Professional Engineers
Ontario

GUIDELINE

Developing Software for Safety Critical Engineering Applications

Contributors: Roger Yiu Ming Cheung, P.Eng. / Jeffrey Coulson, P.Eng. / Eugen Malea, P.Eng. / Corneliu Muntean, P.Eng. / Nick Pfeiffer, P.Eng., Ph.D. (Chair) / Anton Pop, P.Eng. / Paul Spagnolo, P.Eng. / Pak Tse, P.Eng.

November 2013

Contents

1. PEO Purpose for Guidelines

Professional Engineers Ontario (PEO) produces guidelines for the purpose of educating both licensees and the public about best practices.

For more information on PEO's guideline and development process, which includes PEO's standard form for proposing revisions to guidelines, please read our document: http://peo.on.ca/index.php/ci_id/23427/la_id/1.htm.

For a complete list of PEO's guidelines, please visit Appendix 4.

To view other PEO guidelines, please visit the Publications section of the PEO website: http://peo.on.ca/index.php/ci_id/1834/la_id/1.htm.

2. Preface

The Professional Standards Committee formed a subcommittee of engineers from a variety of practice areas who had experience developing software in their professional engineering practice. The group was asked to investigate the legal, ethical and technical aspects of software design and development that could have an impact on the public interest. Furthermore, the subcommittee was instructed to prepare a guideline to deal with the development of software when it is considered to fall within the scope of professional engineering.

The subcommittee met for the first time on July 21, 2010, and submitted a completed draft of this document to the Professional Standards Committee for approval on January 15, 2013.

Following consultations with engineers and other stakeholders, the final draft was approved by Council at its meeting on November 22, 2013.

Note: References in this guideline to engineers apply equally to professional engineers, temporary licence holders, provisional licence holders and limited licence holders.

Practitioners as defined in the *Professional Engineers Act* (Act) refers to engineers and firms holding a Certificate of Authorization to offer and provide engineering services to the public.

For the purposes of this guideline, the term the *public interest* refers to the safeguarding of life, health, property, economic interests, the public welfare and the environment.

3. Purpose and Scope of Guideline

Software may pose a risk to the public interest, either directly or indirectly. The purpose of this document is to outline the ethical and professional responsibilities of engineers to ensure that the public interest is protected. This document also provides guidance for others interfacing with engineers who are developing software, such as clients and owners who are acquiring ready-made software or specifying requirements for new software. This guideline should be regarded as an addition to but not a substitute for specialized software training. The guideline is written with the expectation that the reader is familiar with software development.

The development of certain categories of software is considered to fall within the scope of professional engineering in Ontario when the software is used in a manner that affects the public interest. In June 2008, PEO council approved the following definition of Software Engineering:

Software engineering is deemed to fall within the practice of professional engineering as defined by the Professional Engineers Act:

- *Where the software is used in a product that already falls within the practice of engineering (e.g. elevator controls, nuclear reactor controls, medical equipment such as gamma-ray cameras, etc.); and*
- *Where the use of the software poses a risk to life, health, property or the public welfare; and*
- *Where the design or analysis requires the application of engineering principles within the software (e.g. does engineering calculations), meets a requirement of engineering practice (e.g. a fail-safe system), or requires the application of the principles of engineering in its development.*

For the purpose of this guideline, software that meets all three criteria of the above definition is considered to be safety critical software and falls within the practice of professional engineering, even though the term "safety critical software" may be defined differently by other organizations.

This guideline recommends considerations for developing safety critical software and for incorporating such software as part of larger systems. It addresses factors that are reasonably necessary for the protection of the public interest, such as: software requirements, software design and construction, software process engineering, software quality, software assets management, and management of software projects. It is important to note that the recommendations presented here must be tailored to the specific requirements of each project.

This guideline does not deal with the role and responsibilities of engineers who apply engineering software to perform calculations, modeling and optimization analysis as part of professional engineering services or to provide information used as the basis for professional engineering decisions, judgments and opinions. The application or use of such software is the subject of a separate guideline, entitled *Professional Engineers Using Software-Based Engineering Tools*.

For the remainder of this document, software and software development activities are assumed to refer solely to safety critical software unless explicitly noted otherwise.

Since the development of safety critical software falls within the practice of professional engineering, only engineers or those supervised by an engineer can develop safety critical software. Furthermore, practitioners who develop safety critical software as part of a service to the public are required to have a Certificate of Authorization.

This guideline does not cover software that falls outside the practice of professional engineering, even if engineers often develop software that is not related to professional engineering (e.g. business, financial, or tax software; e-commerce software; database analysis software; and gaming software).

4. Introduction

Modern machines are becoming increasing sophisticated and complex as a means of decreasing costs, removing variability, reducing resources and increasing production. By definition, many of these devices contain safety critical software. Poor design, failure to check functionality, improper use and failure to properly maintain this software can potentially lead to physical injury or death, economic damages, environmental impact or the loss of public trust. Within this context, it is important to note that engineers are always professionally responsible for their work, including the design and development of safety critical software.

Engineers play a key role in designing and developing safety critical software. The intent of this guideline is to provide best practices for engineers developing such software, to set expectations with respect to engineering practices in the domain of safety critical software processes, to provide guidance within the realm of safety critical software to meet the intent of the Act, and to help improve trust in the use of safety critical software. Engineers need to be aware of potential risks and of their ethical and professional responsibilities to protect the public interest.

Safety critical software can have many applications, including but not limited to:

- control and data acquisition;
- sensing and interpretation software, and modeling and design software that is used to make or automate critical decisions and actions that impact the public interest, such as utility (telecommunications, water, electricity, gas, traffic) control and protection;
- public transportation control systems and industrial safety;
- protection and control systems software; and
- medical and diagnostic equipment.

It is the responsibility of the sealing engineer using third-party software to validate results obtained from the software before implementing them into the system.

There is a need to provide proactive means to prevent software system failure that may affect the public interest. This guideline focuses on the professional responsibilities of engineers developing safety critical software or incorporating such software as part of a larger system or product. It clarifies the role of engineers, as well as their duty to protect the public interest in this area.

Many other organizations have developed best practices, guidelines and applicable standards, some of which are listed in Appendix 1. Engineers are advised to direct themselves to these other references, as required. It should be noted that this guideline is neither a standard nor a body of knowledge, but rather a collection of best practices. It provides recommendations–not prescriptive rules–and does not explain theoretical or practical knowledge.

It is recognized that engineers are most often part of a product or software development team that includes others, such as information systems professionals (ISPs), computer scientists and technologists. This document reviews the responsibilities of engineers to ensure the public interest is protected by:

- recognizing professional and ethical responsibilities with software development, especially public interest considerations;
- accepting professional responsibilities in product delivery (i.e. final review and sealing);
- delineating responsibilities for multi-disciplinary projects (i.e. hardware and software interfaces); and
- recognizing the professional responsibilities of engineers in different software development roles and during various stages of software development.

The development of safety critical software requires the same degree of review and validation as the development of any safety system, structure or device that falls within the practice of professional engineering. There is a misconception that software does not carry the same level of importance as drawings, hardware or systems. This mistaken belief is rather perilous, when one considers that software can be easily modified, unlike a structure or device. Because of this fact, safety critical software requires an enhanced level of attention to functionality, documentation and version control (see 5.6 Support). This guideline identifies the responsibilities of engineers in this area, so safety critical software development is undertaken with the level of care and diligence that is required of all engineering activities covered by the Act.

All engineers are professionally responsible for the engineering work they produce. Article 72(2)(b) of O. Reg. 941/90 under the Act identifies one criterion of professional misconduct as "failure to make reasonable provision for the safeguarding of life, health or property of a person who may be affected by the work for which the practitioner is responsible". Engineers must be aware that the concept of "reasonable provision" applies to the development of safety critical software, since it falls under the practice of engineering.

5. Professional Responsibility

5.1 Public Interest

Developing safety critical software is a complex undertaking, comprising different processes, including specifying requirements, design, implementation, testing, verification and validation. Furthermore, interpreting the needs of clients and consumers, balancing budget and schedule constraints, and ensuring the efficiency, effectiveness, integrity, security, privacy, safety, and quality of the software are all activities that involve a degree of risk. These facts should remind engineers of their responsibility for performing due diligence and protecting the public interest, since, ultimately, engineers contribute to the success of software projects.

5.2 Risk Management

To perform due diligence, project risks need to be identified early, analyzed, treated according to the likelihood and impact of the risk, and managed. Risk treatment may include avoiding a risk entirely; transferring (or sharing) the risk with another party; mitigating the risk, either its likelihood and/or its impact; or accepting the risk. In choosing appropriate risk treatments, particular attention needs to be paid to the public interest. Engineers are reminded of the importance of adopting well-recognized software engineering processes to manage any risk to the public interest (for some examples of these processes, refer to Appendix 1). Furthermore, engineers should be aware of relevant software system failures in the past, and be cognizant of their obligations when working with systems of a similar nature (refer to Appendix 2 for case studies of software system failures). Finally, engineers should design safety critical software so that it can handle the threat of malicious software.

Safety critical software shall be sealed to provide assurance that engineers responsible for developing the software have fulfilled their obligations under the Act. The seal provides traceability in case the engineer responsible for developing the software needs to be contacted.

5.3 Human Factors

Human error has played a significant role in several catastrophic system failures throughout the world. Consequently, engineers need to be aware that human error prevention is of paramount importance in designing and deploying safety critical software. Design of human inter-

faces required to operate or maintain the system should account for human capabilities and limitations, in addition to meeting the necessary obligatory requirements of pertinent regulations.

5.4 Code of Ethics

Engineers are reminded of PEO's Code of Ethics, which states that: "A Practitioner shall…regard the Practitioner's duty to public welfare as paramount". Hence, practitioners should minimize risk to the public interest through use of a well-recognized software development process, with system safety considerations as the foremost element in the design.

The Code of Ethics also states: "It is the duty of a Practitioner…to act at all times with knowledge of developments in the area of professional engineering relevant to any services that are undertaken". This provides an obligation for engineers to have knowledge of well-recognized software engineering processes, as well as similar safety critical software systems, including potential modes of failure (Appendix 2 contains case studies of software system failures).

For more information on the Code of Ethics, please refer to the *Professional Engineering Practice* guideline.

5.5 Legal Obligations and Confidential Information

As this is an area where conflict can arise, it is important to document intellectual property rights and ownership, as well as client relationships, properly and get legal advice when needed. Intellectual property rights and ownership includes copyright, patents, industrial design rights, "trade secrets" and trademarks. In addition to legal obligations, engineers also have important ethical and professional obligations relating to confidential information. Following are excerpts from the *Professional Engineering Practice* guideline, which discusses confidential information in detail:

- *Engineers should not divulge any information sensitive to their clients' or employers' business to third parties, unless expressly or implicitly authorized by their clients or employers or required by law to do so.*
- *Engineers are also expected to avoid using (confidential) information for the benefit of themselves or third parties, or to their clients' or other practitioners' disadvantage. Engineers are expected to decline employment or a commission that would require disclosure of such information.*
- *It is generally considered that engineers may apply any general knowledge or expertise, as long as it falls into a "state of the art" category.*

5.6 Support

Software risk is further managed when the software is adequately supported throughout its lifecycle, from requirements definition to production, maintenance and deployment. Such support may include:

- Documentation, such as requirements and specifications, verification and validation reports, manuals, critical function/alarm restoration/preservation processes, safety assessment reports, software version control process, software defect tracking process;
- statements of conformance to applicable standards and of any limitations or restrictions;
- training, where required by agreement, on installation, maintenance, and operation;
- access to source code under suitable contractual terms; and
- expressed warranty, liability limitations (e.g. connecting to third-party middleware).

5.7 Multi-practitioner Projects

Final documents related to safety critical software provided as a service to the public shall be sealed by the engineer responsible for developing the software. In cases where the software is developed by multiple engineers, for example in a large or multi-disciplinary project, each engineer may seal the portion of the documentation for which he or she is responsible. Such sealing shall provide assurance to the engineer responsible for the overall development of the software that the sealed portion of the safety critical software has been developed in accordance with the Act. It is not sufficient that each portion of safety critical software is sealed; the overall software should be sealed by the engineer taking professional responsibility for the work.

Engineers who seal safety critical software shall ensure that their obligations under the Act are fulfilled, including taking responsibility for portions of the software that have been developed by others under their supervision. The guideline *Use of the Professional Engineer's Seal* elaborates on sealing multiple-discipline projects as follows: "For a project covering work within several engineering disciplines, all documents within a particular engineering discipline must be sealed by the engineer taking responsibility for work within

that discipline, with an indication or qualification of which discipline is implied by the seal. The supervisory or coordinating engineer (if there is one) should also apply his or her seal to indicate that the work of the various disciplines has been coordinated. If only one signature and seal is used, it should be that of the engineer taking responsibility for the work, generally the coordinating engineer."

Engineers who use safety critical software as part of a larger project or system shall ensure that the software is fit for use in the particular application, and operating and physical environment. Such fitness for use may include an understanding of functionality, reliability, usability, security, limitations, underlying principles, design constraints, and validity/reliability of results.

6. Intellectual Property

Intellectual property (IP) refers to a variety of intangible assets, such as copyrights, trademarks, patents, and trade secrets. Owners of these assets may grant legal rights restricting use. When the asset is software-related, such as the source code for a program or the details of an algorithm, the granted rights may limit the conditions under which the software can be used, maintained, or included with other software.

Modern software development is often accomplished by combining original software modules with pre-existing/reusable software modules. The end result is a new software product with unique characteristics and functionality. In the process of software development, engineers might act as either users of IP, or authors of IP, or even both users and authors within the same project.

When using third-party software or development tools, engineers should acknowledge and respect the IP rights granted or limited by licences, warranties, redistribution statements, and disclaimers. The IP rights and their implications regarding safety, maintenance, upgrades and use by customers or other parties should be an important consideration for any engineer. In particular, too restrictive third-party IP rights might limit the possibility of comprehensive evaluation of safety-related features for a software module intended to be included into a newly developed software product (e.g. restricted access to data constants covered by third-party IP limitations).

Engineers as authors and/or owners of IP should take necessary actions to protect their rights. These rights are extremely important when the IP relates to safety critical software. Engineers should evaluate the potential IP generated by software development and identify technical limitations in relation to the IP. For example, IP could cover technically strategic algorithms, specialized calculations, data constants obtained through extensive empirical research, or the actual source code.

Engineers should seek legal advice on IP matters when needed. Due to the complexity of IP rights, engineers, either as users or as authors/owners of IP, should initiate a dialogue with trained professionals in IP laws. Some of the legal clarifications may cover, but are not limited to: the rights granted or limited by copyright, licence, warranty, redistribution statements and disclaimers, in order to limit liabilities and avoid creating a risk to the public interest.

More general legal aspects of engineering work in regard to safety critical software is presented in section 5.5 Legal.

7. Sealing

The *Use of the Professional Engineer's Seal* guideline states: "Engineers must seal all final documents that are within the practice of professional engineering, provided as a service to the public." This requirement is contained in the Act. Consequently, all final safety critical software packages provided as a service to the public, including code and professional documents, shall be sealed in accordance with this guideline.

The basic purpose of sealing hardcopy or softcopy professional documents (refer to section 7.2 for more information on professional documents), or a software package, is to identify the work has been performed by or under the supervision of an engineer. The product of engineering work is sealed to indicate that other people can rely on it to be suitable for its intended purpose.

7.1 Safety Critical Software Package

7.1.1 Originally developed safety critical software package

The original version or modifications of a safety critical software package, including code and documents, shall be sealed. The safety critical software package may be sealed on the equivalent of the "cover page" or introduction to that software.

7.1.2 Review of third-party software

Often, software is developed for one purpose, but may be used for another. For example, a commercially available

graphical user interface developed for a non-critical information kiosk application may be used in a critical control application. If such software becomes safety critical software by its incorporation into a different purpose, engineers responsible for the system, process, equipment or machine shall ensure that their obligations under the Act are fulfilled. For instance, developers of safety critical software should ensure that re-use or integration of such software with other products is properly documented, including any limitations or constraints. Where this is not the case, engineers must review and verify that the software functions as anticipated for the intended application.

An engineer's obligations, including his or her duty to the public welfare and providing proper credit for engineering work, can be fulfilled by a thorough review that includes sufficient research, calculations and other professional engineering work, so that the engineer is satisfied that the work meets appropriate codes and standards and that due diligence is exercised. A review does not necessarily imply a complete rework. The test that should be applied is: "Does the work meet the acceptable professional and regulatory standards?" not "Is this the way that I would have performed the work?". Engineers should create documents detailing the review process and seal them.

7.2 Professional Documents

Professional documents associated with but separate from software code for safety critical software, either in electronic or hard copy form, shall be sealed. This may include, but is not limited to, the following design, testing and commissioning documents:

- design documents, requirements, specifications, including documentation of operating environment (compiler, software versions, etc.);
- physical models of monitoring and control systems (i.e. process drawings and mechanical drawings);
- artifacts produced during the course of design and development (i.e. tradeoff analysis, prototypes, analysis elements, software defect tracking, and verification and validation test reports);
- documentation on the purpose and appropriate use of software, including limitations and constraints and re-use or integration with other products, development and maintenance documents, tools, and aids; and
- review documents for third-party software, as described in section 7.1.2.

The appearance of an engineer's seal on a professional document is taken as an indication of the engineer's acceptance of professional responsibility for design, development and testing. These principles apply to both paper documents and electronic documents. Refer to the guideline *Use of the Professional Engineers Seal* for more detailed information.

7.3 Revisions

When a safety critical software package undergoes a revision, the engineer responsible for the revision shall label and seal the software code and professional documents that are part of that safety critical software package release. The seal indicates the engineer's acceptance of responsibility for the revised package and associated engineering work. Care should be taken in documenting the revisions to clearly identify the boundary of professional responsibility between the original and revised software code and documents.

8. Software Development Methods and Procedures

There is no best single process for developing software. However, many methodologies and standards have been created relating to software development to control the process and reduce the variability. Appendix 1 contains a list of well-recognized processes for software development.

Regardless of the particular methodology used, the development of safety critical software is a project that comprises many steps and can be managed by established techniques.

Engineers should choose a software development methodology that is appropriate for the type of software to be developed. Whatever development methodology is chosen, there should be a clear documentation of why it was chosen and the benefits and risks of choosing it.

This guideline does not specify a particular development methodology. However, engineers should choose a reliable methodology and put controls/processes in place that minimize any inherent risks.

In general, software development projects can be divided into the following major tasks:

- requirements;
- design and development;
- process engineering;

- software quality;
- assets management;
- project management; and
- software packaging.

Engineers are often significantly involved in all these different tasks.

8.1 Software Requirements

Software requirements are drawn out using processes and tools for identification of the scope and specifications, the end users' needs, the re-use or integration of existing products, the code or applications, and the constraints. These requirements are then used to describe and document the functional and non functional aspects of the software. The requirements are validated through prototyping, review and modeling. Finally, engineers need to identify potential failure modes in safety critical software as part of their duty to make reasonable provision for the safeguarding of life, health or property of any person who may be affected by the work for which they are responsible.

8.2 Software Design and Development

Software design uses various techniques and tools to make and capture decisions as to how the software will be built. It covers architecture, design, notations, user interface, security and safety, data engineering, and performance engineering.

Once a design has been defined, the software is developed through programming and integration of software components. This stage would also include the creation of development documentation, user documentation and training materials.

8.3 Software Process Engineering

Proven methodologies to develop software include the definition, measurement, and improvement of a software engineering process to create a repeatable, predictable development method or life cycle. There are a number of process models that formalize the task of developing software or use project management techniques to better control the development process.

8.4 Software Quality

During development, it is necessary to ensure that the software product adheres to applicable standards, conforms to its requirements, and meets its end-user needs. Software quality can be assessed through a number of techniques, including verification, validation, measurement, reviews and audits. Testing can be used to verify proper operation and validate whether the software fulfills its intended purpose. This testing can be performed at different stages of the development process, including unit or component testing, integration testing between components, system testing, and system integration testing for systems of two or more different systems. However, in general, testing can never completely identify all the defects within software. In addition, validation is subjective and contextualized based on the reviewer. Therefore, review and participation by the client and/or end-user testing may be required for acceptance of the final product. This can include software safety assessments, simulation, review of sample input and output data, performance testing, acceptance testing, qualification and certification.

8.5 Software Assets Management

Software must be managed through its life cycle. Software asset management is used to control and safeguard the software product versions, as well as the software development artifacts. Software asset management requires identification of all software elements, their version and history, their relationships, and archival data. Change control is implemented for identification and tracking of changes to software elements, including problem reports, contract changes and source code changes. Release management and delivery is the preparation of software for release, distribution, installation and deployment into operation. Deployment represents the activities that make the software available for use and is affected by the target physical environment, architectural and design constraints, and security and performance concerns.

8.6 Management of Software Projects

Software development needs to be effectively managed, just like any other engineering project, to measure and control the progress of the software project. Software project management uses planning and scheduling to define the types of process lifecycles, work breakdown structure, and estimations of workload. The management includes but is not limited to project tracking and metrics of progress, quality, and expenditure. In addition to its initial development, software needs to be maintained over its life cycle and one important aspect is having a process for defects management and tracking.

8.7 Software Packaging

Delivery of the software will require its packaging or application in an electronic format that allows the user, whether a client or another project group or team, to install, understand or use the software. In addition to the software instruction and training materials, the applicable professional documents should be transmitted to the user. This documentation and information can be transmitted electronically or through traditional means. In either case, all engineering documents and safety critical software must be sealed.

9. Summary

Throughout their careers engineers may be involved in developing safety critical software. The failure of safety critical software systems may pose risks to the public interest. Engineers must be aware of these risks and of their ethical and professional responsibilities to protect the public. Appendix 2 contains some case studies of software system failures.

Several other organizations have created well-recognized methodologies relating to the development of safety critical software, some of which are listed in Appendix 1. Engineers are advised to direct themselves to these references and other well-recognized methodologies, as required.

10. Definitions

For the purposes of this guideline the following terms and definitions apply.

Fail-safe system
In the event of a predictable system failure, the system is returned to a safe condition that minimizes risk to the public interest.

Reasonable provision
Requirement that practitioners maintain the standards that a reasonable and prudent practitioner would maintain under the circumstances

Safety critical software documentation
Documents that express a professional opinion, judgment or direction that someone else may rely upon; both hard copies and electronic copies of design documents, professional documents, requirements, specifications, including documentation of an operating environment (compiler, software versions, etc.); testing specifications; test procedures for critical components of software interfaces; interpretation of test results; implementation procedures/guides; user guides; and reports or other documents that express engineering work as contemplated in the *Professional Engineers Act* (sections 1 and 12) or Regulation 941/90 (section 53), or reproductions of same

Software safety assessment
Assessment providing an objective independent safety evaluation of the software in the overall system; a software safety assessment ensures that software engineers/designers have considered safety aspects in their designs (for example fault tree analysis, operational health and safety analysis, etc.)

Software risks
Risks to the public interest caused by software system failure

State of the art
Highest level of development achieved at a particular time

Trade secret
Information including, but not limited to, a formula, pattern, compilation, program, method, technique or process, or information contained or embodied in a product, device or mechanism that:

(i) is or may be used in trade or business;
(ii) is not generally known in that trade or business;
(iii) has economic value from not being generally known; and
(iv) is the subject of efforts that are reasonable under the circumstances to maintain its secrecy

Verification
The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase [taken from IEEE-STD-610]. Verification ensures that the product was built according to the design requirements and specifications. Verification ensures that "you built it right".

Validation
The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements [taken from IEEE-STD-610]. Validation ensures that the product actually meets the user's needs and fulfills intended use and goals. Validation ensures that "you built the right thing".

Appendix 1.–Software Engineering References of Interest to Engineers

This list does not in any way limit the responsibility of an engineer or the scope of this guideline.

Reference	Website
Associations	
Association for Computing Machinery (ACM)	http://www.acm.org/
Control Systems Integrators Association (CSIA)	http://csia.connectedcommunity.org
IEEE Computer Society	http://www.computer.org/portal/web/guest/home
Information Systems Audit and Control Association (ISACA)	https://www.isaca.org/Pages/default.aspx
Books	
Software Engineering by Ian Sommerville	http://www.softwareengineering-9.com/
Code of Ethics	
Software Engineering Code of Ethics (IEEE Computer Society and ACM)	http://www.computer.org/portal/web/certification/resources/code_of_ethics
Guidelines	
Guide to the Software Engineering Body of Knowledge (IEEE Computer Society)	http://www.swebok.org
Papers	
Comparison between five models of Software Engineering (IJCSI)	http://www.ijcsi.org/papers/7-5-94-101.pdf
Software Process Engineering	
Capability Maturity Model Integration (Carnegie Mellon–Software Engineering Institute)	http://cmmiinstitute.com/
Standards	
ANSI UL 1998, *Standard for Software in Programmable Components* (ANSI/UL)	http://www.ul.com/global/eng/pages/solutions/standards/accessstandards/catalogofstandards/standard/?id=1998_2
C22.2 NO. 0.8-12–*Safety functions incorporating electronic technology* (CSA)	http://shop.csa.ca/en/canada/canadian-electrical-code-part-ii-general-requirements/c222-no-08-12/invt/27007812012
N286.7.1-09–*Guideline for the application of N286.7-99, Quality assurance of analytical, scientific, and design computer programs for nuclear power plants* (CSA)	http://shop.csa.ca/en/canada/nuclear/n28671-09/invt/27030082009
N290.14-07–*Qualification of Pre-Developed Software for Use in Safety-Related Instrumentation and Control Applications in Nuclear Power Plants* (CSA)	http://shop.csa.ca/en/canada/nuclear/n29014-07-r2012/invt/27026862007
N290.4-11–*Requirements for reactor control systems of nuclear power plants* (CSA)	http://shop.csa.ca/en/canada/nuclear/n2904-11/invt/27009402011
Protecting against Common Cause Failures in Digital I&C Systems of Nuclear Power Plants (IAEA)	http://www-pub.iaea.org/MTCD/publications/PDF/Pub1410_web.pdf
Software for Computer Based Systems Important to Safety in Nuclear Power Plants (IAEA)	http://www-pub.iaea.org/mtcd/publications/pdf/pub1095_scr.pdf
Nuclear power plants–Instrumentation and control systems important to safety IEC 60880 (IEC)	http://webstore.iec.ch/webstore/webstore.nsf/Artnum_PK/36058

Functional Safety and IEC 61508 (IEC)	http://www.iec.ch/functionalsafety/
Software and Systems Engineering Standards (IEEE)	http://standards.ieee.org/findstds/standard/software_and_systems_engineering.html
Software and Systems Engineering Standards (ISO/IEC JTC1/SC7)	http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=45086&published=on
Software Assurance Standards (NASA)	http://www.hq.nasa.gov/office/codeq/software/docs.htm
Military Standard 498. *Software Development and Documentation* (Mil Std)	http://www.abelia.com/498pdf/498-STD.PDF
Military Standard 1629A. *Procedures for Performing a Failure Mode, Effects and Criticality Analysis* (Mil Std)	http://sre.org/pubs/Mil-Std-1629A.pdf
Software Considerations in Airborne Systems and Equipment Certification DO-178B (RTCA)	http://www.rtca.org/store_product.asp?prodid=581

Appendix 2. Case Studies of Software System Failures

Several high-profile failures illustrate the requirement for formal software engineering. These case studies do not in any way limit the responsibility of an engineer or the scope of this guideline.

Therac-25

Engineers should design safety critical software that either adapts to changes in hardware or flags changes in hardware.

In its original configuration, the Therac medical radiation treatment machine would not function unless a protective shield was in the correct position. The machine had a flawless treatment record despite the danger posed by the large dose of ionizing radiation it was capable of producing. However, the software on the new machine was supposed to incorporate this safeguard, but instead a combination of faulty sensors on the shield, a slow response time to operator inputs and inadequate feedback to the operator led to at least six accidental overexposures, *three of which were fatal*.

Northeast power blackout in 2003

Engineers should ensure that safety critical software is adequately monitored.

The electrical blackout affected an estimated 10 million people in Ontario and 45 million people in eight U.S. states. While this event was due to a combination of factors, one of the systems that failed was a computerized system that should have raised an alarm when loads on many of one company's lines started to be exceeded. Instead, the alarm stayed silent and the operators remained unaware of the problems. On older systems, each line would have had its own control and alarm systems.

Appendix 3. Amendment and Revision Submission Form

Guideline:

Statement of proposed amendment or revision:

Reason:

Submitted by: _____ Date: _____

Mail: Professional Engineers Ontario
101-40 Sheppard Avenue West
Toronto ON M2N 6K9

Attention: Standards and Guidelines Coordinator

Fax: (416) 224-1579 or (800) 268-0496

Email: practice-standards@peo.on.ca

Appendix 4. PEO Professional Practice Guidelines and Standards

Practice Guidelines

1. Acoustical Engineering Services in Land-Use Planning (1998)
2. Acting as Contract Employees (2001)
3. Acting as Independent Contractors (2001)
4. Acting under the Drainage Act (1988)
5. Building Projects Using Manufacturer-Designed Systems & Components (1999)
6. Commissioning Work in Buildings (1992)
7. Communications Services (1993)
8. Developing Software for Safety Critical Engineering Applications (2013)
9. Engineering Services to Municipalities (1986)
10. Environmental Site Assessment, Remediation and Management (1996)
11. General Review of Construction as Required by the Ontario Building Code (2008)
12. Geotechnical Engineering Services (1993)
13. Human Rights in Professional Practice (2009)
14. Land Development/Redevelopment Engineering Services (1994)
15. Mechanical and Electrical Engineering Services in Buildings (1997)
16. Professional Engineer as an Expert Witness (2011)
17. Professional Engineering Practice (2012)
18. Professional Engineer's Duty to Report (1991)
19. Project Management Services (1991)
20. Reports for Pre-Start Health and Safety Reviews (2001)
21. Reports on Mineral Properties (2002)
22. Reviewing Work Prepared by Another Professional Engineer (2011)
23. Roads, Bridges and Associated Facilities (1995)
24. Selection of Engineering Services (1998)
25. Services for Demolition of Buildings and Other Structures (2011)
26. Solid Waste Management (1993)
27. Structural Engineering Services in Buildings (1995)
28. Temporary Works (1993)
29. Transportation and Traffic Engineering (1994)
30. Use of Agreements between Client and Engineer for Professional Engineering Services (including sample agreement) (2000)
31. Use of the Professional Engineer's Seal (2008)
32. Using Software-Based Engineering Tools (2011)

Performance Standards

1. General Review of Construction of a Building (2008)
2. General Review of Demolition and Demolition Plans (2008)

Professional Engineers
Ontario

40 Sheppard Avenue West, Suite 101
Toronto, ON M2N 6K9

Tel: 416 224-1100 or 1 800 339-3716
Fax: 416 224-1579 or 1 800 268-0496

Enforcement Hotline: 416 224-9528 Ext. 1444

Website: www.peo.on.ca

Published by the Association of Professional Engineers of Ontario