An empirical study of the practical application of EN50128 to software engineering

Dr. Andrew Hussey

Maria Hill

Martin Hughes

Lionel van den Berg

Abstract

Software-based Systems are typically complex and the process to develop such systems therefore requires rigour and professional discipline to be applied to minimise error and rework. In the railway domain, the standard EN50128 has been developed, based on IEC 61508-3, to aid the development of Software Systems for Railway applications. Although EN50128 is widely used, there are few empirical studies of the efficacy of the standard in helping to manage complexity or reduce error and rework. EN50128 has also been used to guide Software development at Hitachi Rail. We examine an Industrial Case Study involving the application of EN50128 to a Basic Integrity Railway Software System. The Case Study is a part of the Train Control System deployed by Hitachi for several clients. We consider techniques applied in accordance with EN50128, as well as what was considered by the team applying them as giving the most benefit. We also consider limitations that were identified and suggest some possible improvements to our approach for the future.

Keywords: Software Systems, Industrial case study, Railway, EN50128

1 Introduction

The CENELEC standard EN50128 is commonly applied to railways Software Engineering as part of the overall CENELEC lifecycle, in conjunction with EN50126 and EN50129. The objective of the EN50128 standard is to provide a framework for the Software process, based on the SIL of the Safety Functions implemented by that Software. Human error is at the root of all systematic safety risks, as opposed to random failures. The EN50128 standard is specifically intended to help avoid systematic software failures by reducing human error rates when creating software.

Based on the application of the EN50128 process, a corresponding Functional Failure Rate may be claimed for the Software in respect of its Safety Functions. Implicitly, the application of EN50128 should lead to more robust Software with fewer faults.

However, despite this, there are few empirical studies of the application of EN50128, confirming that the application of the standard has resulted in more robust Software with fewer faults, or has given other indirect benefits, in line with commonly accepted Software Best Practices.

This paper considers a Case Study for the application of EN50128 to the development of a Train Control System Software product, with integrity level Basic Integrity. We considered the application of EN50128 from two perspectives:

- Whether the Software Development team perceived benefits in line with current Software Best Practice
- Whether objective measures of the Software process indicated improvements in the quality of the Software.

The key contributions and additions to current learning discussed in this paper are as follows:

1. a comparison of EN50128 to commonly accepted Software Best Practices
2. The empirical study of the application of EN50128 to a real-world Software project
3. Consideration of which aspects of EN50128 were seen as most useful to improve the quality of the Software, taking account of Software Best Practices

2 Acronyms, Abbreviations, and Definitions

2.1 Acronyms and Abbreviations

Acronym	Description
GBMS	Global Business Management System
KPI	Key Performance Indicator
RAMS	Reliability Availability Maintainability & Safety
RSNL	Rail Safety National Law
RTIO	Rio Tinto Iron Ore
SFAIRP	So Far As Is Reasonably Practicable
SIL	Safety Integrity Level
STS	Hitachi Rail STS
THR	Tolerable Hazard Rate

2.2 Definitions

Requirement

(1) A condition or capability needed by a user to solve a problem or achieve an objective.

(2) A condition or capability that must be met or possessed by a product, service, or product component to satisfy a supplier agreement, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2).

Refer to the IEEE Standard Glossary of Software Engineering Terminology.

System

The abstraction level related to the Scope of Work under analysis. The System is composed of a set of Subsystems and Interfaces.

Test
The process of executing a set of checks under controlled conditions as to ascertain behaviour and performance compared to the corresponding requirements specification.

Validation
Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled (i.e. “Are we building the right product?”).

Verification
Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled (i.e. “Are we building the product right?”).

3 Literature Survey

[Boulanger19] studied the EN50128:2011 standard and the new conditions imposed on the software process. The standard CENELEC 50128 identifies a complete process for the software development of railway applications. The new version of 2011 introduced many new needs and a completely new structure. Boulanger presented the new version of EN 50128 and described how we can instantiate it. However, no case studies are given to support the process proposals made.

Myklebust, Stålhane, and Lyngby [Mykelbust15a] have studied the use of agile methods in conjunction with EN50128 to develop software and show how agile developments can be adapted to meet the requirements of EN50128.

Agile methods are gaining increasing popularity, also in safety-critical areas such as railway and offshore. The challenge is to introduce agile development without compromising safety. Development of signalling systems needs to be compliant with EN 50128. The EN50128 standard imposes rigor, but the proper adaptation of agile methods can add flexibility and efficiency. However, no real-world case studies were considered.

Likewise, Myklebust, Stålhane, and Hanssen [Mykelbust15b] have studied how an agile approach rather than an explicit V lifecycle, can be used to achieve EN50128 compliance.

Moving from a Waterfall/V-model to an agile model may affect several parts of the standards. Myklebust, Stålhane, and Hanssen have analysed the IEC 61508 (offshore/process domain) and EN 50128 (railway domain) standards and present all the requirement parts that have to be considered when applying agile methods.

In particular, for EN 50128 (railway domain) the standard itself notes: "This European standard does not mandate the use of a particular software development cycle"

Again, no real-world case studies were considered.

Lukacs and Bartha [Lukacs22] give a case study of an urban railway control system that is formally modelled and verified. They illustrate the application of their methodology via a case study of a tram-road level crossing protection system. However this is not a “real world” case study, and the focus is only on the part of the process managed vis the formal verification.

[Meisner04] documents a case study in the application of the CENELEC standards, including EN50128. The process framework applied in the case study was deemed effective in achieving compliance. The verification activities can be interpreted as a rigid schema of predecessors, but this would lead to inefficiencies, so in practice, parallel work must be tolerated. Training was seen as essential for success and the requirements phased so that the most important to close were considered first, before proceeding to other activities.

Other researchers have considered the application of parts of the process in real-world projects.

For example, [Hadj-Mabrouk20] noted that the obligations imposed by the EN50128 standard to analyse and understand the behaviour of the software has led to the development of tools to support the software developer.

Similarly, Zheng, Feng, Miao, and Pu [Zheng21] made a study of requirements-based testing and linked the outcomes they found to compliance arguments for EN50128. Their approach encouraged the automated derivation and use of test cases.

In general, few case studies analyse the application of the EN50128 process in real-world projects.

According to [Bhatt22], best software development practices lead to software being created more quickly, simply, efficiently, and with less cost. Likewise, [Włodarczyk22] asserts that following software development best practices leads to positive results for both the software project itself and the company as a whole. [Dhaduk22] further notes that software engineering best practices not only help the project to be run more efficiently but also to be more secure with fewer defects that can be exploited.

[Forsbak21] highlighted the importance of not just the software development practices, but the people that implement and apply them, and how they need to be fully engaged and committed to the software development practices for them to have positive outcomes.

[Mokevnin] looked at a checklist approach of good and bad aspects of applying practices and standards, noting that projects need to determine which engineering practices are most effective for the project. Similarly, [Reeves20] identifies a checklist of best practices for beginners.

It is also important to consider the best methods of measuring whether best practice is being applied, and the effect that it is having on software development.

Dustin, Garrett, and Gauf [Dustin09] have studied how best to measure the success of automated software testing. [Radigan] notes some metrics that can be used to assess the quality of software and track team performance. [Robson] discusses the most important metrics for focusing on the most important things.

4 EN50128:2011 and Best Practice

Comparing EN50128:2011 and software engineering best practices, we identify 5 major commonalities:

1. Emphasis on requirements – deriving them early (well before coding begins) and verifying and validating them.

2. Emphasis on testing – at all levels: component, subsystem, system, and on the need for robust test planning and test reporting.
3. Emphasis on traceability at all levels:
    a. Requirements to project scope/contract;
    b. Requirements to tests;
    c. Tests to test status and defects.
4. Emphasis on good application data processes.
5. Emphasis on maintenance activities for software.

Overarching all these points is the need for good documentation for all aspects.

5 Case Study

Rio Tinto Iron Ore (RTIO) operates a heavy-haul railway in the Pilbara region of Western Australia designed to move iron ore from mines located 300 to 500 km inland to ports for shipping overseas. The AutoHaul® Project has been delivered to Operations and is in revenue service. This includes the onboard, control centre, and wayside systems used to control and monitor locomotives and ensure the safe movement of trains, both with Drivers and in Driverless modes of operation.

The AutoHaul® Train Control System (TCS) is the control centre-based system used to monitor and control the RTIO railway and provides the Graphic User interface to the Train Control staff (Train Controllers, Maintenance Coordinators, and Train Control Supervisors). Train Controllers and Train Control Supervisors, who are responsible for the safe working of the railway, interact with the TCS to:

1. Set routes for trains;
2. Track the movement of trains through the RTIO network;
3. Monitor the status of, and receive alarms from, field equipment such as signals, track circuits, points, and asset protection devices;
4. Receive indications and status from the RTIO network signalling infrastructure and Asset Protection devices;
5. Setup the Automated and Driver Assisted missions for the train; and
6. Perform maintenance monitoring and remedial action that was previously undertaken by the Driver for AutoHaul® Trains.

The TCS was a legacy system that was already in use by RTIO before the introduction of the AutoHaul® project and the introduction of driverless trains. It has significantly been modified and upgraded and is now a complex software system made up of 41 separate subsystems at the time of publishing.

The contract for the original AutoHaul® project was signed in 2006, resulting in EN50128:2001 being the version of the standard that applied to the project. The 2001 release of the standard was strict on the processes and deliverables for subsystems with safety functions (e.g. subsystems with a SIL greater than zero), but essentially left the amount of rigour to be used for subsystems without safety functions (i.e. SIL0) up to the organisation.

In 2020 the original AutoHaul® project transitioned to a continual maintenance project called AutoHaul® Enhancements, this resulted in the new project having the latest version of the standard applied. EN50128:2011 renamed SIL0 as Basic Integrity, and was expanded to include a level of rigour for Basic Integrity subsystems that approached that of subsystems with higher SIL ratings, and left the AutoHaul® Enhancements project needing to ensure that its development practices for such subsystems were strengthened to meet the new standards.

6 Method

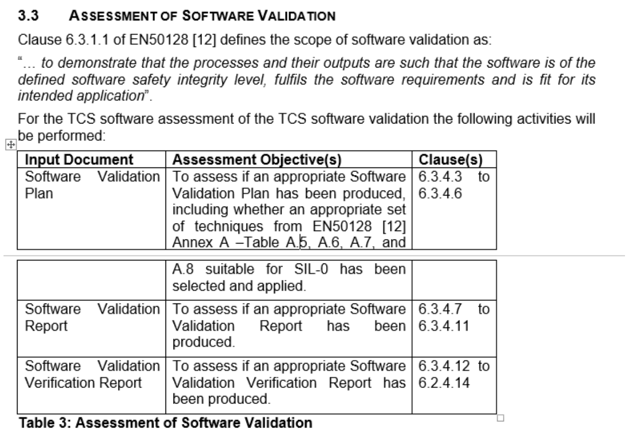
Under the requirements of the AutoHaul® Enhancements project and following the Hitachi Rail Global Business Management System (GBMS), new functionality and major changes made to the System are required to have Basic Integrity. The GBMS RAMS Roles and Mandates also define the need for governance of the subsystems' V&V processes. In advance of the closure of the AutoHaul® project and transition to the AutoHaul® Enhancements project, the RAMS team created a TCS SIL0 Assessment Plan to assess the gap in compliance to EN50128:2011. Specifically, Clause 6.4 of EN50128:2011 outlines the scope and recommendations for Software Assessment, in particular, Clause 6.4.1.1 defines the scope of Software Assessment as:

"To evaluate that the lifecycle processes and their outputs are such that the software is of the defined software safety integrity levels … and is fit for its intended application".

These clauses were used as the main guideline for the Assessment activity that covered the following activities:

- Audit sessions with the various TCS teams (requirements management, development, testing), noting that this audit approach was more collaborative than a typical audit, which was acceptable considering that this software is at Basic Integrity level
- Multiple reviews of applicable documents
- Several additional clarification meetings and discussions between relevant departments
- Previous assessments

For the TCS SIL0 assessment audits, the Plan outlined the applicable clauses and documents to be audited. An example is shown below:



3.3 ASSESSMENT OF SOFTWARE VALIDATION

Clause 6.3.1.1 of EN50128 [12] defines the scope of software validation as:

"*… to demonstrate that the processes and their outputs are such that the software is of the defined software safety integrity level, fulfils the software requirements and is fit for its intended application*".

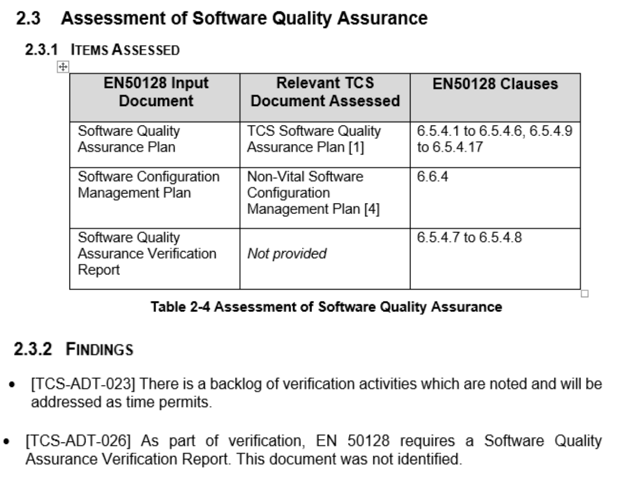
For the TCS software assessment of the TCS software validation the following activities will be performed:

Input Document	Assessment Objective(s)	Clause(s)
Software Validation Plan	To assess if an appropriate Software Validation Plan has been produced, including whether an appropriate set of techniques from EN50128 [12] Annex A –Table A.5, A.6, A.7, and A.8 suitable for SIL-0 has been selected and applied.	6.3.4.3 to 6.3.4.6
Software Validation Report	To assess if an appropriate Software Validation Report has been produced.	6.3.4.7 to 6.3.4.11
Software Validation Verification Report	To assess if an appropriate Software Validation Verification Report has been produced.	6.3.4.12 to 6.2.4.14

Table 3: Assessment of Software Validation

At the end of the assessment activity, a TCS SIL0 Assessment Report was produced outlining the results of the activities performed. In summary, the assessment identified several areas for improvement of the management process, particularly in the area of demonstration of verification. However, it also noted that work had been in progress since the audits were conducted and general improvement has been observed whilst writing the report.

An example of the results is shown below, together with sample findings:



2.3 Assessment of Software Quality Assurance

2.3.1 ITEMS ASSESSED

EN50128 Input Document	Relevant TCS Document Assessed	EN50128 Clauses
Software Quality Assurance Plan	TCS Software Quality Assurance Plan [1]	6.5.4.1 to 6.5.4.6, 6.5.4.9 to 6.5.4.17
Software Configuration Management Plan	Non-Vital Software Configuration Management Plan [4]	6.6.4
Software Quality Assurance Verification Report	*Not provided*	6.5.4.7 to 6.5.4.8

Table 2-4 Assessment of Software Quality Assurance

2.3.2 FINDINGS

- [TCS-ADT-023] There is a backlog of verification activities which are noted and will be addressed as time permits.
- [TCS-ADT-026] As part of verification, EN 50128 requires a Software Quality Assurance Verification Report. This document was not identified.

This report was then shared with and accepted by the TCS teams and further actioned. Some time later a review was held with the TCS team to determine if the assessment activity had been useful and to get feedback from the team. This was done as a post-mortem brainstorm, and the results are discussed in the next section.

7 Outcomes

As a result, the team made several changes to process that aligned with the five best practices listed above in Section 4.

The team agreed that EN50128:2011 provided a framework for deciding what activities to perform or not perform for a software release (best practice #5), and gave a language for justifying why activities were required, including requirements updates and testing (best practice #1 and #2), indicating that this had not been previously well understood, planned and managed. This language for communicating with management and the customer improved the TCS team's ability to justify the timeframes required for them to deliver updates to the software.

The team also identified that regression testing was more targeted (best practice #2), especially for Specific Application changes and there was a focus on the differences between Generic Application and Specific Application, which drove more focused testing, including for application data updates (best practice #4).

As a result of explicitly complying with EN50128:2011, it was felt that the software delivery became process-driven, the delivery scope was sized according to what was achievable following that process, and hence the TCS team was better able to manage customer expectations.

The team also highlighted that the approach taken for the assessment, where the existing process was traced to the EN50128:2011 clauses, also elicited some process gaps that needed to be addressed, especially concerning verification (best practice #1).

Similarly, it also led to the traceability of requirements being re-examined (best practice #3), which led to improvements in the requirements (best practice #1), which was a challenge because the software was already existing and in operation, but also meant that the TCS team were better able to enforce clearer deadlines regarding inputs.

It was acknowledged that even when the TCS team already had a process in place, such as the coding standard which was already broadly aligned with the applicable EN50128:2011 Annexes, this assessment activity led to some specific improvements: source code verification report, using code and database differences, configuration management of changes to ensure the proper process for a specific release (best practice #1 and #5).

The TCS team highlighted the benefit of considering the existing processes and documentation when tracing to and showing compliance with the EN50128:2011 clauses, particularly concerning the verification reports, looking to see if a verification activity was genuinely missing and needed to be addressed, or was already partially or totally covered in some existing document (best practice #1 and #5).

Further, as a result of the assessment activity the TCS team introduced code coverage checks to better implement component testing, complimented with a risk analysis approach for those areas of code not well covered (best practice #2).

7.1 What challenges were faced

The team noted that the software had originally been developed to conform to the EN50128:2001, but had not been formally assessed. The assessment against the updated EN50128:2011 version of the standard was therefore a challenge, highlighting the importance of the assessment activity.

The TCS team felt that an understanding of EN50128, particularly in terms of the terminology used, was an obstacle at the start of the assessment activity and created inefficiencies. Training and familiarisation with EN50128 was identified as a need for all software personnel.

In summary, the team noted that their potential advice for others who would embark on this exercise would be to start by understanding and document the existing process before trying to trace to or show compliance with EN50128:2011, to derive the full benefit of what is already being done and to highlight the gaps that need to be addressed.

8 Assessor Outcomes

- We were able to work well with the standard because we could work collaboratively – which benefited the TCS team and resulted in on-the-job training. If a higher SIL is required, a collaborative "assessment" could still benefit to prepare for the formal independent assessment The assessment process gave a framework for the learning by the TCS team.
- The EN50128 standard is complex and unless the team has experience and training, full compliance

is unlikely without additional inspection and assistance

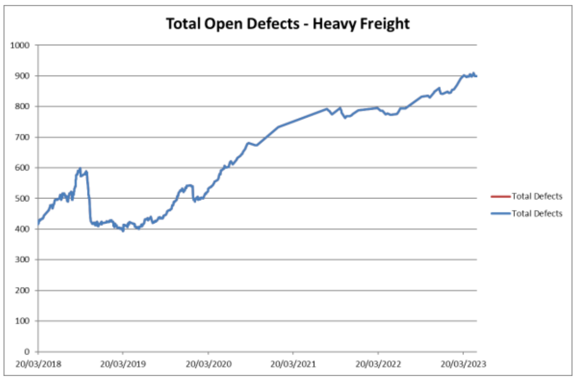
- The EN50128 standard appears extremely onerous, intimidating, and intractable for an inexperienced software team, which obstructs the application of the standard, so the collaborative assessment approach helps to overcome those barriers.

9 KPIs

To ascertain the quality of the software and how compliance with EN50128 was impacting this quality, defects, and build releases reported by the Hitachi Rail development team were tracked from 2018 to the present day. Whilst it is acknowledged that there are different methods of measuring software quality, the RAMS team identified three particular metrics as being good indicators, important in being able to measure process change and easy to model using the data available:

1. Total number of open defects;
2. Defect density; and
3. The number of baseline changes.

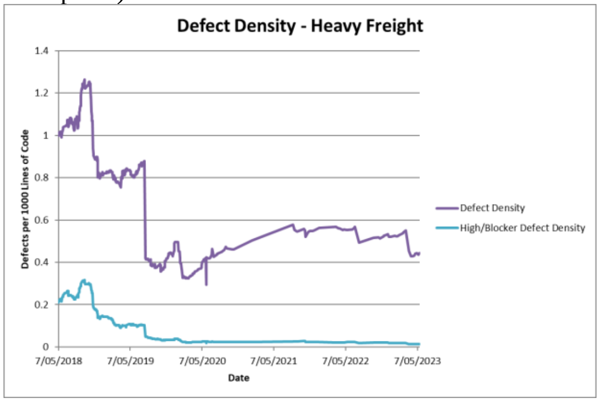
Whilst there will always be defects raised in software, the speed with which the total number of open defects rises or falls is an indication of increased or decreased software quality, and is generally accepted within the software industry to be expected to approach a bell curve over time. On the AutoHaul® project, the open defect count over time is shown below:



Continued development requested by the customer causes this line to never reach the top of the bell curve and curve back down. However, improved process compliance has seen this line flatten over time, particularly from early 2020 onwards where the rate of curve begins to decrease markedly. This indicates that:

- Defects are being cleared at a rate commensurate with the speed at which new code is being requested and developed;
- Planning and scheduling of software development on the project may have become more planned;
- Code quality may have increased; and
- Defect clearing may have become more organised.

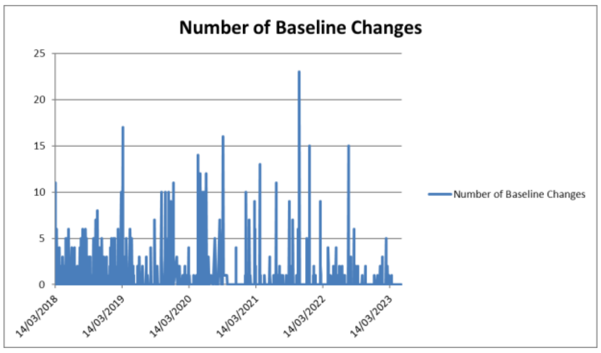
The defect density metric tracks the number of defects per thousand lines of code, and demonstrates the quality of the code written against the amount produced. A standard software component should aim for a defect density of 1 [Hamilton23], while mission-critical systems are generally acknowledged to require a target density of 0.1 [Shi20]. On the AutoHaul® project, the defect density over time is shown below. This tracks both overall defect density and that of high/blocker defects only (those defects that will cause a system crash or prevent operation of the system in the first place).



The defect density of the software has dropped over the lifetime of the project and is now at a broadly acceptable level. This is particularly evident from the sharp drops in 2018 and 2019, caused by application of an improved defect management process to align with EN50128:2011. Although the size of the software code base is continually increasing, the defect density is not. This indicates that:

- The quality of new code being added to the software code base has increased;
- The rate at which new defects are being raised may have slowed; and/or
- The safety-related functions of the software may be approaching a defect density that is considered acceptable for mission-critical systems.

The baseline changes metric tracks the number of build releases for the individual components that make up the greater system. If a large number of components are being changed at a fast and/or sporadic rate, this indicates an amount of churn and thrash in the software development process that is not commensurate with EN50128. On the AutoHaul® project, the number of baseline changes over time is shown below:



The number of changes occurring in the software baseline has lessened over the lifetime of the project and has started to occur at regular intervals, particularly from mid-2020 onwards. This indicates that:

- Software releases are now more planned;
- There is less thrash and churn in the development process; and
- The code quality has improved (as there are fewer unplanned releases to fix critical defects).

All three of these KPIs show marked improvement between 2018 (when the original TCS SIL0 Assessment Report began to highlight the gaps in standard compliance) and 2020 (when the AutoHaul® Enhancements project began and the project deliverables became subject to independent safety assessment under the new standard). While different factors affected each of these KPIs and their relative improvement, and while it is acknowledged that none of these metrics will be able to perfectly capture the state of development on the project, the fact that each of these KPIs have shown improvement over time has to an extent demonstrated increased following of best practice and standard compliance.

10 Future ideas

The software process could be improved to define more specific objective measures and KPIs. These KPIs could be linked specifically to the Software Best Practices, to determine the extent to which the application of EN50128 processes has led to positive outcomes in terms of Software Best Practices being fulfilled. For example:

- Defect number per release of the software
- For each defect time between when raised and when closed
- since the overall objective of the EN50128 is to assure a corresponding Functional Failure Rate, longitudinal studies could be initiated, to track failures over time and calculate the empirical field performance of the Software in respect of its Safety Functions.

11 Conclusions

This paper has considered how EN50128 can be used to develop Software that meets commonly accepted principles for Best Practice, as well as to reduce the occurrence of faults.

The key contributions and additions to current learning discussed in this paper were:

1. Comparison of EN50128 to commonly accepted Software Best Practices.
2. An empirical study of the application of EN50128 to a real-world Train Control System Software project, which was studied via audit of the activities undertaken.
    a. The assessment activities were themselves seen to be beneficial, particularly when done collaboratively.
3. Consideration of which aspects of EN50128 were seen as most useful by the Software team to improve the quality of the Software, taking account of Software Best Practices.
    a. Driving the software team toward best practice
    b. Giving a framework for training the team to achieve best practice, including the required competencies, not only concerning EN50128 specifically but also software development, verification and validation generally.

12 References

[Meisner04] M. Meisner, A. Meyer-Eschenbacha, L. Blessing, Adapting a Design Process to a New Set of Standards – A Case Study from the Railway Industry, In International Design Conference – Design 2004, May 18-21 2004

[Hadj-Mabrouk20] H. Hadj-Mabrouk, Application of Case-Based Reasoning to the safety assessment of critical software used in rail transport, Elsevier, 2020

[Mykelbust15a] T. Mykelbust, T. Stålhane, N. Lyngby, Application of an Agile Development Process for EN50128/railway conformant Software, Researchgate, 2015

[Mykelbust15b] T. Mykelbust, T. Stålhane, G. Hanssen, Important considerations when applying other models than the Waterfall/Vmodel when developing software according to IEC 61508 or EN 50128, Researchgate, 2015

[Zheng21] H. Zheng, J. Feng, W. Miao, G. Pu, Generating Test Cases from Requirements: A Case Study in Railway Control System Domain, IEEE Xplore, 2021

[Lukacs22] G. Lukacs, T. Bartha, Formal Modeling and Verification of the Functionality of Electronic Urban Railway Control Systems Through a Case Study, Urban Rail Transit, 2022

[Boulanger19] J-L. Boulanger, The new CENELEC EN 50128 and the use of formal methods, HAL open science, 2019

[Dustin09] E. Dustin, T. Garrett, B. Gauf, Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality, 2009

[Bhatt22] S. Bhatt, 7 Best Software Development Practices to Follow in 2023. Available at: https://www.botreetechnologies.com/blog/best-software-development-practices/, 2022

[Włodarczyk22] W. Włodarczyk, 10 Software Development Practices that Can Positively Impact Your Project. Available at: https://www.netguru.com/blog/best-software-development-practices/, 2022

[Dhaduk22] H. Dhaduk, Software Development Best Practices for High-Performing Teams in 2023. Available at https://www.simform.com/blog/software-development-best-practices/, 2022

[Forsbak21] O. Forsbak, Types of Software Engineering Practices. Available at https://www.orientsoftware.com/blog/software-engineering-practices/, 2021

[Hamilton23] T. Hamilton, What is Defect Density? Formula to Calculate with Example. Available at https://www.guru99.com/defect-density-software-testing-terminology.html, 2023

[Mokevnin] K. Mokevnin, Essential Software Engineering Practices Checklist for Your Company. Available at https://guides.hexlet.io/check-list-of-engineering-practices/]

[Reeves20] S. Reeves, Software Development Best Practices Checklist: 5 Tips for Beginners. Available at https://dzone.com/articles/software-development-best-practices-checklist-5-ti, 2020

[Radigan] D. Radigan, Five agile KPI metrics you won't hate. Available at https://www.atlassian.com/agile/project-management/metrics

[Robson] S. Robson, 5 examples of metrics that matter. Available at https://www.softed.com/au/news/5-examples-of-metrics-that-matter

[Shi20] Y. Shi, Software Reliability in Space Applications – Facts, Trends and Challenges. Available at https://ntrs.nasa.gov/api/citations/20200000671/downloads/20200000671.pdf, 2020