WIKIPEDIA

Software system safety

In software engineering, **software system safety** optimizes system safety in the design, development, use, and maintenance of software systems and their integration with safety-critical hardware systems in an operational environment.

Contents

Overview

Software system safety is a subset of system safety and system engineering and is synonymous with the software engineering aspects of Functional Safety. As part of the total safety and software development program, software cannot be allowed to function independently of the total effort. Both simple and highly integrated multiple systems are experiencing an extraordinary growth in the use of computers and software to monitor and/or control safety-critical subsystems or functions. A software specification error, design flaw, or the lack of generic safety-critical requirements can contribute to or cause a system failure or erroneous human decision. To achieve an acceptable level of safety for software used in critical applications, software system safety engineering must be given primary emphasis early in the requirements definition and system conceptual design process. Safety-critical software must then receive continuous management emphasis and engineering analysis throughout the development and operational lifecycles of the system. Software with safety-critical functionality must be thoroughly verified with objective analysis.

Functional Hazard Analyses (FHA) are often conducted early on - in parallel with or as part of system engineering Functional Analyses - to determine the safety-critical functions (SCF) of the systems for further analyses and verification. Software system safety is directly related to the more critical design aspects and safety attributes in software and system functionality, whereas software quality attributes are inherently different and require standard scrutiny and development rigor. Development Assurance levels (DAL) and associated Level of Rigor (LOR) is a graded approach to software quality and software design assurance as a pre-requisite that a suitable software process is followed for confidence. LOR concepts and standards such as DO-178C are NOT a substitute for software safety. Software safety per IEEE STD-1228 and MIL-STD-882E focuses on ensuring explicit safety requirements are met and verified using functional approaches from a safety requirements analysis and test perspective. Software safety hazard analysis required for more complex systems where software is controlling critical functions generally are in the following sequential categories and are conducted in phases as part of the system safety or safety engineering process: software safety requirements analysis; software safety design analyses (top level, detailed design and code level); software safety test analysis, and software safety change analysis. Once these "functional" software safety analyses are completed the software engineering team will know where to place safety emphasis and what functional threads, functional paths, domains and boundaries to focus on when designing in software safety attributes to ensure correct functionality and to detect malfunctions, failures, faults and to implement a host of mitigation strategies to control hazards. Software security and various software protection technologies are similar to software safety attributes in the design to mitigate various types of threats vulnerability and risks. Deterministic software is sought in the design by verifying correct and predictable behavior at the system level.

Goals

- Functional safety is achieved through engineering development to ensure correct execution and behavior of software functions as intended
- Safety consistent with mission requirements, is designed into the software in a timely, cost effective manner.
- On complex systems involving many interactions safety-critical functionality should be identified and thoroughly analyzed before deriving hazards and design safeguards for mitigations.
- Safety-critical functions lists and preliminary hazards lists should be determined proactively and influence the requirements that will be implemented in software.
- Contributing factors and root causes of faults and resultant hazards associated with the system and its software are identified, evaluated and eliminated or the risk reduced to an acceptable level, throughout the lifecycle.
- Reliance on administrative procedures for hazard control is minimized.
- The number and complexity of safety critical interfaces is minimized.
- The number and complexity of safety critical computer software components is minimized.
- Sound human engineering principles are applied to the design of the software-user interface to minimize the probability of human error.
- Failure modes, including hardware, software, human and system are addressed in the design of the software.
- Sound software engineering practices and documentation are used in the development of the software.
- Safety issues and safety attributes are addressed as part of the software testing effort at all levels.
- Software is designed for human machine interface, ease of maintenance and modification or enhancement
- Software with safety-critical functionality must be thoroughly verified with objective analysis and preferably test evidence that all safety requirements have been met per established criteria.

See also

- Software assurance
- IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems
- ISO 26262 - Road vehicles – Functional safety
- Functional Safety
- Software quality
- System accident

References

This article incorporates public domain material from the United States Army document: "Software handbook" (http://www.monmouth.army.mil/cecom/safety/sys_service/software_handbook.htm).

Retrieved from "https://en.wikipedia.org/w/index.php?title=Software_system_safety&oldid=1045888761"