Ethics in Military and Civilian Software Development

Sam Nitzberg

Abstract:

The quality of systems created for either civilian or military purposes has systemic, infrastructure-wide consequences. A number of the concomitant ethical considerations present in developing military and civilian software are similar, and are examined.

1 Introduction

In section one, *Introduction*, Civilian and defense work entails all the risks that knowledge workers face. There are personal hazards as well as responsibility for dissemination of information in their charge. In section two, *Nature of the Beast*, issues related to the consequence of flawed systems and their capability to produce harm are discussed. In section three, *Dual-use technologies and the Millennium Bug*, notions are presented exploring how the future moral or immoral use of systems may not be evident, as well as the future consequences of software flaws. Section four, *Professionalism*, discusses codes of conduct and other societal factors which impact software quality – that is, its safety and integrity. Section five, *Quagmire*, states that organizations must ensure the quality of their software to safeguard persons, property, and information. Section six, *Conclusion*, finds that ethical considerations can not be decoupled from software development.

A fundamental ethical concern pertinent to both military and civilian industry is fidelity in maintaining secrets. Adversaries do exist and it is often cheaper to steal technology than to develop it "in house." Those with expert or rarified knowledge may find themselves in physical jeopardy or face some sort of emotional coercion, testing mission allegiance.

Simulations are used to model traffic flow, stock market and business situations, and the effects of nuclear devices being designed. As the results of these simulations are realized and effected, their impacts are made very clear and real.

2 Nature of the Beast

Software product developers must face certain ethical challenges. Due care and expertise must be responsibly applied to ensure that systems perform their mandated functions properly, and that they do not corrupt vital information or produce wanton destruction of life or property. Although a great many applications being currently developed have theoretic foundations which date back decades, very often poor implementation and practices are used, resulting in buggy, if not dangerously flawed software products. Modern weapons systems are increasingly being built upon conventional software products, ill suited in many ways to the demands of the modern warfighting environment [Nitzberg, 1998].

The Patriot missile system which failed to successfully track the Iraqi Scud missile which killed twenty eight American soldiers during the gulf conflict may very well have failed to perform as desired due to a software problem [Littlewood, 1992]. An illustration of the real-life consequences of poor software development practices in civilian systems may be explored through the example of a computer-controlled x-ray medical diagnostic machine when two cancer patients died as a direct result of software error [Gotterbarn, 1998].

Errors which exist due to poor or erroneously documented requirements or specifications could allow such a system to dispense lethal doses of radiation through no actual malfunction of the unit itself. On the other hand, common errors in programmer code could compel the machine to apply lethal doses, as well. Similar problems exist in military systems. Flaws in systems requirements, specifications, or program code can have very severe effects - including mission failures. Erroneous requirements or program specifications can result in a wide range of failures, even resulting in fratricidal engagements, as can errors in the program code itself. Traditionally, software complexity has been viewed as the source of errors in systems. At least in theory, defect-free software can be produced [Littlewood, 1992].

3 Dual-Use Technologies and the Millennium Bug

Software systems work poses its own ethical dilemmas. There may be no distinction between military and civilian software components in weapons platforms. The software developer may not be able to choose or foresee the ultimate use of software developed; even in the case of military applications, the developer does not enjoy a role in determining the product's use. Even the nature and impact of poorly created software can be similar in both environs – with the potential to produce tragic and devastating loss and harm to civilian and defense interests.

Dual use technologies are hardware and software appliances to be used in ostensibly traditional commercial or "civilian" use but they may also be used to improve a nation's warfighting or intelligence capability. Advanced computer systems may be used for pharmaceuticals research, or may be applied in the development of nuclear weapons; GPS (Global Positioning Systems) technologies may be used to assist mountain climbers, or to land warheads on target. The Internet itself, a long-term result and ancestor of work performed under the auspices of the United States Defense Advanced Research Projects Agency (DARPA), may itself be viewed as a "dual-use" technology. The Global Positioning system (GPS) was developed by the US military in order to better effect its many missions. GPS can be used to effectively guide and provide navigational support to military vessels, combat aircraft, covert action teams, and missile systems. What was not predicted when GPS technologies were being developed was how GPS would eventually be incorporated into civilian aircraft systems, automobiles, personal GPS systems which plug into computers equipped with mapping software, and personal handheld GPS units for hikers or other recreational use. It is not always possible to work on technologies and understand what their ultimate use will be. After all, the birth of computing was nestled in military technologies.

Some perceive the costs associated with the "Millennium Bug" as being forbidding - "If you knew what the experts know, you'd be buying guns too" - and see the potential for a near total collapse of civilization [Poulsen, 1998]. Sam Nitzberg would refer to these proposed situations of failing power grids, communication systems, air traffic systems, and virtually all necessary computing functions as the "new nightfall" scenario. The President's Commission on Critical Infrastructure Protection has identified five sectors (Information and Communications; Banking and Finance; Energy, including Electrical Power, Oil, and Gas; Physical Distribution, and Vital Human Services) of critical importance to the well-being of the United States, and presumably to any first-world nation. As if fears of an impending social collapse is not a serious enough concern for ostensibly a simple programming problem with a simple cause, the military has a like view. According to a National Security Agency (NSA) representative, "The DOD’s Y2k conversion effort is a national security interest… All information detailing these information systems and the progress being made toward their conversions is considered to be highly sensitive." [Brewin, 1998]

The effectiveness with which these issues are addressed may significantly impact on nations’ security postures. In order to mitigate the risk of accidental nuclear exchanges related to Year 2,000 issues, the Pentagon will share traditionally restricted information with other, less Y2K prepared countries regarding the nature of American nuclear and missile early warning systems [Brewin, 1998].

On the small-scale, the developers of Y2K affected system understood the temporal constraints under which they would function, and that customers of complex systems were generally informed of the consequences and their systems specifications. In the large picture, with the multitudes of stand-alone and interacting systems, no one really knows precisely what will happen when the clocks do in fact roll-over. Considering how simply the Y2K problem could have been remedied on a system-by system basis at its source, this represents a grievous scientific, managerial, and moral failing.

4 Professionalism

The software industry is not very well known for its warrantees, but is much more famous for its legal disclaimers absolving software firms for any and all liability for its products. One such unfortunate and sweeping disclaimer promises that in no way will the specific software product meet any requirements or measurement of reliability, and that the only reason the disclaimer was even proffered was due to the insistence of the firm’s attorneys [Forester, 1994] Further, the user generally pays for updates. Regardless of the nature of a software package’s failings, the user ends up paying for more correct (less incorrect) versions. One common myth in computing is that there are no standards for producing software code. Quite to the contrary, there are a number of standards and methods not only for producing high-quality software products, but for software testing methodologies as well [Roetzheim, 1991; Freedman, 1990; Musa, 1987].

Codes of ethical conduct can be powerful facilitators in promoting professionalism and safeguarding society [Gotterbarn, 1997]. A desirable goal would be to define ethical standards and educational criteria, as well as through addressing domains of professional specialization [Bagert, 1999]. So doing should produce significant gains in the quality and efficacy of software, thereby protecting people and property interests. The licensing of software professionals is considered by some as a possible remedy to all poor practice and incompetence in the industry, and has been cited as a mechanism to answer the call to "protect us from their incompetence." [Gotterbarn, 1996]

Philosophically aired in the context of software developer mores, the differences between virtual and actual resist accommodation:

> "Ultimately, though, as professionals with particular roles and responsibilities, carrying out practical tasks the ramifications of which are often profoundly unclear, the sorts of guidance that many normative ethical theories provide us also depends on our social knowledge of what it is that we are doing, on our understandings of the possible impacts of the projects we undertake, and on our ability to integrate abstract ethical theories with the (apparently) more practical decisions of the workplace. Just as no code of ethics guarantees ethical behaviour, no normative ethics can compel assent or assure its own appropriate application." [Rooksby, 1998]

5 Quagmire

Organizations must take a very broad view of how their products are to be used, i.e., to consider harm which may result in the direct or indirect use of their products, and take all appropriate steps to ensure that policies, procedures, and methods are in place to address them. A common thread running through most of the issues which must be addressed to mitigate needless damage in either civilian or military environs appears to be the basic attempt for organizations to maintain some degree of foresight in how they develop and deploy their systems. Nearly every qualitative problem outlined above may be addressed by due care and the proper use of software development methodologies, plus a critical eye for detail.

While credentials may reflect a certain basic knowledge or sophistication, they do not necessarily demonstrate in any material way that a job candidate will produce quality work on any safety or mission - critical application. Critically important elements are often absent from software professionals' resumes, and seldom appear either as prerequisites to performing needed software engineering work, or as vitally important areas in which job holders are to be trained. Such oft-neglected areas include computing security, software testing, and understanding the consequences of "mission failures."

Software testing is considered by many in the computing industry as an annoyance, and a hurdle to overcome in developing a product, prior to its release. Professionals responsible for designing, implementing, and performing system tests may not have a background or familiarity in more advanced software testing methods, and the quality and accuracy of software tests may be seriously jeopardized as a result.

Almost all commercial and military applications are built to some degree upon "closed" operating systems or applications. There are traditional methods of security analysis which allow security decisions to be based on the likelihood of certain events being weighed, as well as their potential costs [Amoroso, 1994]. The use of such "closed" systems defy such analysis and are therefore used without any quantifiable estimation of the implicit risks.

6 Conclusion

Ethical concerns and considerations can not be decoupled from software analysis, planning, and execution. Most of the challenges facing either the current civilian or military software developer are not totally new. While the technologies and mechanisms specific to any particular application may be unique, what is of increasing importance and consequence is that software developers have an underlying humanistic philosophy and setting in which they perform their tasks. The fundamentally sticky problem in software development continues to be that developers must understand both the nature of their work and the consequences of their potential failures and take steps to ensure that their projects lead to safe and successful deployment.

7. References

Amoroso, E. (1994), Fundamentals of computer security technology, PTR Prentice Hall.

Bagert, D. (1999), Viewpoint: taking the lead in licensing software engineers, Communications of the Association of Computing Machinery,Vol. 42, No. 4, 27-29.

Brewin, B., Harreld, H., and Verton, D. (1998), NSA concerns could hamper DOD Y2K fix, Federal Computer Week, May, 1.

Brewin, B., and Harreld, H. (1998), U.S. to share Y2K nuclear data, Federal Computer Week, June, 1.

Forester, T., and Morrison, P. (1994), Computer ethics: cautionary tales and ethical dilemmas in computing, 2nd Ed., The MIT Press.

Freedman, D., and Weinberg, G. (1990), Handbook of walkthroughs, inspections, and technical reviews: evaluating programs, projects, and products, Dorset Publishing Co.

Gotterbarn, D., and Kizza, J. (ed) (1996), Computer practitioners: professionals or hired guns, The Social and Ethical Effects of the Computer Revolution, McFarland & Company, Inc.

Gotterbarn, D, and Miller, K., and Rogerson, S. (1997), Software engineering code of ethics, Communications of the Association of Computing Machinery, Vol. 40, No. 11, 110-118.

Gotterbarn, D. (1998), Informatics and professional responsibility, Reader in Ethical Computing and Business, Blackwell Ltd.

Littlewood, B., and Lorenzo, S. (1992), The risks of software, Scientific American, Vol 40, No. 11, 62.

Musa, J., Iannino, A., and Okumoto, K. (1987), Software reliability: measurement, prediction, application, McGraw-Hill.

Nitzberg, S. (1998), Improving computing security during the development of DOD computerized weapons platforms, National Information Systems Security Conference.

Poulsen, K. (1998), The Y2K solution: run for your life!!, Wired, August, Vol. 6, No. 8,Tagline from mailer page.

Roetzheim, W. (1991), Developing software to government standards, Prentice Hall, Inc.

Rooksby, E. (1998), Posting on computer-ethics@mailbase.ac.uk, October 29 10:20:27 PM.

8. Contact

Sam Nitzberg may be reached through his personal web page at: www.iamsam.com, and via his personal electronic mail address: sam@iamsam.com.