Software Safety - Ethics, Professionalism, and Legal Issues

A. J. Kornecki, Embry Riddle Aeronautical University, Daytona Beach, FL
B. W. Cunningham, Northrop Grumman Information Technology, Dahlgren, VA

Keywords: ethics, professionalism, software safety, licensing, malpractice

Abstract

Software safety is an essential aspect of safety-critical software. Included in the software safety discipline are many different processes and products. Developers and managers are mostly concerned with the technical and business aspects of the discipline. Risk and hazard analysis, risk mitigation, safety process, and safe design techniques and methods are the primary means to achieve a safe product. This paper, however, concentrates more on the human side of the software engineering profession addressing the issues of professionalism and ethics. These factors, which may contribute significantly to software and system safety, are often overlooked. Software engineers frequently make decisions regarding ethics and professionalism without realizing they do. These decisions can have serious impact on the safety of a system.

The paper describes various codes of conduct focusing on the Software Engineering Code of Ethics and Professional Practice - a document created by the IEEE/ACM Joint Task Force on Software Engineering Ethics and Professional Practice. Legal Issues are another aspect of the paper. In our litigious society lawsuits are very popular. In many cases they can be avoided or won if proper precautions and actions are taken. Software and system engineers working in safety critical systems must be aware of such issues. The paper explores the legal side of software engineering, including the different forms of software engineering legal suits and describing selected examples/case studies.

Introduction

The paper compiles facts collected during research on professionalism, ethics, and legal issues related to system and software safety. The information comes from previously published works and official documents (codes of ethics, legal trial documentation, case studies, etc.). Access to detailed information beyond the scope of this paper is available through the references.

Professionalism and ethics are evidently of value to system and software safety and are of great importance and interest to many software engineers. They often experience difficult situations requiring a sound viewpoint on these issues. It is sad to say that an informal survey of students engaged in software development shows that many employers might not pay enough attention to this aspect of the profession. As a breakdown in ethics and/or professionalism may have wider reaching implications, we have also been exploring related legal matters. Some of the issues of social and ethical consideration as well as the legal aspects of developing safety critical systems have been described in literature (ref.1).

Ethical concerns are often associated with the professional responsibility. The responsibility defines the sense in which one is accountable for achieving (or maintaining) a good result. The failure to be sufficiently careful in a matter in which one has responsibility may be a case of negligence. Certainly, the artifacts produced are a matter of contract between the parties involved, which bring to the forefront the issue of legal obligation, i.e. liability.

The paper begins with a look at software engineering as a profession, from the perspective of software intensive safety critical systems. We shall discuss the code of ethics and professionalism available in general engineering and specifically designed for the software profession. We then explore legal issues with software engineering, which includes the different forms of software engineering legal suits as well as some examples/case studies. The issues presented lead to a discussion on the relation of these topics to system/software safety.

Software Engineering as a Profession

Professional engineering emerged to meet requirements for public safety. Building bridges, airplanes and tall buildings requires not only knowledge of physics and mathematics but also practical application of these scientific principles in a disciplined manner. Software engineering is a relatively new discipline. The term itself was introduced slightly over twenty years ago (in comparison to civil or mechanical engineering reaching over a century). The software crisis where many large projects failed miserably due to lack of discipline, planning, and organization was the proverbial straw breaking the camel's back. Over the last decade software engineering has made significant progress while facing a variety of challenges due to the rapid growth of the computing discipline and proliferation of software in all facets of modern society.

There are two aspects of individual professional assessment: certification and licensing. Certification is a voluntary process administered by an appropriate professional society or organization. IEEE Computer Society, which has been active in developing software engineering standards for more than twenty years, currently offers the Certified Software Development Professional designation, as specified on the IEEE Computer Society website http://www.computer.org/certification/ with three critical components:

- exam-based testing demonstrating mastery of a Body of Knowledge (BOK),
- extensive experience base in the performance of the work or profession being certified,
- continuing professional education, measured and relevant to the BOK.

Licensing is a legal, often mandatory process designated to protect the public. This is why typically legal authorities of a specific locality (state, city) administer licensing. The specifics of the licensing examination are being coordinated with a respective professional organization.

We should also note that it might be unlawful to use the title of engineer. Kaner (ref.2) gives as an example Section 6732 of the California Business and Professions Code, which requires an individual to be registered with the State in order to be called an engineer. To answer the question "Is software engineering technically a profession?" Kaner lists features, which commonly characterize a profession:

1. *the requirement of extensive learning and training*
2. *a code of ethics imposing standards above those normally tolerated in the marketplace*
3. *a disciplinary system for members who breach the code*
4. *a primary emphasis on social responsibility over strictly individual gain, and the corresponding duty of its members to behave as members of a disciplined and honorable profession; and*
5. *the prerequisite of a license prior to admission to practice*

The first point is met since in most circumstances, being a software engineer does require extensive training. A degree in science or engineering with a focus on computing is typically a condition for hiring in most software development organizations (web hacking notwithstanding). The second requirement is also met since the code of ethics does exist, as we shall discuss in the subsequent section of the paper. The third and fourth points are less convincing: there is currently

no disciplinary system in place and an emphasis on social responsibility is important, but not necessarily followed by each software engineer due to lack of appropriate training. The most important point is the licensing requirements. In the state of California over thirty professions require a license to engage in the occupational activities. They range from mule jockeys and guide dog instructors, to barbers and locksmiths, to architects and professional engineers. No license is required to be a software engineer and this is usually the point that acquits the defendant in computer malpractice cases.

Table 1- Examples of Professional Licensing Requirements (from Nitzberg, ref. 3)

Profession	**Duties Include/May Include**	**Licensing Required?**
Barbers / Hairdressers	Cutting and styling of hair Applying of dyes to hair Use of proper hygiene	Generally required
Manicurists	Cutting and trimming of nails. Grooming of nails Maintaining proper sanitization of equipment	Generally required
Civil Engineers	Design and construction of bridges, roadways, industrial buildings and complexes, military complexes and transit systems, dams, etc…	Mandatory
Prostitutes	"Personal Services" Regular medical testing. "Safe" practices.	Mandatory where lawful e.g. The Netherlands, United States (Nevada, outside the Las Vegas City Limits)
Software Engineers	Design and construction of software components of: o Medical diagnostic equipment, o Medical dosing systems o Air Traffic Control Systems, o Strategic nuclear weapon control systems, o Command and Control Systems, o Anti-aircraft/missile systems o Aircraft fly-by-wire systems o Automotive computer control systems o Banking and financial systems	No requirement

Table 1 shows an amazing comparison. Software engineers who are responsible for millions of lines of code controlling extremely safety-critical items such as medical equipment and space craft have no requirement for licensing, whereas those working in general hygiene do. This is a very intriguing point. Perhaps the future will hold licensing requirements for those of us in the software engineering field, specifically those engaged in safety critical software development.

Software Engineering Code of Ethics and Professional Practice

All professional organizations abide by some well-accepted ethical norms. Several of the organizations have gone through an additional effort to verbalize them and establish a specific

code of ethics in a quest to give the specific industry easy to consult material on the issue. Such codes of ethics provide additional guidance for the workforce.

National Science Foundation support established the Online Ethics Center for Engineering and Science available at http://www.onlineethics.org/. The mission of the Center, located at Case Western Reserve University is to provide engineers with resources useful for addressing ethically significant problems that arise in their work. The site contains many interesting links that may serve as the starting point for exploration of the subject.

Many organizations have produced very structured standards for ethics and professionalism. A British Centre for Computing and Social Responsibility contains an exhaustive list of codes of ethics at http://www.ccsr.cse.dmu.ac.uk/resources/professionalism/codes/. The major entries related to North-American software professionals are:

- The National Society of Professional Engineers (NSPE) "Code of Ethics for Engineers"
- The Association for Computing Machinery/Institute of Electrical and Electronics Engineers (ACM/IEEE) "Software Engineering Code of Ethics"
- The Association for Computing Machinery (ACM) "Code of Ethics and Professional Conduct with Guidelines"
- The Institute for the Management of Information Systems (IMIS) "Code of Ethics"
- The Canadian Information Processing Society "The Code of Ethics and Standards of Conduct"

In addition to the above, the site provides links to codes related to specific disciplines such as mechanical and medical. Many national organizations have created their own codes (e.g. Great Britain, Australia, Hong Kong, Germany). Some industrial organizations dealing with safety critical software also publish their internal codes of conduct. One should not forget the Conference Sponsor: the System Safety Society Constitution's By-Laws include Fundamental Canons of Ethics (Article III) with associated guidelines (Article IV). Seven sections specify activities and codes of conduct related to system safety within context of greater than ever role of technology (http://www.system-safety.org/AboutGenInfoBylaws.htm).

We focus here on two examples. NSPE site http://onlineethics.org/codes/NSPEcode.html provides link to the Code of Ethics for Engineers, which gives generic guidelines, acceptable by all engineers. In the preamble the NSPE stresses the impact of engineers on public quality of life. The document mandates engineers to adhere to the highest principles of ethical conduct, honesty, impartiality, and fairness. The base of the document is six "Fundamental Canons":

Engineers, in the fulfillment of their professional duties, shall:

1. *Hold paramount the safety, health and welfare of the public.*
2. *Perform services only in areas of their competence.*
3. *Issue public statements only in an objective and truthful manner.*
4. *Act for each employer or client as faithful agents or trustees.*
5. *Avoid deceptive acts.*
6. *Conduct themselves honorably, responsibly, ethically and lawfully so as to enhance the honor, reputation and usefulness of the profession.*

The NSPE document also elaborates on rules of practice and professional obligations providing more specific guidelines on expected conduct.

Of particular interest to software engineers is the "Software Engineering Code of Ethics and Professional Practice" (SECEPP) that can be found at http://www.acm.org/serving/se/code.htm

\

Association for Computing Machinery web site. It is a document created by the IEEE/ACM Joint Task Force on Software Engineering Ethics and Professional Practice.

The eight basic principles, referred to by the IEEE/ACM as *"Principles that software engineers should commit themselves to following and practicing"* constitute the document's core. They were created to cover major aspects of software engineering ethics and professionalism. Quoting directly from the document, they are related to:

1. *Public - Software engineers shall act consistently with the public interest.*
2. *Client and Employer - Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.*
3. *Product - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.*
4. *Judgment - Software engineers shall maintain integrity and independence in their professional judgment.*
5. *Management - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.*
6. *Profession - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.*
7. *Colleagues - Software engineers shall be fair to and supportive of their colleagues.*
8. *Self - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.*

These eight statements take on a very broad scope, containing something for everybody. Although presented first as a simple list, the document elaborates on each of these principles in detail. Under each principle is a list of rules or guidelines to be followed by software engineers. A few of the rules and guidelines are discussed below.

The development practices and theory should be in line with the public interest. Perhaps the employer, interested in the bottom line, requests the developer to cut corners to assist in saving resources. If such a request may jeopardize the safety of the general public, the responsibility of the developer is to stand up and object to those unethical requests.

The software engineer should always act in the interest of both client and employer. This could raise interesting points of divided loyalties. But the information the developer presents to both must be honest and forthright. If a software engineer discovers a critical error late in the project, which would induce some risk into the system, they should report it to their client and/or employer even though it will most likely increase cost and push back the delivery date.

Developers should strive to make the products meet *"the highest professional standards possible."* Certainly it leaves room for tradeoffs and environmental constraints. Sometimes cost or time constraints prohibit creating a product with the highest possible quality. Nevertheless, software engineers should *"Strive for high quality, acceptable cost, and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public."*

It is important for software engineers to be honest and impartial in their judgment as they work on a project. They need to look not only at technical aspects when making decisions, but also consider human values and safety issues. The document is very clear on the need to be honest, impartial, and objective in all tasks: *"Maintain professional objectivity with respect to any software or related documents they are asked to evaluate."*

\

The management of software engineering projects should promote ethical practices in their projects and lead by example. The management must make all company and industry policies and standards known to their team. Another point unique to management is the issue of honesty in any estimates that they make for their client or employer *to "ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work, and provide an uncertainty assessment of these estimates."* The project manager must also act as a fair and impartial judge in any disputes or issues with employees.

As we discussed earlier, over the last ten years software engineering has been striving to establish itself as an accepted profession. It is required that software engineers act in a way, which upholds the integrity of the practice and promotes and improves its reputation. This aspect seems to be often neglected by software engineers who often come from ranks of computer science graduates with good technical skills, but discipline leaving a lot to be desired. Software engineers should *"Promote public knowledge of software engineering."* This is an interesting point and does not seem to have much to do with ethics, but could fall under the category of professionalism. Software engineers should *"Be accurate in stating the characteristics of software on which they work, avoiding not only false claims, but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful."* This is especially important when dealing with user's manuals and license agreements. We should avoid deceptive practices or trying to mislead the client.

Software engineering is a team engagement. As such the developers must be helpful, encouraging and supportive of their colleagues in the workplace and the profession. This includes things such as giving colleagues the credit for their work, assistance in adhering to standard work practices, and following security measures.

Finally, in the light of a rapid changes and fascinating growth of the software engineering body of knowledge, it is an ethical obligation of each professional to continue improving their knowledge and grow as a software engineer.

Issues with Ethics and Professionalism

The Software Engineering Code of Ethics is a good document, but does it really work on an actual project? We attempt to consider few issues that might come into play in real life situations, pointing out possible problems with the implementation of the code of ethics.

Management Involvement: It is extremely important that the managers and the project leaders be in full support of the code of ethics and professional practice. A team of software engineers almost always will follow the example set by their person in charge. With this in mind, the manager of the project really has significant influence to set the tone for those working under him/her and the project as a whole. If the manager has a poor attitude or is generally apathetic with regards to quality, safety, or project deadlines this will also be the attitude of his/her employees. When this type of attitude is present, it begins to be hard to focus and work on a project and the quality, and in turn safety, will most likely be compromised. However, if a project manager decides to adopt and truly endorse a code of conduct, the productivity and quality will most likely increase. Without management buy-in, the value of a code of ethics and professional practice is greatly reduced because the full, team-based potential cannot be realized.

Stress and Deadlines: Stress and deadlines can have a major effect on how a code of ethics is used. According to some estimates schedule pressure affects 75% of medium size and 90% of

large-scale projects. It always seems like things start falling apart as the project gets closer to the specified deadlines. This is usually true of both quality and of the use of coding standards, commenting, and following certain guidelines. While a code of ethics is a great and useful thing to have, it will almost always fall into the "nice to have" category. The practice shows that it will be one of the first things to be overlooked when the heat is on.

The Chain of Command: The chain of command on a project or in an organization can seriously affect the use of the code of ethics and professional practice. Within a chain of command there are generally a few different levels of authority. For example, one chain of command going from the lowest to highest authority might be: embedded developer responsible for implementation, software engineer responsible for design, project manager, system engineer, and vice president of the company. In this example the person of lower authority will report directly to their immediate supervisor, which would be the next person up on the list. This situation may get tricky when a code of ethics and professional practice is involved. Let us suppose that we are in the middle a very complex project getting close to the end of the prearranged time and money for the project. The embedded developer finds a significant error in one of the modules. According to the code of ethics the developer should document this error and report it to his supervisor, employer and client. This report should contain a full explanation of what will happen if the problem will not be fixed and should also contain an analysis of how the cost and time will be affected. This could be bitter information to swallow, almost surely shaking things up all the way to the top of the chain of command. This will make the developer look bad to the project manager, and the project manager look bad to the system engineer and vice president. We could see some organizations where the embedded developer would be urged to keep quiet and move on to avoid any problems with the client and upper management. The easiest way to deal with a problem is to ignore it. However, an organization, which is truly concerned with quality and safety, will embrace the code of ethics and professional practice and reward the developer for finding the error and ensure that the proper steps are taken to document and correct it. We sincerely believe that in most organizations working in safety critical industries this will be the only acceptable way of proceeding.

Legal Issues in Software Engineering

No matter what we do in life, we are always responsible for our actions. This is true also in our professions. As members of a software development team we are responsible for the actions of our creation, which in most circumstances is a software package or code, embedded in a hardware system.

Lawsuits are prevalent in all aspects of today's society. With this fact in mind, we as software engineers must always be able to defend what we are doing or making. There are three basic types of lawsuits that may be encountered in the software world. They are *Breach of Contract*, *Negligence*, and *Malpractice*.

The breach of contract lawsuit is a situation where the product or service agreed upon failed to be delivered. For example, if the contract says that an organization's data will be backed up and protected, the responsibility of the contractor is to meet these conditions. If the contractor fails to protect the data it is liable under a breach of contract clause.

A negligence lawsuit is more complex than a breach of contract lawsuit and is more difficult to prove. In a negligence case, the legal team will compare the behavior of the allegedly negligent party to what a reasonable person would have done under the same circumstances. Using the previous example of data backup, to sue for negligence they the client must prove that the steps

the contractor took to protect the data were unreasonable in comparison to what anybody else would have done. A concept of accepted practices is the one to be used.

Kaner (ref.4) uses the following example to explain negligence as related to inadequate software testing. A failure of program leads to fatality. The program failed when reaching a specific line of code that was never tested. A tool that allows the tester to determine which lines of code have not yet been tested, was available but not used. The following arguments provided by the plaintiff's lawyer might thus produce a winning argument: if the product had been tested to a level of complete coverage (all lines tested), this defect would be found, and the victim would be alive. The negligence of the software company can be shown. Such argument could not be made if a tool was not available.

A computer malpractice lawsuit can be filed against a company for a variety of reasons. According to Daich (ref.5), this may even include improper documentation. He states the following: *"Yesterday's state-of-the-practice is often today's malpractice. Let me go on record as saying that organizations not implementing a disciplined document review program will one day be considered to be conducting software malpractice."*

Computer malpractice suits are rather hard to prove. Malpractice suits compare your behavior to what a reasonable member of your profession would have done. These professional standards are usually very high and should be well documented. This poses an interesting question. Is software engineering technically a profession?

Malpractice insurance is available for software developers and companies. It is difficult for a software company to take responsibility for almost limitless potential damages the software could cause. From a lawyer's perspective, as cited from Grossman (ref.6), the insurance is a *"must have"* for any developer or company that can afford it. This insurance is also sometimes referred to as an *"E and O policy"* (Errors and Omissions Policy). These policies can come in different forms with different premiums and can cover such infractions as breach of contract, negligence, fraud, and copyright infringement.

Legal Examples

Therac-25 is the best-known case of safety violation attributed to software. Leveson exhaustively described the case in an appendix to her seminal text (ref.7). This medical therapy machine released in 1980s was responsible for administering radiation treatments to patients. The great innovation was that it was one of the first machines to rely on software control, which made setup simpler and allowed the doctors more time to speak with the patients. The design was based on an earlier model and many of the hardware controls were directly converted to software. The previous (hardware controlled) model had already met FDA (Food and Drug Administration) requirements and was only required to have minimal testing. A problem occurred when the software team focused testing only on wear and tear errors instead of actual software flow testing. With safety in mind they decided to use Fault Tree Analysis to make sure the system was safe and functioning properly. However, the actual functionality of the new software control structure was never adequately tested. The result of this design/testing error was multiple radiation overdoses, two of which ended in fatality. Of course the families of the surviving and deceased patients filed lawsuits immediately. All of the lawsuits were settled out of court. This is a grim, but illustrative example of how software engineers can be liable. Safety is the primary objective and software developers must be prepared to face the consequences of their actions, in particular when human life is at stake.

On a slightly less dreadful note, we bring the case of Chatlos Systems vs. National Cash Register Corp (1979) (ref.2). A salesperson recommended a particular NCR solution after performing a detailed business analysis of the client's system. Based on this advice of the NCR representative, Chatlos Systems bought an NCR system, which never provided its promised functionality. Chatlos Systems then sued NCR for malpractice. The court decided that there was no case because of the lack of a standard for computer malpractice. However, the verdict was passed as a breach of contract.

To show that similar conditions may not always result in the same outcome, we present the case of Invacare Corp. vs. Sperry Corp (ref.2). The setup of this case was similar to the previous one. Invacare had relied on the advice of Sperry Corp. and purchased a system, which did not performed as needed. The court reverted to the Chatlos decision and stated that it could not be computer malpractice. However, Invacare was able to prove that no reasonable professional would have given such a recommendation. Sperry Corp was found guilty of ordinary, but not professional negligence.

In a well-publicized case Sun Microsystems sued Microsoft Corp. (ref.8) for breach of contract stating that Microsoft distributed a version of Sun's Java programming language that was not compatible with theirs. The contract clearly stated that Microsoft was not allowed to manipulate the language in such a manner. This case never actually went to court. Microsoft agreed to settle out of court, paying Sun $20 million. This is important to note because even though the breach of contract lawsuit is the easiest to prove, it can still be very costly. Another software giant, AOL (America On Line), was sued for criminal negligence (ref.9). The suit, alleging a deceptive trade practice and violation of computer-tampering laws, seeks damages of up to $1,000 for each of the approximately 8 million people who have downloaded version 5.0 of AOL. When installed, the other software on a user's computer was often disabled, altered, or interfered with. It has been apparent that AOL knew about that problem all along, but elected not to fix it. This seems to be a clear-cut and expensive case of negligence.

Relation of Ethics and Professional Practice to Safety

Ethics and professionalism are closely tied software safety. It is evident that ethics, professionalism, and adherence to law are required to ensure the safety of the public. Any bending of the rules in ethics and professionalism may compromise the safety of a system and thus the public. Safety conscious organizations should require and encourage adherence to a code of ethics. It should also foster an environment that makes it easy for an employee to use, promote, and question the code of ethics. Safety Critical development teams, more than anyone else, should know their legal responsibilities when developing software given the severity of the situation they are in. A safety critical system often may be responsible for the life or death of the user, and that is a very important responsibility to take on.

When safety is a concern in a system, many regulations will be present. These regulations and guidelines need to be considered since they help to give assurance that the equipment, methodology, etc. is standardized and approved. The attitude of software engineers and project managers should be no different. With an effective code of ethics and professional practice in place a project is more likely to be successful because the guidelines will form a more serious and sound approach toward the project's success.

Conclusion

Software engineering professionalism, ethics, and legality are very important and interesting topics. They are applicable in a wide variety of environments and projects and are particularly essential in a safety critical environment. A code of ethics should be presented as early as possible in the training/education of software engineers. The earlier the better: that way it may become second nature to software engineers in their practice. It is important to know where the individual stands on the issue of software ethics and professionalism and to encourage others to think about it as well. And it is also important to be aware of the legal ramifications of software development. In our litigious society lawsuits are very common. However, they often can be avoided or won if proper precautions and actions are taken. The concerns of ethics and professionalism in relation to software safety contain intriguing information. With the proliferation of software in all aspects of life and the increasing role of embedded computing, it will be interesting to see how these concerns evolve in the software engineering profession, particularly the legal side of it.

References

1. N.Storey, Safety-Critical Computer Systems, Addison Wesley, 1996
2. C.Kaner, Computer Malpractice, Software QA, Volume 3, #4, p. 23, 1996, http://www.kaner.com/malprac.htm
3. S. Nitzberg, Software Engineering and Ethics, seminar presentation Iona College, NY, November 2002, http://www.iamsam.com
4. C.Kaner, Software Negligence and Testing Coverage, 1996, http://www.kaner.com/coverage.htm
5. G.T.Daich, Document Diseases and Software Malpractice, Crosstalk, Nov 2002, http://stsc.hill.af.mil/crosstalk/2002/11/daich.html
6. M.Grossman, Malpractice Insurance for Software Developers, Computer Law Tip of the Week, 1999, http://www.mgrossmanlaw.com/articles/1999/malpractice_insurance_for_softwa.htm
7. N.Leveson, Safeware - System Safety and Computers, Addison Wesley, 1995
8. J.Niccolai, Sun, Microsoft Settle Java Lawsuit, IDG News Service, 2001, http://www.nwfusion.com/news/2001/0123msjava.html
9. P.Kent, Software Companies Guilty of Criminal Negligence for Failing to Correct Problems, the Boulder County Business Report, 2000. http://www.bcbr.com/mar2400/geek2.htm

Biography

Andrew J. Kornecki received MSEE'70 and PhD'74 degrees from the University of Mining and Metallurgy in Krakow, Poland. After teaching and doing research on three continents, currently he is employed as a faculty at Embry Riddle Aeronautical University (ERAU), Daytona Beach, FL. He has been teaching a variety of undergraduate and graduate courses: computer organization, modeling and simulation, real-time systems, performance analysis, software safety. Recently he has been engaged in research on testing and certification issues and assessment of development tools for safety critical real-time systems.

Bradley W. Cunningham received B.Sc. in Aviation Computer Science and Master of Software Engineering degrees in 2000 and 2002 respectively from Embry Riddle Aeronautical University. Currently he is employed as a software safety engineer at Northrop Grumman Information Technology Defense Mission Systems.

\