ONRSR Fact Sheet

Safety Critical Software Assurance

July 2020

Failure to adequately assure the safety of critical software can lead to serious safety incidents.

Internationally, there have been significant incidents associated with issues surrounding software assurance, which the Australian railway industry can learn from. These show how important it is for Rail Transport Operators (RTOs" to correctly manage the development and assurance of safety critical software.

Good practice safety assurance

RTOs must ensure that their safety management systems support the complex activities associated with assuring the safety of software systems.

International standards exist to support such activities particularly in view of the prevalence of computer-based signalling systems and rolling stock control systems now being utilised and the migration away from wayside control. Good practice includes compliance with the current versions of relevant international standards, such as but not limited to:

- EN 50126 - Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)
- EN 50128 - Railway applications - Communication, signalling and processing systems software for railway control and protection systems
- EN 50129 - Railway applications - Communication, signalling and processing systems - safety related electronic systems for signalling
- IEEE 1474.4 - Recommended practice for functional testing of a communications-based train control system.

International lessons learnt

Lessons can be learnt from previous incidents associated with safety critical software failures. Notable examples include recent investigations involving critical safety incidents on newly introduced signalling systems on the UK's ETCS Cambrian Coast Line[1] and Hong Kong's CBTC Tsuen Wan Line[2].

Reports from these are publicly available and show outcomes that may have been avoided by following a rigorous software development lifecycle.

Understand the full scope of the software system

When selecting and applying safety critical software, an RTO must fully understand the complex software system that is being provided to ensure it can adequately assess compliance and associated safety risk. This involves knowing its restrictions or limitations, and its operational and maintenance requirements. A lack of understanding can contribute to failures to implement safety critical software or data correctly.

Example: Signalling data was not reproduced during automatic failover of a vital signalling computer. This led to temporary speed restrictions not being implemented despite the signaller's terminal reporting that they had been loaded correctly

Example: Signalling data was not fully reproduced during automatic failover of the two main signalling processors to the third 'warm standby' processor. This led to a serious protection failure at a junction between two railway lines.

Implement good practice software design

When procuring safety critical software, RTOs should ensure that good practice software design is implemented across the whole development lifecycle.

This should include a clear understanding of the safety requirements, how non-standard features impact the safety of the whole system, and the basis upon which the final safety argument is presented.

Example: Design progressed with the safety argument not being planned for operational readiness testing. Design safety did not consider operational readiness, which omitted any relevant

safety requirements for the software. The software could not undergo full validation testing, resulting in a serious safety incident.

Verify and validate software systems

Throughout the lifecycle of implementing the safety critical software, RTOs must ensure that safety critical software is appropriately verified and validated in accordance with good practice.

Example: A systems integrator failed to show appropriate verification and validation of the software design through the design and development phase. As a result, the safety critical software was non-compliant with good practice development, leading to a serious safety incident.

Independent safety assessment

Independent safety assessment is a powerful tool to support assurance of complex safety critical software systems. However, care must be taken to avoid any assessment errors, and to challenge the assumptions and activities behind such assessments.

Example: An assessment accepted a generic application safety case that was still in development, in place of a safety argument supporting the specific application of the software system. This led to a flawed safety argument and the introduction of software which caused a safety incident.

Example: An independent safety assessor (ISA) was engaged with a limited remit in respect of the railway operations. As its scope did not cover "readiness for testing", the project failed to identify, and hence stop, an inappropriate software version being used for drills and exercises. Consequently, inadequate controls were implemented, leading to a serious safety incident.

Acceptance of software systems

RTOs must ensure that robust acceptance processes are in place for new or revised safety critical software systems. It is important to challenge the work done by projects, through regular audits, inspection and assessments, to assure safety.

Example: An operator employed their own discipline experts to accept safety cases associated with complex safety software systems and any observations raised by an ISA. They did not review the safety case documents being provided to an adequate level, relying on the ISA to perform the bulk of the review. This allowed safety deficiencies to be commissioned into service.

Example: An operator engaged an ISA who clearly identified safety concerns about the system developer's software development processes. These concerns were not adequately addressed leading to safety deficiencies in the commissioned software systems.

Key considerations

RTOs must ensure all participants in safety critical software projects understand that:

> systematic failures resulting from software errors can often be undetectable and unpredictable

> mitigation against systematic failures (e.g. proper planning, coding, configuration-control, testing) needs to be adopted throughout the software lifecycle, including updates, patching and maintenance

> railway operations must comply with the software limitations (e.g. application conditions and use of failure recording and management systems)

> good practices must be adopted for software development

> safety critical software should be verified and validated throughout its development lifecycle in accordance with good practice standards

References

1. RAIB Rail Accident Report Loss of safety critical signalling data on the Cambrian Coast line (Report 17/2019)
2. EMSD Investigation Report on Incident of New Signalling System Testing on MTR Tsuen Wan Line (5 July 2019)