSC91] National Research Council System Security Study Committee. *Computers at Risk: Safe Computing in the Information Age*. National Academy Press, 1991.

[unk97a] unknown. Defense reform initiative: The business strategy for defense in the 21st century. Technical report, Office of the Secretary of Defense, 1997.

[unk97b] unknown. High-risk program: Information on selected high-risk areas. Technical Report GAO/HR-97-30, General Accounting Office, May 1997.

[unk97c] unknown. Improving information assurance. Technical report, OSD, Pentagon, 1997.

5 Acknowledgements

The author would like to acknowledge the time and assistance provided to him by the following people who provided time and background briefings for this report: Ken Geide and Mark Pollitt of the FBI; Susan Gragg of the CIA; Teresa Lunt and Hillary Orman at DARPA; Terry Hawkins, Gary Christoph, and Tony Burris of Los Alamos National Labs; John Davis of the PCCIP; Roger Callahan of the OSD; Julian Noll and Terry Mayfield of IDA; and some other individuals who asked not to be named in these acknowledgements. My thanks (and apologies) to anyone whose name I forgot to list. My special thanks to Bill Hurley and Nancy Licato for their support and patience in this entire effort.