Software Safety Standards: Evolution and Lessons Learned[†]

Shou-Yu Lee, W. Eric Wong, Ruizhi Gao
Department of Computer Science
University of Texas at Dallas
Richardson, Texas, USA
{sxl128630, ewong, gxr116020}@utdallas.edu

Abstract—As safety issues occur in many domains, software safety standards provide guidelines for development of software systems that operate in safety-critical environments. However, evolution of existing software safety standards diverges under various circumstances and environments. To understand the purpose of these standards on their domains and the effect of changing the environment on evolution of these standards, we conducted a survey on the history of the families of DO-178 (Commercial avionics), MIL-STD-882 (US Department of Defense), and DEF-STAN 00-56 (UK Ministry of Defense). Additionally, we learned that even in different environments, there are certain features in common that are preferred by industry and would likely be added to newer versions of the standard. In other words, these features are very likely to be must-haves when constructing new standards in the future.

Keywords-software safety; system safety; safety standard; safety-critical software; hazards

I. INTRODUCTION

As software is becoming a critical part of many vital environments such as the transportation, nuclear energy, defenses, and aeronautics industries, erroneous or faulty design can be disastrous, potentially resulting in not only immense financial loss but also human casualties [52][53]. However, rapidly growing software technology also makes the system architecture more bulky and complicated, thus bring about inflated software development efforts and system maintenance budgets. As a result, software safety standards are established to provide guidelines to meet the requirements in constructing safe and reliable systems and ease the workload of software engineers within the development process while following the system constraints. In other words, once these software safety standards are properly obeyed by the developer, the risk of disaster can be reduced.

In this paper, the evolution history of three popular standards are reviewed, especially their advantages in distinct backgrounds and the effect of changing requirements and new technologies on their evolution. In section II of this paper, DO-178 [42], a standard that is commonly used in various commercial avionics industries, will be introduced in detail. In section III, the properties of DEF STAN 00-56 [11], a standard constructed by the U.S. Department of Defense, will be presented. MIL-STD-882 [31], which was developed by the U. K. Ministry of Defense, will be shown in section IV. In section V, we will evaluate the evolution history of these standards in a more comprehensive way and try to identify similar properties in developing such standards. Finally, our conclusion and future work will be given in Section VI.

II. EVOLUTION OF DO-178

First created in 1980, DO-178 [42] was the first software safety standard for the avionics industry [28]. It was created to establish a basis for software certification approval by identifying and documenting the software development best practices known at the time. From its inception to its evolution through DO-178A [43], DO-178B [44], and DO-178C [45], the standard has faced various challenges and made significant changes to its requirements and recommendations.

DO-178 [42] introduced the idea that the rigor applied to software development and safety assurance of a system could vary by the criticality of the system [22]. In DO-178, a system could be classified as critical, essential, and non-essential. Furthermore, DO-178 established the need for a safety certification plan, which would include software requirements. However, DO-178 was written in a highly conceptual manner, and projects achieved compliance by meeting the "intent" of the standard [38]. It failed to provide specific guidance regarding the methods, processes, and techniques for the development of safety-critical software.

Published in 1985, DO-178A intended to incorporate the lessons learned and experiences gained from the use of DO-178 in the avionics industry. It was a significant departure from the previous version, DO-178. Unlike the vague guidelines found in the earlier DO-178, DO-178A planned to establish specific techniques and methods for the creation of safety-critical software [22].

DO-178A [43] introduced the use of specific software integrity levels based on the criticality and intended application of the system. It included more structured development and verification activities and varied the level of effort required for the different software levels [22][38].

However, DO-178A was weak in several areas. Diagrams and examples were often misinterpreted, and items necessary for certification were often contended. The required level of effort was also in contention. The reasons behind the requirements for certification were not understood or appreciated [22].

[†] This paper appeared in the Proceedings of the First International Conference on Trustworthy Systems and Their Applications (TSA 2014), June 2014.

By the late 1980s, the avionics industry was also making rapid advances in technology. Most avionics systems were much more complex, and the industry was transitioning from the use of analog to digital systems [22]. DO-178A was unable to keep up with these increasing demands.

In 1989, the RTCA convened a new committee, SC-167, to address these issues and update DO-178A to provide further guidance in developing safety-critical software [27]. SC-167 intended to address significant safety-related shortcomings for avionics software found in DO-178A [21]. The committee focused on five key areas for improvement: (1) document integration and production, (2) system issues, (3) software development, (4) software verification, (5) software configuration management and software quality assurance [24]. DO-178B, published in 1992, provided extensive guidance on these topics and was a major update to DO-178A.

In DO-178B [44], the failure condition categories were changed from non-essential, essential, and critical, to no effect, minor, major, hazardous, and catastrophic. This also caused the number of software levels to change from 3 to 5. Traceability was also added as a major aspect of software development, and various software development and verification activities were clarified [38].

Although it has become the de-facto standard for avionics software, DO-178B has nevertheless also come with some problems. The DO-178B process model progresses from requirements to design and code to integration and test in a linear fashion, much like the software development “waterfall” model [48]. The distinction between the requirements and software development process has often caused problems with a lack of discrimination between low-level requirements and design. Furthermore, DO-178B focuses primarily on top-down testing and does not stress the importance of testing early in the software development process. As a result, beneficial techniques such as static analysis and formal testing are not addressed [48]. The qualification of development tools is also difficult under DO-178B, since there is no guidance on how to achieve compliance with the standard for tools used to develop a system [24]. In general, DO-178B has been criticized for inadequately allowing for innovation in software development practices [20].

DO-178C [45] is the next iteration of the standard and plans to address these issues. While DO-178B was a significant update to DO-178A, the core of DO-178C is expected to be only a minor change to DO-178B [21][27]. The changes to the core of the standard address some inconsistencies in wording and incorporate the errata of DO-178B [41]. However, these changes are relatively modest and do not make up the bulk of the updates which are found in DO-178C.

The most significant change to be incorporated into DO-178C is the inclusion of technology-specific supplements that provide guidance for the use of new methods and advanced technologies in software development. These supplements address model-based development, object-oriented technology, tool qualification, and the use of formal methods for verification and validation [27][37][48]. DO-178C details the activities needed for these processes, as well as the certification criteria for software utilizing them.

DO-178B was created before model-based development and object-oriented technology came into widespread use for avionics software systems, so it assumes the use of procedural programming languages such as Ada 83 or C [27]. However, modeling and object-oriented techniques have gradually become more popular in the avionics industry. As a result, the FAA has accepted these methods for use in safety-critical software, and DO-178C allows for the controlled use of modeling and object-oriented software in all avionics systems, including Level A (the highest level) [21]. DO-178C addresses the use of object-oriented programming languages and their potential pitfalls, as well as guidelines for acceptable use. The standard also provides guidance as to the specific acceptability criteria for the use of modeling techniques, and traceability is highly emphasized.

Furthermore, unlike DO-178B, DO-178C officially accepts the use of formal methods in the development of safety-critical avionics software to reduce software testing. It allows formal methods to verify requirements correctness and consistency. Furthermore, the standard accepts the use of formal methods to augment code reviews and to verify or replace test cases used for low-level requirements [21]. DO-178C also addresses the qualification of tools used for automating development and verification activities, including third-party commercial off-the-shelf tools, which are becoming increasingly popular [27]. In general, the revision is expected to allow avionics projects to adopt modern safety- and software-engineering practices.

Transitioning from DO-178B to DO-178C is not expected to be difficult for pre-existing systems or those currently in development to DO-178B. In general, the core DO-178B document is unchanged in DO-178C, and the major updates are to be found in the additional technology supplements. Furthermore, compatibility with DO-178B has been a major concern in the formulation of DO-178C. As a result, systems certified to DO-178B does not require re-certification to DO-178C, and projects currently in-development should encounter minimal costs related to the transition [27].

III. Evolution of DEF STAN 00-56

DEF STAN 00-56 [11] is a safety standard created by the UK Ministry of Defense (MOD) that describes the requirements for the management of the safety of defense-related systems. The standard specifies safety management procedures, analysis techniques, and safety verification techniques that are intended to aid in ensuring system safety. This standard is applicable to all Ministry of Defense authorities and projects, and is intended to provide guidance for the development of safety-critical projects.

The current version of DEF STAN 00-56, Issue 4 [11], covers both hardware and software safety issues, and its safety requirements are broadly applicable to all MOD projects. However, the expansive scope of this standard is relatively recent and previously overlapped with the use of other MOD standards such as DEF STAN 00-54 [7], 00-55 [8], and 00-58 [12].

Issue 3 of DEF STAN 00-56 [10] was a significant departure from Issue 2. In addition to changing its focus from a requirements-based to a goal-based standard, it incorporated aspects of DEF STAN 00-55, rendering that standard obsolete. Several major criticisms of DEF STAN 00-55, as well as previous issues of DEF STAN 00-56, contributed to this major overhaul.

Experience and feedback from MOD stakeholders and users in industry had shown that the rigorous requirements of DEF STAN 00-55 and earlier issues (namely, Issues 1 and 2) of DEF STAN 00-56 were needlessly strict for contractors [14]. Both DEF STAN 00-55 and Issue 2 of DEF STAN 00-56 [9] had focused on the use of Safety Integrity Levels (SILs), which were determined by analyzing both the consequence and probability of failure of system risks. However, these SILs proved difficult to allocate and highly specific in the techniques they required for proof of safety [6]. Moreover, SILs were widely misunderstood and misused in industry applications [6][46][47].

Additionally, DEF STAN 00-55 and Issues 1 and 2 of 00-56 had been criticized for not allowing contractors the flexibility to tailor their approach for each individual project to best achieve the safety requirements [14]. In general, the standards were disparaged for heavily over-emphasizing process rather than product [18], imprecise requirements [18], and not sufficiently addressing safety issues [26]. DEF STAN 00-55 was considered too long and unclear [3][39][40], and experience showed that examples given by Issues 1 and 2 of 00-56 which were intended to provide guidance were often copied directly by contractors rather than altered to fit the project [14].

As a result of these criticisms, the MOD released Issue 3 of DEF STAN 00-56 [10], a major departure from both 00-55 and the previous issue of 00-56 [9]. The new goal-based approach for the standard provides general requirements but does not mandate a specific method for how they are to be met [6][11][14]. Furthermore, the use of SILs has been completely eliminated. In Issues 3 and 4 of DEF STAN 00-56, instead of following a specific process to ensure safety, contractors must propose and justify their chosen methods of compliance and provide evidence for the safety of their systems in a safety case [11]. Rather than requiring a pre-determined methodology, the standard now requires evidence-based proof of safety of the system. This puts a higher burden of proof on contractors but also allows them more flexibility in tailoring their safety approach to fit the needs of their specific project. This flexibility can also permit contractors to use other relevant standards and utilize their recommendations in their argument for proof of the safety of their system.

As previously noted, Issue 3's publication of DEF STAN 00-56 made several other MOD standards obsolete (00-54 [7], 00-55 [8]), and 00-58 [12]). However, DEF-STAN 00-55 was chiefly concerned with the production of safety-critical software, while the new DEF STAN 00-56 is broadly applicable to all safety-critical defense systems and does not stress software specifically. Due to this change, there were some industry concerns that Issue 3 of DEF STAN 00-56 did not provide enough detailed guidance regarding how safety-critical software should comply with the standard in the absence of DEF-STAN 00-55's guidance [5][7].

Issue 4 of DEF-STAN 00-56, published in June 2007, clarified Interim Issue 3 of the standard. It removed the "Interim" status but did not introduce any significant new requirements or policies [11]. However, in response to the criticisms regarding the lack of guidance for safety-critical software and electronic systems, in August 2007 Part 1 of the standard was updated to include a section on "Additional Guidance Regarding the Safety of Systems Containing Complex Electronic Elements", which included recommendations for how to deal with safety-critical software [11]. This added section provided guidance for contractors in how to achieve safety of systems containing safety-critical software, but retained the goal- and evidence-based approach of the standard.

The evolution of DEF STAN 00-55 and 00-56 has not received much criticism. However, the changes to the standards have posed some challenges for both pre-existing and future projects.

First, since Issues 3 and 4 of DEF STAN 00-56 are goal-based, there is some uncertainty in how exactly to achieve compliance with the standard. Since specific methods and processes are no longer required, contractors must determine for themselves how to ensure and prove safety. Furthermore, Issues 3 and 4 of DEF-STAN 00-56 include some ambiguity as to the level of evidence that is sufficient to prove safety. A safety case is required, but the standard does not in detail describe what is sufficient for meeting its goals. Issues 3 and 4 of 00-56 do not prescribe required methods or types of proof, so it is not entirely clear how to comply with the standard [23]. As a result, contractors are faced with the challenge of determining what constitutes sufficient safety evidence.

Furthermore, the removal of SILs from Issues 3 and 4 of 00-56 means that there is also no pre-determined risk acceptability scheme. Thus, it becomes a burden on the contractor to determine acceptable or tolerable levels of risk, which can be a problematic moral question. Since the standard does not strictly define the level of risk that is acceptable, the details for risk tolerance for the project must be negotiated among the various parties involved, among which it may be difficult to find agreement. Contractors may also lack sufficient expertise in order to make judgments on risk acceptability [23].

Another challenge that may result from the evolution of the standards is that of cost estimation and program management. In the UK, projects are often contracted on a fixed-price basis. Previously, the process-based nature of the standards ensured that most projects were approached in fairly similar ways, and some of the cost could be estimated based on knowledge of the necessary safety activities. However, with Issues 3 and 4 of 00-56, methods for compliance are open, and what constitutes sufficient safety evidence is not entirely clear. As a result, contracting at a fixed price may be high-risk [23].

On the other hand, the new versions of the standard are a benefit to projects utilizing new technologies or safety strategies [19]. The specific methods for safety assurance provided in process-based standards such as Issues 1 and 2 of DEF STAN 00-56 and 00-55 could prove potentially restrictive as safety- and software-engineering practices evolve. Furthermore, some prescribed approaches to determining safety may not be appropriate to all projects. The goal-based approach in Issues 3 and 4 of 00-56 allows for contractors to utilize new techniques, choose which methods are applicable to ensure safety for their project, and incorporate recommendations from other relevant safety standards as needed. As a result, it is expected that the current issue of DEF STAN 00-56 will be widely applicable and less affected when technology changes [19].

For projects previously using DEF STAN 00-55 and Issues 1 or 2 of 00-56, transition to Issues 3 and 4 of 00-56 is not expected to be difficult. The goal-based nature of Issues 3 and 4 of 00-56 allows contractors to choose and justify their safety methodology. As a result, it is compatible with previous versions of the standard, since the required practices established under DEF STAN 00-55 and Issues 1 and 2 of 00-56 can be justified by the contractor in a safety case. For example, the HEAT/ACT project, by UK-based Westland Helicopters, was in development when Issue 3 of 00-56 was released [4]. This project involved significant revisions to a helicopter's aircraft systems, including changes to the hydraulics and the migration of the flight controls from a mechanical system to two new software-intensive fly-by-wire computers. Although the project was initially conceived when Issue 2 of 00-56 was in use, the creators of the project noted that the safety plan could be easily converted to comply with Issue 3 of the standard. Instead of referring to specific clauses in Issue 2 of 00-56, they could provide an explanation and analysis of their safety processes and principles to achieve compliance with Issue 3 of the standard. However, the actual processes used to ensure safety did not need to be changed [4]. The general safety case, as well as the sub-system safety cases, would remain largely the same in their arguments.

IV. Evolution of MIL-STD-882

US Department of Defense (DoD) MIL-STD-882 [31], the first standard for the assessment of system safety, was published in 1969 and made the use of a system safety program mandatory for all DoD projects. Since its release, this standard has been the primary reference for system safety for the DoD, while experiencing significant updates and revisions throughout its history.

MIL-STD-882's 1969 release was the result of the defense industry's new focus on system safety engineering in the 1950s and 1960s as military systems increased in complexity. At the time, much documented guidance existed for how to achieve safety of technologically complex systems.

To fill this void, in 1962 the US Air Force Ballistic Missile Division published the first system safety specification, BSD Exhibit 62-41 [1]. This document discussed basic safety engineering concepts, including hazard classification, design order of precedence, and systematic analysis through the design and development phases of a project. However, BSD Exhibit 62-41 was only designed for ballistic missile systems. The scope of this document was expanded in 1963, when the US Air Force released MIL-S-38130 [29], which increased the intended audience to include the creators of aeronautical, space, missile, and electronic systems [17][25]. MIL-S-38130 also further discussed the definitions of hazards and their classifications.

In 1966, MIL-S-38130 was revised to MIL-S-38130A [30], which expanded the safety engineering lifecycle and introduced the Gross Hazard Study (now known as the Preliminary Hazard Analysis). Furthermore, the revision began the focus on the importance of management control of the system safety program [51].

In 1967, the DoD took steps to formalize the safety engineering principles discussed in the MIL-S-38130A specification. The culmination of this process was MIL-STD-882, released in July 1969, which made a system safety program mandatory for all DoD projects and systems [55]. The new standard greatly expanded the guidance provided in the previous documents, and adopted a phase-oriented approach to system safety, in which safety activities were associated with the various phases of system development.

MIL-STD-882 was revised in 1977 to MIL-STD-882A [32]. The primary change was a focus on risk acceptance as a criterion for system safety programs. The update also added the concept of hazard probability and frequency of occurrence in order to refine the hazard severity categories [25]. Management responsibilities also became more specific due to an increased focus on contract definition [17].

The standard continued its evolution with the publication of MIL-STD-882B [33] in 1984. This update was a major change to the previous version, expanding the guidance on risk acceptance, tailoring, and off-the-shelf acquisition. Furthermore, MIL-STD-882B was the first version of the standard to include a detailed discussion of software [17][51]. The DoD added Notice 1 to MIL-STD-882B in 1987, which further discussed software tasks and their relation to system safety.

The next revision of the standard, MIL-STD-882C [34], was published in 1993 and integrated hardware and software into the system safety effort. This update got rid of separate software tasks and addressed safety analysis as a system-wide process rather than one that focused on hardware and software separately [25].

MIL-STD-882D [35], published in 2000, was a significant revision to the previous versions of the standard. It reflected the DoD Military Specifications and Standards Reform initiative, which focused on performance rather than process [17]. As a result, MIL-STD-882D was a goal- and performance-based standard, which focused on the intended safety results rather than specific processes or activities (in other words, "what to do" rather than "how to do it") [55]. The new revision required the use of a System Safety Program, but removed the recommended hazard analysis tasks found in previous versions [2]. This change allowed for greater freedom to contractors for how to meet safety expectations, but removed guidance as to the tasks that can be used to satisfy the requirements of the standard. This has led to some confusion among contractors due to the decreased step-by-step guidance, and the new performance-based focus of the standard requires more understanding and up-front planning to ensure that a properly structured system safety program is followed [2].

However, government and industry found difficulties on utilizing the guidelines provided in the standard without expertise instructions and lack of specific implementation steps for the safety requirements [50]. Therefore, MIL-STD-882C, sometimes with supplemented DO-178B, is treated as the alternative [49] while brought about some confusion on recommended task for safety engineering. Furthermore, DoD did not have any policy or agreement for software development to mandate any safety standards back then. The demands for these standards are significantly lowered by these issues [50].

Fortunately, in September, 2004, DoD published a memo “Defense Acquisition System Safety” to request risk management in software system development by using MIL-STD-882D [54], and a course of “System Safety in Systems Engineering” was created at the Defense Acquisition University, which was the very first exceptional training for the standard [50]. Also, in the updated information and documents found in November, 2006 Defense Acquisition Guidebook [13] and December 2008 DoD 5000.02 “Operation of the Defense Acquisition System” [15][16], more guidance are provided to resolve the confusing issues of integrating MIL-STD-882D safety tasks into the system development process [50].

In November 2009, the Environmental Support Office met to discuss criticism of and potential changes to MIL-STD-882D. Change 1 was the revision that resulted from the discussion. Due to a lack of standardization and some confusion regarding hazard risk identification found in the standard, Change 1 sought to improve these issues as well as to further discuss health and environmental risk management.

In May 2012, important changes were introduced in the newer revision of the standard, MIL-STD-882E [36], including a deeper emphasis on software related technology, emphasis on reducing confusion between Mishap, Hazard, and Risk, and new or updated definitions [50]. Furthermore, risk assessment matrices and software safety matrices have been added to the standard [25] in order to provide more standardized guidance, which was lacking in the previous version and is present in several other standards such as DO-178B and the FAA System Safety Handbook. Safety integrity levels, such as those found in DO-178B, have also been included [2].

V. Lessons Learned

After examination of these three software safety standards, four lessons for developing useful standards in software safety can be learned:

A. *Levelized Effort*

These software safety standards tend to categorize criticality based on risk assessment of the system, and require greater levels of effort for more highly critical categories in system development and verification processes. For example, both DO-178 and DEF STAND 00-56 use software integrity levels to rank hazards caused by software failures. Furthermore, the latest issue of MIL-STD-882E also includes the leveling feature.

B. *Expertise*

Though justified schemas and notations such as software levels have been commonly used in constructing software safety standards, these schemas and notations have sometimes been misunderstood or misused when confronting different situations. To resolve the problem, we may:

- Consult a trained expert.
- Establish a unified standard with exclusive property lists for different interpretations.
- Replace the schemes with simpler and more flexible methods.

C. *Flexibility*

As mentioned above, older standards tend to have rigorous schemes to restrict the system development process. However, due to the increasing complexity of software systems and rapidly evolving new technology, highly flexible and widely applicable standards are preferred when developing new systems or introducing new features into a legacy system. To provide this flexibility, goal-based guidelines considering both software safety and cost efficiency can be used.

D. *Backward compatibility*

Backward compatibility is also an important feature. For those pre-existing projects migrating to a new environment or introducing new technology, older standards sometimes do not include the views and experience needed for emerging software safety issues. Newer versions of software safety standards (such as Issues 2 and 3 of 00-56) may have different schemes. Thus, some conversion techniques must be provided. For example, the “required practices” in older versions of 00-56 can be defined as a safety case.

VI. Conclusion and Future Work

In this paper, we presented the history of three major software safety standards that provide either guidelines to follow during the development process or goals to achieve for safety purposes. After evaluating the evolution of these standards, we found four major properties that are most likely to exist in the current version of software safety standards or to be included in future versions. First, it is most efficient to categorize the risk level and put effort on the more critical issues. Second, it is helpful to seek expertise to clarify the schemes of standards. Third, goal-based guidelines provide flexibility for new technology in newer versions of standards. Finally, compatibility of newer standards with older versions is beneficial for developers of pre-existing projects.

In future work, we plan to propose criteria to evaluate the performance of different software safety standards, and to demonstrate the application of such criteria to standards selection for a given project.

ACKNOWLEDGMENT

This paper is based on the work by the students who participated in the REU (Research for Undergraduate Students) program at the University of Texas at Dallas from 2009 to 2013. The program is funded by the National Science Foundation (NSF CNS-1156977). More details of our REU program can be found at http://paris.utdallas.edu/reu.

REFERENCES

[1] BSD Exhibit 62-41, *System Safety Engineering: Military Specification for the Development of Air Force Ballistic Missiles,* USAF Ballistics System Division (BSD), April 1962.

[2] M. E. Caro, "Lessons Learned with the Application of MIL-STD-882D at the Weapon System Explosives Safety Review Board," 8th Systems Engineering Conference, 26 October 2005.

[3] P. R. Caseley, N. Tudor, and C. O'Halloran. *The Case for An Evidence Based Approach to Software Certification*, MOD Equipment Safety Assurance, 2003.

[4] P. Chinneck, D. Pumfrey, and J. McDermid. The HEAT/ACT *Preliminary Safety Case: A case study in the use of Goal Structuring Notation*, 2004.

[5] Cobham Technical Services., *The Certification of Software Developed to DO-178B.*

[6] R. M. Connor, *Vetronics Standards and Guidelines,* QINETIQ /EMEA/TS/CR0702540, Issue 3, October 2009.

[7] Defence Standard 00-54, *Requirements for Safety Related Electronic Hardware in Defence Equipment,* Issue 2, 13 December 1996.

[8] Defence Standard 00-55, *Requirements for Safety Related Software in Defence Equipment,* Issue 2, 1 Aug 1997.

[9] Defence Standard 00-56, *Safety Management Requirements for Defence Systems*, Issue 2, 13 Dec 1996. (Now at Issue 4 as of 1 June 2007)

[10] Defence Standard 00-56 (Interim), *Safety Management Requirements for Defence Systems,* Issue 3, 17 Dec 2004. (Now at Issue 4 as of 1 June 2007)

[11] Defence Standard 00-56, *Safety Management Requirements for Defence Systems*, Issue 4, 1 June 2007.

[12] Defence Standard 00-58, *HAZOP Studies on Systems Containing Programmable Electronics,* 19 May 2000.

[13] Defense Acquisition University, *Defense Acquisition Guidebook,* 19 February 2010.

[14] Director General Safety & Engineering, DStan, *UK Ministry of Defence, Standards in Defence News*, July 2007, Issue 205.

[15] DoD 5000.2-R, *Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System (MAIS) Acquisition Programs*, US Department of Defense, 5 April 2002.

[16] DoD Instruction 5000.2, *Operation of the Defense Acquisition System,* US Department of Defense, 5 April 2002.

[17] C. A. Ericson, "System Safety: What, Why, and How We Got There," *Naval Sea Systems Command*, Volume 7, Issue No. 3, pg. 10-17.

[18] N. Fenton, *How to Improve Safety Critical Systems Standards,* Center for Software Reliability, City University, London, 1997.

[19] R. D. Hawkins, *Using Safety Contracts in the Development of Safety Critical Object-Oriented Systems*, PhD Thesis, University of York, Department of Computer Science, March 2006.

[20] K. J. Hayhurst, M. C. Holloway, C. A. Dorsey, J. C. Knight, N. G. Leveson, G. F. McCormick, and J. C. Yang, *Streamlining Software Aspects of Certification: Technical Team Report on the First Industry Workshop,* National Aeronautics and Space Administration, Langley Research Center, April 1998.

[21] V. Hilderman, *DO-178C Synopsis: DO-178C Facts, DO-178C Resources, & DO-178C Answers,* HighRely Inc., 6 January 2010.

[22] L. A. Johnson, *DO-178B, "Software Considerations in Airborne Systems and Equipment Certification,"* Flight Systems, Boeing Commercial Airplane Group, October 1998.

[23] T, Kelly, J. McDermid, and R. Weaver, "Goal-Based Safety Standards: Opportunities and Challenges," University of York, Heslington, York. In: *Proceedings of the 23rd International System Safety Conference*, 2005.

[24] A. J. Kornecki, *DOT/FAA/AR-06/35: Software Development Tools for Safety-Critical, Real-Time Systems Handbook*, Office of Aviation Research and Development, Federal Aviation Administration, June 2007.

[25] B. McAllister and J. Turner, "Evolution of MIL-STD-882E," 8th Annual Systems Engineering Conference, San Diego, 26 October 2005.

[26] J. A. McDermid, "Software Safety: Where's the Evidence?" 6th Australian Workshop on Industrial Experience with Safety Critical Systems and Software, Brisbane, 2001.

[27] J. McHale, *Upgrade to DO-178B certification -- DO-178C to address modern avionics software trends*, Aerospace and Defense Media Group, 8 October 2009.

[28] S. Messner, *An Overview of RTCA DO-178B*. 1997.

[29] MIL-S-38180, *Safety Engineering of Systems and Associated Subsystems and Equipment*, General Requirements for. US Department of Defense, September 1963.

[30] MIL-S-38130A, *Safety Engineering of Systems and Associated Subsystems and Equipment*, General Requirements for. US Department of Defense, June 1966.

[31] MIL-STD-882, *System Safety Program Requirements,* US Department of Defense, 15 July 1969.

[32] MIL-STD-882A, *System Safety Program Requirements,* US Department of Defense, 28 June 1977.

[33] MIL-STD-882B, *System Safety Program Requirements,* US Department of Defense, 30 March 1984.

[34] MIL-STD-882C, *System Safety Program Requirements,* US Department of Defense, 19 January 1993.

[35] MIL-STD-882D, *Standard Practice for System Safety*, US Department of Defense, 10 February 2000.

[36] MIL-STD-882E, *Standard Practice for System Safety,* US Department of Defense, 11 May 2012.

[37] Military Embedded Systems, *HighRely Synopsis of National FAA Software and Hardware Meeting Includes DO-178C Status*, July 2006.

[38] J. F. Muller and C. Pinaud, *Current use of methods and standards for development and certification of safety-critical software,* Contract N. 14 899/01/NL/JA, Version 1.0, 11 August 2002.

[39] C. O. O'Halloran, "Acceptance Based Assurance," 16th IEEE International Conference on Automated Software Engineering, San Diego, California, November 2001.

[40] C. O. O'Halloran, *Recommendations for the New Safety Defence Standards,* QINETIQ/KI/TIM/CR021237/1.0, June 2002.

[41] Qualtech Consulting, Inc. *Summary of Difference Between DO-178B and DO-178C.*

[42] Radio Technical Commission for Aeronautic, Inc., *Document RTCA/DO-178*, RTCA, Inc. 1980.

[43] Radio Technical Commission for Aeronautic, Inc., *Document RTCA/DO-178A*, RTCA, Inc. 1985

[44] Radio Technical Commission for Aeronautic, Inc., *Document RTCA/DO-178B*, RTCA, Inc., 1992.

[45] Radio Technical Commission for Aeronautic, Inc., *Document RTCA/DO-178C*, RTCA, Inc., 2012.

[46] F. Redmill, *Understanding the Use, Misuse, and Abuse of Safety Integrity Levels*, Safety-Critical Systems Symposium, Southampton, UK, 2000.

[47] M. Squair, *Are Safety Integrity Levels Pseudo-Science?* June 2009.

[48] B. StClair, *DO-178C will arrive, then drive safety-critical software*, Special Interview, VME and Critical Systems, June 2009. B. StClair, "Growing Complexity Drives Need for Emerging DO-178C Standard," *COTS Journal*, November 2009.

[49] Savive Pty Ld., *MIL-STD-882 "System Safety Program Requirements" / "Standard Practice for System Safety,"* Retrieved 14 November 2010.

[50] R. E. Smith and S. G. Forbes, *Overview of Draft MIL-STD-882D w/Change 1*, NDIA Systems Engineering Conference, San Diego, CA. 28 October 2009.

[51] G. Tarajiki, *Tools of Risk Management*, 23 April 2003.

[52] W. E. Wong, V. Debroy, and A. Restrepo, "The Role of Software in Recent Catastrophic Accidents," *IEEE Transactions on Reliability*, Volume 59, Issue 3, pp. 469-473, September 2010

[53] W. E. Wong, V. Debroy, A. Surampudi, H. Kim, and M. F. Siok, "Recent Catastrophic Accidents: Investigating How Software Was Responsible," in *Proceedings of the 4th IEEE International Conference on Secure Software Integration and Reliability Improvement* (SSIRI), pp. 14-22, Singapore, June 2010

[54] M. Wynne, *Memorandum for Defense Acquisition System Safety*, 23 September 2004.

[55] T. C. Yin, *Architecting the Safety Assessment of Large-Scale Systems Integration*, Master's Thesis, Naval Postgraduate School, December 2009.