Development of Quality Framework Adopting Multiple Safety-Critical Standards for Software Development Incorporating Defect Prevention Techniques

Pradeep Prabhakar Oak and Umakant P. Kulkarni

February 14, 2022

Development of Quality Framework adopting multiple Safety-Critical Standards for Software development incorporating Defect Prevention Techniques

PRADEEP PRABHAKAR OAK[a,1] and *UMAKANT P KULKARNI*[b]
[a] *Director*
Oak Systems Pvt Ltd
Bengaluru, India
[b] *Professor, Dept. of Computer Science & Engineering*
SDM College of Engineering and Technology
Dharwad, India

Abstract. This is a case study of implementing a robust Quality Process Framework in a global embedded product company ensuring adoption of defect prevention techniques while adhering to the international safety-critical standards. It is a challenge to realise safety-critical software in adherence to multiple safety standards in an organisation. This paper presents a real-life scenario of developing and implementing an integrated Process Framework while ensuring adherence to multiple safety-critical standards. This single Process Framework supports both safety-critical and non-safety-critical software development in compliance to the requisite standard. Care was taken to ensure that sufficient defect prevention techniques were incorporated in the Process Framework. Defect prevention by way of a robust measurements at source, metrics monitoring, root cause analysis of defects and continual improvement of processes to prevent defects were incorporated. All the critical processes were piloted on multiple live projects and metrics were collected and analysed to fine-tune the process framework. Some of the key metrics are presented in this paper.

Keywords. Defect prevention, Hazard and Operability (HAZOP), IEC61508, ISO26262, Metrics, MISRA, Quality Process Framework (PFW), Safety-critical software, Safety Integrity Level (SIL), Software Development Lifecycle (SDLC), UL 1998.

1. Introduction

Contribution of software in Safety critical systems is increasing manifolds in modern times. There are number of international standards for each category of domains to certify for use of safety critical systems and these standards now include software standards as well [1][2][3][4][5][6][19]. It is observed that while developing a safety critical system, based on the target industry, a specific standard is identified and the

[1] Corresponding Author, Corresponding author, Book Department, IOS Press, Nieuwe Hemweg 6B, 1013 BG Amsterdam, The Netherlands; E-mail: bookproduction@iospress.nl.

system is developed for certification as per that standard. Normally, the quality process framework gets developed as per that standard and used within the organisation [9]. However, if an organisation has to develop different products for different domains, it becomes a challenge to implement the domain specific standard in each project and there is no commonality left for org-wide process implementation. This gets further complicated if an organisation has both safety-critical and non-safety-critical products to develop. If safety critical processes are applied to non-safety-critical projects, it is an overkill with wasted effort and cost. Hence, there is a need to develop an integrated quality process framework that can seemlessly satisfy the multiple safety-critical standards as well as non-safety-critical needs.

While it is important to adhere to the standards, it is equally critical to ensure better productivity and predictability by incorporating robust defect prevention processes in the quality process framework based on the metrics analysis [14][15][16][17][18]. While there are few academic attempts explored by combining few standards [5], their industry implementation experience is not known to have been established.

This paper presents study of existing system and proposes an unified Quality Process Framework [PFW] catering to both safety-critical and non-safety-critical projects without causing hurdles to the project teams. International standards ISO26262, IEC61508 and UL1998 are covered by the PFW. Metrics based defect prevention process is included into the PFW for continual process improvement [10][11][12][14][18]. Metrics are collected by implementing the PFW on a set of projects over a sizable period of time, the results analysed and key findings are presented in this paper.

2. Related works

There have been successful attempts by researchers and the industry to implement PFW adhering to a single standard and results published [4][9]. There are also few attempts to integrate couple of safety-critical standards into a PFW [5].

However, these solutions do not cater to both safety-critical and non-safety-critical systems at the same time.

There has been extensive research carried out on metrics based defect prevention approaches and lessons learnt in testing [10][11][13][14][15][16][17][18].

3. Taxonomy of the research

3.1. Safety-critical and Non-safety-critical systems

A safety-critical system is one whose malfunction or failure may result in damage to equipment, loss of human life or injuries, damage to surroundings and/or environment.

If there is no impact on any of these by a system malfunction, that system is categorized as non-safety-critical system.

3.2. IEC 61508

This International Standard sets out a generic approach for all Safety lifecycle activities for systems comprised of Electrical/Electronics/Programmable electronics systems that are used to perform safety functions. This standard provides a method for the development of the safety requirements specification necessary to achieve the required functional safety for safety related systems. It adopts a risk based approach by which the safety integrity requirements can be determined. It introduces safety integrity levels (SIL) for specifying the target level of safety integrity for safety functions to be implemented by the safety related systems. The IEC 61508 series comprises of seven parts. Out of these seven parts, Part 3 (IEC 61508-3) covers the Software requirements. This can be applied to any software that is used in the safety-critical system or is used to develop a safety-related system. IEC 61508-3 provides life cycle details for development of safety related software. Depending upon the intended SIL level, the standard provides “techniques and measures” for software development appropriate at that SIL level. IEC 61508 is considered as mother of all the safety standards. More details related to IEC61508 are available in [1][4][5][6].

3.3. UL1998

The requirements in UL 1998 address non-networked software residing in programmable components, which are application-specific. The requirements in UL 1998 are applicable when used in conjunction with an application – specific standard that contains requirements for safety related functions implemented using software. UL 1998 does not apply to software in programmable components used in general-purpose applications when the risk for the end-application cannot be identified. UL 1998 is intended to be used in conjunction with other safety standards that address programmable component hardware. More details about this standard are available in [2].

3.4. ISO26262

ISO 26262 standard applies to safety related electrical/electronic/programmable electronic systems in the automotive domain. ISO 26262 caters to Automotive Safety Integrity Levels (ASIL) A, B, C, D (where A is for minimum safety and D is for maximum safety) [3].

3.5. Safety Integrity Level (SIL)

As part of product design and development, a Hazard and Operability (HAZOP) and Risk analysis [7][8] is carried out, resulting in arriving at appropriate (SIL) for the product. Based on the Industry Standards, there could be 3 or 4 levels identified for the SIL and is a measure of risk-reduction effort employed. SIL1 to SIL4 represent the safety hazards and risk involved from lowest to highest order and hence corresponding magnitude of safety functions to reduce the same.

4. Discussion

4.1. The Problem

A global product development company with geographically distributed teams spread across multiple locations, taken as a case, carries out projects mainly in the areas of Embedded Vision Sensing, Industrial Communications, Automotive, Industrial IoT and Wireless Sensor Networks. Some of the projects are safety-critical in nature while some are not. Some of the safety-critical projects need to undergo international certification such as IEC 61508, ISO 26262 and UL 1998 due to market needs.

The projects are of different categories viz. Proof of Concept (POC), Minimum Viable Product (MVP), Full lifecycle Development project (DVP), Enhancement project (ENH). There was no Quality Management System established. However, a guideline at the organizational level was available which used to be referred by the project teams at times. However, it was not mandated and most of the times, the projects felt that this guideline document was not useful and hence was not referred.

Each project was being implemented with its own plans and most of the activities were being carried out by individual team members without any uniformity even within the same project. It was a challenge to ascertain and evaluate quality of work product and thus the key quality evaluations were getting pushed to the last phase of SDLC - system testing. The system testing phase was taking enormous amount of effort and time. With little time to fix the defects in the time left between release date and system testing, often the releases used to be made with a list of known defects.

In addition, there were numerous installation and UAT issues that used to be reported from the customer. There was no predictability in project outcome. This had led to severe customer dissatisfaction and company’s image had taken a beating. The management was unable to understand the issues as there was no uniformity in carrying out project activities and hence the management was unable to make and meet commitments to its customers. Project data was not being collected uniformly. While few projects used to gather effort and defect information, others were not collecting the same.

There was a need to fix these issues with a well-defined, easy to use, unified quality process framework. The project teams also needed a flexibility in choosing required SDLC model for implementation.

4.2. Research Methodology

Hence, it was decided to define and develop a Quality Process Framework (PFW) that is compliant to IEC 61508-3 [1] and UL 1998 [2]. Aim was to integrate prevalent practices that are tuned to ISO26262 standards [3] with other industry standards IEC61508 and UL1998 and provide a framework that meets all the project needs.

While the intended PFW was to address each of these project categories, it also had to be flexible to accommodate any other similar SDLC models that might be adopted in future. This was approached with a two phase strategy, viz. Define PFW, Implement PFW. Figure 1 outlines the approach roadmap adopted for carrying out the scoped activities.

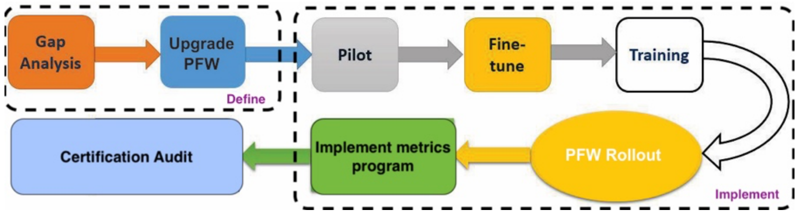

Figure 1. Approach roadmap

4.3. Define PFW

This phase involved studying the existing PFW and conducting gap analysis against the required standards and then developing the new process framework.

4.3.1. Conduct Gap Analysis

It is often seen that, even when an organization has not formally aligned its processes with any industry standards, its proven practices and processes tend to fit into the industry-standard processes. The extent of this alignment may vary from organization to organization and from process to process. Hence, a gap analysis is conducted to understand the existing practices against the proposed industry standards. Major findings from the gap analysis are listed below –

1. Existing PFW is functionally derived from ISO 26262 [3].
2. MISRA [19] has been used as a mandatory coding standard in existing PFW. While this may be necessary for Automotive software, this and such other aspects have to be relooked while building the new PFW.
3. Existing PFW does not take into account the SIL of the intended system and thus does not differentiate between need for different rigor for different SIL levels.

These identified gaps helped in identifying the specific process components for change and new process components/artifacts required to be developed in the PFW.

Though there is certain amount of reusability from existing process assets, in order to build homogeneous and seamless process assets, most of the process assets need to be redeveloped while keeping the architecture of document linkages similar.

4.3.2. Develop integrated Process Framework by upgrading existing PFW

Once the gaps are identified and agreed, the next step was to develop and update the existing PFW to incorporate the requirements of the proposed standards. This involved interpreting the standards' definition and adapting the same to the organization's products, services and domain requirements. While upgrading the existing process framework, existing culture and sensitivities of the organization are borne in mind.

Since, there was already an existing process practice in place, the intent was to minimize the impact of changes to the users. Thus, it was decided to adopt the following approach-

- To make applicable standards transparent to the users: Users need not worry about each of the standards and their interpretation. They simply need to follow the defined PFW.
- To localize the extent of changes in PFW: Instead of spreading the implementation of standards across all the processes, try to localize them such that effectiveness is retained and ease of use is established.
- To make PFW flexible enough to cater to different types of projects and different SDLC models while retaining the objectives of the standards intact.

The processes were organized and grouped on the basis of project flow, roles and interaction. Figure 2 lists the processes that were developed.

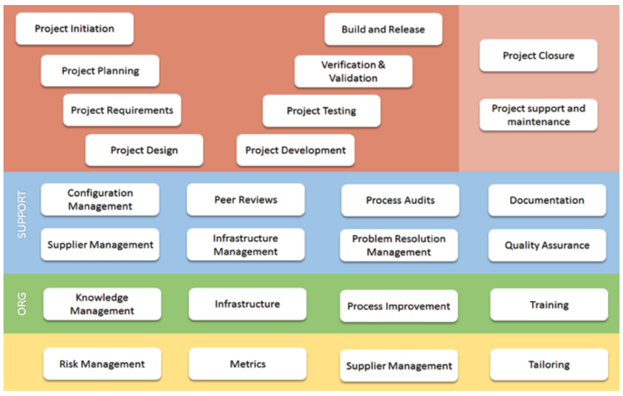

Figure 2. Process List

While defining these processes, experience of researchers and industry was useful [10][11][12][13][14][15][16][17][18]. Key aspects covered in each process are outlined in Table 1.

Table 1. Key coverage in each process

Process	**Key coverage**
Project initiation	Pre-project kick-off activities that shall be performed jointly by the software and system teams. Key inputs to this process are – (a) the SIL, (b) Hazard analysis report, (c) hardware software interface.
Project planning and tracking	Process enables to effectively manage the project from project kick-off to product release to the extent required by the SIL of the safety functions.
Safety Software Requirements	Activities in developing System and S/W requirements for safety critical products in the context of SIL, budget and schedule.
Safety Software Design	Details aspects of design of the product software, which includes s/w architecture, hardware-software interfaces, and Low-level design.
Safety Software Implementation	Consists of coding, code review and unit testing activities.
Safety Software Testing	Details generation and execution process for Test Plan, Test suite development, Testing and verifying for the desired output.
Safety Software Validation	Helps define the validation plan for the safety-related software aspects of system software.

Safety Release	Details the procedure for controlled release of a product to the intended customer.
Safety Change Management	Establishes and maintains all the changes identified and approved changed work products of a process or project.
Safety Configuration Management	Details the Configuration Management process for version control of all Configuration Items (CIs).
Safety Problem Resolution Management	Ensures all discovered problems are identified, impact analysed, managed and controlled to resolution.
Safety Quality Audit	Defines the method by which functional safety internal quality audits are implemented. Internal audit shall be conducted twice in a calendar year.
Safety Software Quality Assurance	Defines the various activities of the SQA, its functions, roles and responsibilities.
Safety Verification & Validation	The verification and validation methods as planned to verify and validate all the functional safety related documents and work products.
Safety Supplier Management	Details the process to be followed for projects that employ Sub-contract supplied components.
Safety Infrastructure Establishment	Activities for developing, identifying, selection and qualifying the tools and equipment needed for development of safety critical software.
Project Closure	Explains the activities during project closure. Collect Customer and Team members' feedback. Collate key learnings from the project, best practices, issues faced, risks identified and managed.
Project Support	Details the activities to be taken up during project support phase.
Document Identification & Control	Details the procedures for identifying the documents, controlling, changing and maintaining documentation.
Review	Describes the various review methods that may be deployed into projects – technical reviews, project status reviews and customer reviews.
Knowledge Management	Defines guidelines and provides a framework for knowledge sharing and continuous improvement across the organisation in a systematic manner.
Process Improvement	To continually improve the organization's effectiveness and efficiency through the processes used. Problem reports coming from quality audits, customer complaints, metrics report, and project reviews/closure meetings considered as input for identifying the Process Improvements.
Training	To provide training to the organization and project members who need additional skills and knowledge to perform their roles effectively.
Risk Management	Involves five (5) basic steps: Risk Identification, Risk Analysis, Risk Treatment, Monitor and Report the risks, Document lessons learned.
Metrics	Project measurements, analysis and actions to effectively manage the projects from start to end. Objectively demonstrate the quality of product.
Tailoring	Provides guidelines for tailoring the process, lifecycle models, and tools that can be adapted to the project.

4.4. Implementation

Once the PFW meeting the organization's needs is defined, a plan was laid out to implement the processes into the projects. It was decided that a subset of representative projects is first identified as pilot projects. These projects were selected on a sample basis spread across the various types of projects. Plan also included methods to collect feedback from the pilot projects on the processes being implemented and use the feedback to fine-tune the PFW before rolling it out to the entire organization. A robust metrics program with data collection, analysis and metrics based decision making was put in place as part of continual improvement and defect prevention measures.

4.4.1. Pilot the processes

It is important that the processes defined are piloted with representative projects in each category such that the processes are usable and help meet the project objectives. The processes were piloted such that each process was experienced at least in one project. While piloting, the members of process definition team worked closely with the project team members so that the process implementation was experimented and monitored at every step and experience/feedback was gathered. The experience and feedback was collated, analyzed and based on the learnings, processes were fine-tuned. There were minor improvements made in process explanations, checklist items and template formats.

4.4.2. Develop PFW portal

In order to disseminate the process artifacts and assets, an intra-organization portal was setup. The portal acts as one-point store-house for all of the processes, procedures, templates, checklists, guidelines – that can be referred by the employees. It also stores training assets. The project-specific documents created, such as Plans, Reports, Metrics, Technical documents are also maintained on the portal. Portal contents have a need-based controlled access. The PFW is not maintained in hard-copy format.

4.4.3. Conduct PFW training and Roll out PFW

Role based training for the members of the organization was planned. One training module for each process was developed. Tips and FAQs on the process usage were included. A common module was developed to provide overview of new PFW to all the users. Then processes which are specific to one or more roles were delivered. The training was imparted for different roles such as Project managers/leads, System analysts/engineers, Designers/Developers, Testers, V&V engineers, Support functions.

4.4.4. Implement Metrics program

Metrics data collection is an important and integral part of the process implementation and is also used to measure the efficiency and effectiveness of the processes. Projects can also identify additional data parameters, and their collection frequency to further improve the quality of the outputs. Projects have a metrics analysis mechanism to derive intelligence from the data to improve the process and quality. Four basic data source categories were identified –Size, Schedule, Effort and Defect.

The sources for these data points were identified, data collection mechanisms and metrics computation techniques were defined. The metrics so generated would provide guidance on the efficacy and effectiveness of processes intended to enhance goal achievements and customer satisfaction.

4.5. Results and Analysis

As the processes were implemented, measurements were made on the project activities. The data was collated and metrics were generated. The metrics data collection, collation, and analysis is an ongoing process. Metrics data is a reflection of the process that is being used to carry out the activity. Metrics data provides a quantitative insight into the behavior of the processes. Some of the key metrics collected over a period of few months of implementation with data from 7 different projects are presented here. While Figure 3 gives the estimation accuracy in projects (ranging from 0.95 to 1.40 across projects) and Figure 4 gives schedule variance (ranging from 0% to 40%), changing requirements (0% to 15%) shown in Figure 5 seem to have an impact on both estimation accuracy and schedule variance.

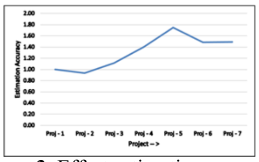

Figure 3. Effort estimation accuracy

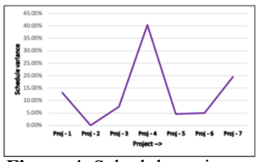

Figure 4. Schedule variance

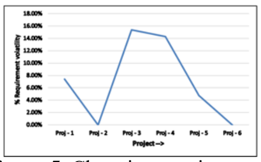

Figure 5. Changing requirements

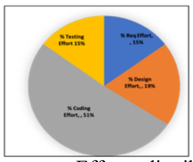

Figure 6. Average Effort distribution across different phases in SDLC (all projects)

Overall effort distribution across SDLC phases is presented in Figure 6. Figures 7 to 10 provide effort consumed by each phase as a percentage of overall effort in each project. Two interesting observations are made. One is an insight into the effort consumption pattern which is fairly uniform across projects and is very useful for estimation and resource planning. Second point makes us realise that less effort spent in requirement analysis and design possibly increases the coding effort.

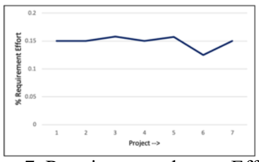

Figure 7. Requirement phase – Effort%

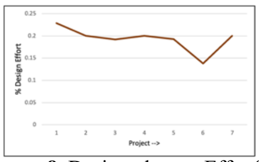

Figure 8. Design phase – Effort%

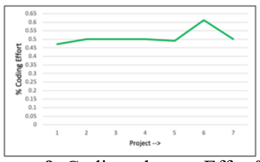

Figure 9. Coding phase – Effort%

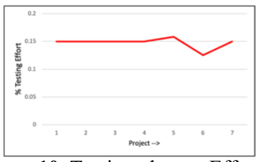

Figure 10. Testing phase – Effort%

5. Conclusion

These initial results are encouraging. This case study confirms that it is possible to implement a common PFW while catering to the specific mandates of industry standards IEC61508, ISO26262 and UL1998. As more project data gets collected over time and the metrics analysed, leading to continual improvement of the PFW to prevent repetitive defects at their source of occurrence will bring value. Use of the PFW by all projects, continuity in collecting project metrics, conducting defect analysis and incorporating process improvements is key to success. This would lead to better user experience, better customer satisfaction, improved employee morale and improved company image and profitability.

References

[1] IEC61508-3: International Standard on Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements. Edition 2.0; 2010.

[2] UL1998: Standard for Safety for Software in Programmable Components. Third Edition; Dec 2013.

[3] ISO26262: International Standard on Road Vehicles-Functional Safety. Second edition; Dec 2018.

[4] H. Lloyd and P. J. Reeve. IEC 61508 and IEC 61511 assessments - some lessons learned. 4th IET International Conference on Systems Safety, 2009, pp. 1-6.

[5] Gadila Prashanth Reddy, et al. Achieving ISO 26262 & IEC 61508 objectives with a common development process. Computer-Aided Developments: Electronics and Communication, CRC Press, 2019, eBook ISBN 9780429340710, (pp.247-255).

[6] Christopher Preschern, Nermin Kajtazovic, and Christian Kreiner. Applying and Evaluating Architectural IEC 61508 Safety Patterns. Lecture Notes on Software Engineering, Vol. 2, No. 1, 2014

[7] Akihiro Hata, et al. Using Hazard Analysis Method STAMP/STPA in Developing Model oriented Formal Specification toward Reliable Cloud Service. IEEE 2015 International Conference on Platform Technology and Service, Jeju, South Korea, 26-28 January 2015. pp 23-24.

[8] Andy Brazier, David Edwards, Fiona Macleod, Craig Skinner, Ivan Vince. Chapter 1 - Hazard and operability (HAZOP) analysis. Trevor Kletz Compendium,. Elsevier, 2021, pp 9-45, ISBN 9780128194478.

[9] Julieth Patricia Castellanos Ardila, Barbara Gallina, Guido Governatori. Lessons Learned while Formalizing ISO 26262 for Compliance Checking. Conference: 2nd Workshop on Technologies for Regulatory Compliance, Groningen, NL, February 2019.

[10] Renu Rajani, Pradeep Oak. Software Testing - Effective Methods, Tools and Techniques, 2nd edition, McGraw Hill Education, ISBN: 9789387432673 ; 2017

[11] Gerard O'Regan. Concise Guide to Software Testing. Springer, 2019. ISBN 978-3-030-28494-7

[12] Muhammed Basheer Jasser, Jamilah Din , Rodziah Atan, Yusmadi Yah Jusoh. The Measurement of Safety Criteria in Safety Critical Systems. International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No.1.4, 2019, pp 501-506.

[13] Vilela Jéssyka et al. Requirements Communication in Safety-Critical Systems. Proceedings - Workshop on Requirements Engineering, Recife, PE, Brazil, August 13-16, 2019.

[14] Tian, Jeff. (2005). Defect Prevention and Process Improvement. Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement, 2005. pp.223-236.

[15] Huang, B., Ma, Z., Li, J. Overcoming obstacles to software defect prevention. International Journal of Industrial and Systems Engineering, Vol. 24, No. 4, 2016, pp 529-542.

[16] Huang, Fuqun & Liu, Bin. Software defect prevention based on human error theories. Chinese Journal of Aeronautics. Volume 30, Issue 3, June 2017, Pages 1054-1070.

[17] Suma. V., T. R. Gopalakrishnan Nair. Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels. World Academy of Science, Engineering and Technology 42 2008.

[18] Ram Prabhat, et al. Actionable Software Metrics: An Industrial Perspective, April 2020, pp 240-249.

[19] Bagnara R., Bagnara A., Hill P.M. The MISRA C Coding Standard and its Role in the Development and Analysis of Safety- and Security-Critical Embedded Software. Lecture Notes in Computer Science, vol 11002. Springer, Cham. 2018