Safety Critical Embedded Software: Significance and Approach to Reliability

Shobha S Prabhu
Gas Turbine Research Establishment
Defence R&D Organisation
Bangalore, Karnataka, India
shobha.prabhukamath@yahoo.co.in

Hem Kapil
Gas Turbine Research Establishment
Defence R&D Organisation
Bangalore, Karnataka, India
hem_kapil2001@yahoo.com

Shashirekha H Lakshmaiah
Department of Computer Science
Mangalore University
Mangalore, Karnataka, India
hlsrekha@gmail.com

***Abstract*—Safety critical embedded software applications are developed for systems whose failures contribute to hazards in the system for safety of life. Such software, as a part of extremely critical component of any system, requires high reliability index in its design, development or maintenance. Enhancing reliability and thereby achieving best quality software is a concern for safety critical software. In order to build highly reliable software, attributes of quality that are applied at each phase of development lifecycle are necessary to be considered for improvement. Usage of formal method based software tools during the development improves overall quality of the software by removing ambiguities and early detection & removal of faults. This paper highlights the requirements and significance of reliability on the overall performance of the safety critical software, the approach to higher reliability through software project planning, development with standard methodical process, automation and configuration control. Further, this paper emphasizes on enabling safety and reliability into the critical software systems by adopting factors such as development process, formal methods and relevant tools in order to build continued confidence.**

***Index Terms*—Safety critical embedded software, software reliability, DO-178B, lightweight formal methods, automation**

I. Introduction

Critical embedded software is developed for the functioning of a system, whose failure can contribute to a hazard in the whole system either for safety of life or for achieving critical mission. Examples include aircraft engine control systems and point-of-care medical device monitors. As systems are intertwined with human lives, system design itself must address potential errors more effectively at early stages. This leads to defining safety measures, assuring safety process and reliability consideration during development in a systematic way, thereby consuming 200-300% more resources and effort compared to any other system.

Safety or mission critical systems are basically real-time systems in which the correctness of the system depends on the logical result of computations and equally on the time at which the results are produced. These systems are developed with embedded computer and a requirement oriented specific software. In totality, the objective of embedded system must be to implement functions that provide an element of control which is executed using optimum amount of code and with highest level of reliability. Software being a major component of these systems shall follow highly desirable unified engineering approach of development process, a set of methods and an array of tools for building complex systems in a timely manner with high quality through enhanced reliability [1].

In this paper, we are discussing the design and development of an in-house state of the art safety critical embedded software for Full Authority Digital Engine Controller (FADEC) system which provides engine control and monitoring of propulsion system over the complete flight envelope of the unmanned air vehicle. This is taken as the reference throughout this paper and all the experimental studies & technical activities are carried out on this software. Over the past few years the software is incrementally developed with several enhancements of features as well as different methodologies of development. Earlier versions of embedded software followed the DOD-STD-2167A standard for development where in safety and reliability aspects were considered as part of the system requirements and not as part of standard. In the latest version of the software, we have followed the DO-178B process with safety and reliability aspects alongside the usage of better methodologies. We would also like to share the observation on enhancement of reliability achieved with the improved process and methods.

The glimpses of this paper is given here: Section II explains the relevance of various concepts to the software project currently working on. Section III talks about the survey carried out on the similar work and the motivation behind this work. Section IV presents the methodologies adopted to implement the above concepts into the software product. Section V highlights the software quality improvement based on the merging of various concepts in comparison with the implementation of single concept. Section VI throws light on conclusion with some scope for future work.

II. Relevance to Current Work

Full Authority Digital Engine Controller (FADEC) system developed indigenously provides aero engine control and monitoring of propulsion system over the complete flight envelope of the unmanned air vehicle. Indigenous development of the safety critical embedded software depending on the system requirements and the state-of-art hardware platform is a notable activity for this FADEC. This software encompasses several unique features to enable the unit to work efficiently in terms of functionality and also reliably in terms of performance.

978-1-5386-5314-2/18/$31.00 ©2018 IEEE

Unique features include - execution of the expected functionality as per system requirements with hard real time behaviour, priority based preemptive scheduling to cater to safety requirements, optimum and well planned design strategies with standardized DO-178B software development life cycle process etc. The wide range of functionality covers features such as acquisition of analog & digital inputs, conversion of signals, control law logic, redundancy signal management, validation of signals, cross channel data link, voting algorithms, sensor modeling, multi rate control, fault classification with handling & accommodation, lane changeover logic, analog & digital outputs, communication over interface bus and built-in tests for health check of the unit. [2]

As the software is developed for the aero engine, it involves safety of human life. This makes the software to implement control laws for normal functioning as well as to take care of abnormal / emergency conditions. With this requirements, the software assumes the complexity and involves highest level of safety (Level A as per DO-178B guidelines). Following attributes of software development are considered further to enhance the quality through reliability improvement:

- Software project planning and management
- Good software engineering methodology
- Process automation and tool qualification
- Formal methods and effect on quality metrics

The below subsections highlight the concepts considered for the FADEC embedded software.

A. Software Development Process

Avionics industry and regulatory authorities have come to a consensus with stringent standard for avionics software through DO-178B [3] keeping in view the confidence in precise and safe functioning of software. The FADEC embedded software is developed based on this process with considerations on configuration management, tool qualification of auto code generator and application of lightweight formal method based software tools for testing, static analysis, traceability, verification, run time error analysis.

B. Software Reliability

Software Reliability, an important attribute of embedded software quality (which in turn effects the system performance), is the probability of failure-free software operations for a specified time period in a specified embedded environment. In order to build highly reliable software, there is a need to measure the quality attributes that are applied at each phase of software development lifecycle. The FADEC embedded software considers some of the metrics (detailed in section IV B) during the software development lifecycle which contributes to reliability measures.

C. Formal Methods

Formal methods are unambiguous, have mathematically based syntax & semantics and contain techniques for the specification, development and verification of software aspects of digital systems. The advantages of using formal notations include improving overall quality of the artifacts, removing ambiguities and imprecisions, enabling automatic analyses, early detection of problems and better maintainability. As a cost effective initial option, FADEC embedded software employed lightweight formal methods i.e., formal methods based commercial off the shelf computer aided software engineering (CASE) tools during various stages of software development lifecycle process.

D. Automation and Tool Qualification

Automation of development lifecycle of software plays a vital role, as time and correctness are important factors during the entire process. For the FADEC embedded controller software, two types of automation is used. First type is automation of coding with an in-house developed qualified tool and second type is off-the-shelf commercial tools for analyses, verification and configuration management. These automation is beneficial for improving software quality, consistency & correctness, faster process management and reduced manual effort.

III. Related Work

Several researchers have come up with developments in the fields related to safety critical embedded software. These literatures have revealed the limitations, improvements and future scope of work which is required to be carried out to strengthen the process of software development life cycle for achieving the higher reliability of safety critical embedded systems. Dr. Linda Rosenberg, et al. [4] discusses how NASA projects depend on software metrics to improve the quality and reliability of software products by identifying the critical areas where problems and failures may occur. Currit, P.A., et al. [5] describes the importance of defects and procedure for reliability certification before the product release to users. Lohar, Debasmita, et al. [6] discusses the strict reliability and safety constraints for complex software where in, system level reliability estimation is a combination of relationships between reliabilities of various components and the expected system level reliability. Eldred Nelson [7] explains the techniques for improving reliability through formal verification of functional capability concepts. Kashyap, S., et al. [8] proposes the software engineering metrics with deterministic quantitative model of failure patterns for reliability enhancement.

Milena Krasich, [9] presents the methodolgy to improve reliability with early planning and reliability assessment during the software development process. Yannick Moy, et al. [10], brings out the essence of certification standard for avionics software, which provides guidance and data concerning verification based software life-cycle processes. Dassault-Aviation and Airbus replaced part of testing with formal verification for their production environment which were proved to be time and cost-effective during repeated activities. An approach to attack a problem by employing formal specification and verification on the traditional software process is brought out by Qiu Fang, et al. [11]. The paper details the development of tools based on the formal method technique and would be a relevant research area in the safety critical software. David

P. Gilliam, et al [12] reiterates the necessity of an integrated approach which is a combination of several formal techniques in increasing the confidence in the verification of software quality attributes and in turn reducing vulnerabilities in software during the development and maintenance stages of life cycle process. Marilyn Wolf [13] awakens the software world by stressing on several high profile embedded software failures and forces in ensuring the software design related artifacts meet higher standards to reassure users that embedded systems are safe and reliable. N. Polikarpova, et al. [14] experiments extensively with testing against strong specifications with a favorable benefit to effort ratio. Timothy Wang, et al. [15] presents a prototype tool-chain based on auto coding and verification framework for control systems with domain specific knowledge transformed into formal specification language.

A. Motivation

In a nut shell, the literature review has revealed several limitations / future scope with respect to the researches carried out in the fields relevant to safety critical embedded software in terms of improvement on software development process, reliability, formal methods and automation of integrated process. Some of them are as follows:

- Improvements in Certification - Development of certified software products with reliability projection
- Improvements in software development process - Process model for development which is suitable for specific embedded safety-critical software
- Limitations of usage of formal methods - Cost and time limitations could be overridden with attempting and applying formal methods (through tools) during at least one phase of the software development life cycle
- Limitations of automation - Tools for automating the process with qualified code generator
- Improvement to the existing development process - Migration from DOD-STD-2167A development standard to DO-178B development process

This paper highlights the aircraft Full Authority Digital Engine Control (FADEC) safety critical embedded software development with DO-178B Level-A guidelines, review processes, application of lightweight formal methods for various analyses, auto coding from models, automation of change control process to generate safe and reliable software.

IV. Methodology

The following subsections detail the methodology in which the above mentioned theoretical aspects are applied to the FADEC software for aero engine application.

During initial stage of the project, gathering software requirements such as functional, non-functional, reliability, safety etc. from system requirement are the most important aspects of the application. With this information and the management focus on 4Ps (Process, People, Product and Project), the much needed technical environment, man power, the work elements, infrastructure is worked out for the software development. The basic component is taken as estimated source lines of code (SLOC) and COCOMO-II [16] model is adapted for the estimation of effort and development time. Based on the industry data, additional activities related to software are considered and effort related to them are included [17]. Scheduling strategies are derived with appropriate man power distribution and development time aligning with the project duration.

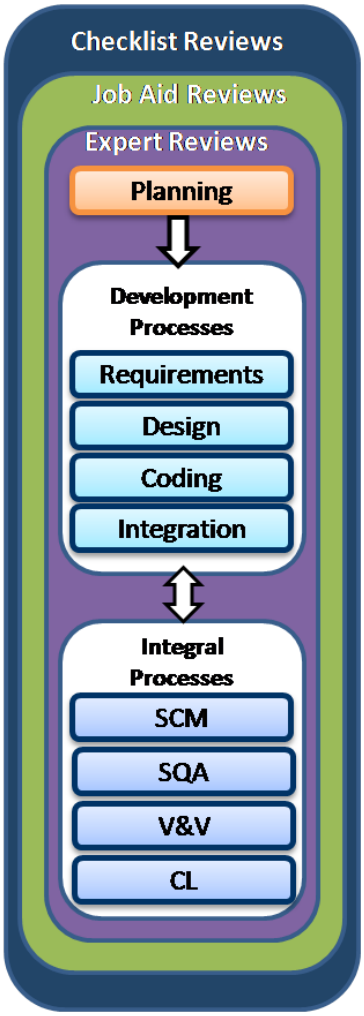

Fig. 1. Development process model

A. Process model based on DO-178B

DO-178B has become the de-facto standard for providing detailed guidelines for aviation software development processes that perform intended functions with a level of confidence in safety and compliance with airworthiness requirements. These guidelines are in the form of objectives, list of activities and reports generated as evidences. DO-178B guidelines include the planning process, software development processes (requirements, design, code and integration) and integral processes (software configuration management- SCM, software quality assurance- SQA, verification & validation-

V&V and certification liaison-CL). The set of activities for meeting the objectives cover the attributes such as compliance, compatibility, correctness, verifiability, traceability, completeness, conformance to standards and consistency [3]. These attributes are great reliability booster for FADEC safety critical embedded software. Fig. 1 gives a comprehensive visibility of the process model implemented for the software development.

Safety assessment is carried out for FADEC system and declared it as safety critical system with software design assurance level-A(DO-178B) which compliments to achieve the system failure rate as low as $1 * 10^{-7}$. This sort of software has more rigorous objectives and DO-178B is chosen as the development process model which enabled us to enhance the reliability during development itself. Since this project is a technology demonstrator under the Ministry of Defence, there is no review from Federation of Aviation Administration (FAA). Hence a combination of reviews of different scales is worked out to clear the software at every process of development lifecycle.

Further if one observes the intricacies of the processes followed in this figure, the reviews are integrated with the model at different levels- Expert review (reviews by experts in the field), Job Aid reviews (conducting software reviews as per FAA certification process) and reviews as per the consolidated international aerospace industry's checklist for safety critical software. The verifiability and bidirectional traceability at every stage of the development process is an added reliability builder. Various review activities are adapted with this model for each process and with varied scope to further enhance the reliability of the software. Verification at different stages ensures correctness and consistency. Mandatory conformance to standard is ensured for better quality of the artifacts. Following paragraphs detail the various processes.

1) Planning process: To adhere to DO-178B, the first process to enter the software development is planning process, which generates artifacts that help the project team to handle the smooth functioning of software development with the pre planning of the methods, development environment, tools and standards. This process also accomplishes relationship between the processes, their sequencing and focus on transition criteria.

2) Development process: The next process called development process covers requirements, design, coding and integration stages. Requirement stage ensures the artifacts are covering functional and interface requirements along with performance and safety requirements based on the user/ system requirements. High level requirements have been derived at this stage to completely cover the system requirements keeping in view the assessed safety.

Design stage refines the high level requirements into software architecture and the low level requirements- which will be used for implementing the source code. During this stage, the control function requirements are analysed along with redundancy signal management and fault detection & voting logics in the form of Simulink design models. The combined low level requirements are derived at the end of design stage of software.

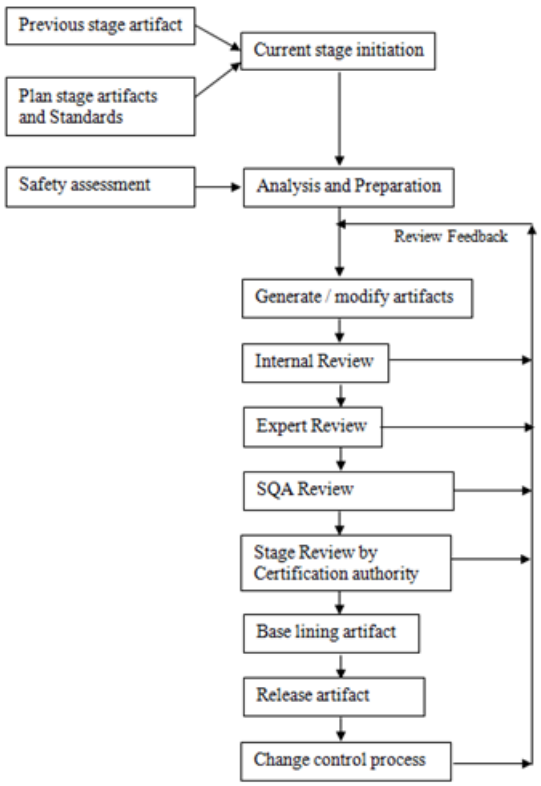

Fig. 2. Flow chart for each stage of development process

Next comes the coding stage, where the low level requirements are implemented using a high level language which is compliant to coding standard defined during the planning phase. Source code is developed with a focus on traceability, verifiability, consistency and correctness as per low-level requirements and adhering to software architecture during this stage.

Integration stage is responsible for generating the executable object code which is generated using the source code, linker and loader for the target computer of the control system. This is tested for its compatibility and correctness with the target computer in all aspects.

A generalized flow chart as per Fig. 2 is worked out to give the clarity on how at each stage artifacts are generated, reviews are conducted, base lining is fixed and change process is configured for the FADEC software.

3) Integral processes: The next process called integral process covers all the umbrella activities- software configuration management, software quality assurance, verification & validation and certification liaison.

Verification is nothing but the technical assessment of the results of software development processes. Verification is carried out as per the software planning process and is a combination of reviews, analyses and testing. This activity ensures the traceability, completeness and correctness along with detecting and reporting errors.

Configuration control process which is a part of the integral process is highlighted to ensure that expected changes need to follow a stringent process in order to maintain all the configuration items under control. Configuration items are data or evidences produced during the software life cycle activities to plan, direct, explain, define or record. This data is not only used during the software life cycle processes but also required for certification and post-certification modifications of the software product in a consistent manner. This process exists when the software product is accepted, and continues throughout the service life of the safety critical system. Configuration identification, software change control, baselining and archiving of the software life cycle data are major activities of this process. The systematic configuration control process used in our FADEC embedded software provides:

- Configuration items of the software which are defined and controlled throughout the software lifecycle
- Control of process inputs and outputs that ensures consistency and repeatability of process activities
- Establishment of baseline
- Handling of problems with changes recorded, approved, and implemented with the ability to consistently replicate the Executable Object Code (EOC)
- Confidence that configuration items are secured for physical archiving, recovery and control

B. Reliability and metrics

A software metric is a standard of measure of a degree to which a software system or process characterises a property.

At each stage of the development process of FADEC embedded software, we have defined metrics which aided us in enhancing the attributes of reliability directly or indirectly. Direct measures of reliability attributes used by us are Mean Time To Fail (MTTF) and Mean Time To Repair (MTTR). We have identified potential areas of problems and rectified them at early phases, which prevented ripple effects at later stages and increased MTTF. Fig. 3 shows that the maturity of planning, requirement, design and coding over the series of reviews has increased clarity and reduced the probability of failure, which leads to the increase in MTTF. Similarly, our software versions are bidirectionally traceable from requirements through software build and maintained through configuration control decreasing the MTTR drastically. Indirect measures of reliability attributes used by us which affect the software reliability during the software lifecycle are [18] given below:

- **Product metrics:** This depicts whether the product is good enough for usability, maintainability, reliability & portability. Ex. Lines Of Code, Test coverage metric, cyclomatic complexity
- **Project management metrics:** This depicts whether the product is able to complete on time, within cost and with the desired quality objectives. Do-178B development process and configuration management process helped in achieving better reliability within the cost and time limits.
- **Process metrics:** Since the quality of the product is a direct function of the process, better process is demanded for estimating and enhancing reliability. Process metrics described the effectiveness and quality of the process in our project includes number of defects found during testing, maturity of the process etc.

As seen from the above paragraphs, the avionics software developed by us follows the internationally acclaimed DO-178B standard process keeping reliability in view. We have conducted several reviews with varied scope at each stage of the software development for increasing the quality. The review comments are taken as the key index to evaluate the maturity of the artifacts at each stage.

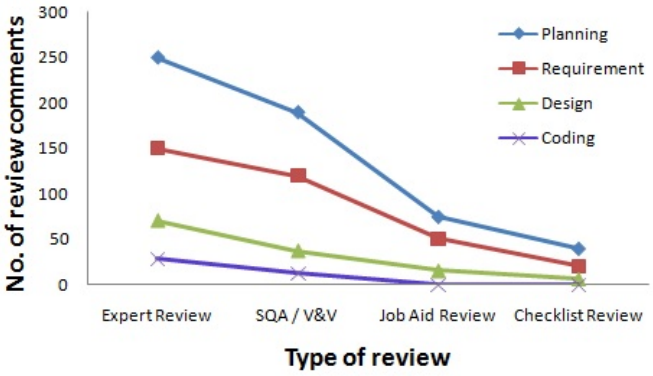

Fig. 3. No. of review comments Vs. Type of review

From Fig. 3 it is evident that, each stage has matured over various reviews during the period of development. The complexity of review is also in the increasing order from expert review to checklist review. Pending review comments will be resolved at the appropriate time during the final integration and testing of the system.

C. Automation and Qualification

Automation of code is one of the initiatives taken to ensure uniformity, reduction in turnaround time, avoiding manual error and confidence in correctness. For FADEC embedded software, we had a requirement to convert the control schedules, hardware related gains, validation limits and control law functions into C code. The requirements are received as a combination of Simulink models & Microsoft Word documents and the C code generated is expected to be compliant to coding standard defined during the planning phase. As no such readymade tools are available in the market, we had developed an in-house tool to convert the combined requirement into C code [19]. The code generated through this tool is also compliant to Motor Industry Software Reliability association (MISRA) C safe coding standard. The auto code generator tool contains 2 important parts, one for constants & schedules and the other one for converting Simulink models. The first part is achieved through parsing the MS Word document and converting into appropriate C file format. The second part is achieved by parsing the Simulink model file, understanding the

keywords and converting them into appropriate C code compliant to coding standards. The same tool can be modified for different applications depending on the formats of requirement keeping the base as foundation.

As per DO-178B, if any tool is used for automating the process, it is required to be qualified if the subsequent processes are eliminated or reduced without its output being verified. The use of software tools to automate software life cycle process activities can help achieve consistency and correctness efficiently. Confidence level builds up when the tool gets qualified and used for elimination of the processes. Since the auto code generator mentioned above is a deterministic tool, it is eligible for qualification.

As given in Fig. 4, the control laws developed as a Simulink model is fed to two different auto code generators, one in-house developed and the other one is Mathworks itself. The test cases are generated through the Simulink design verifier and test outputs are generated through two different off-the-shelf test tools. The outputs of the testing are compared and qualification is determined.

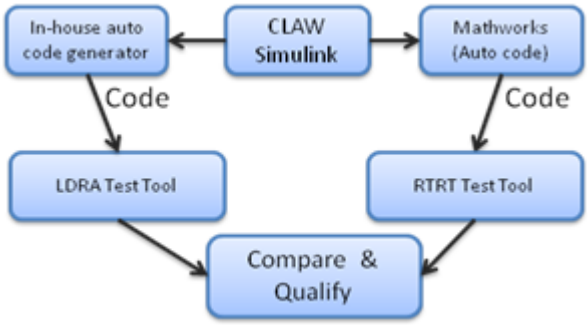

Fig. 4. Tool Qualification process

D. Lightweight Formal Approach

Even though formal systems are attractive in theory, their practical implementations have weaknesses such as limitations on computational models, cost factor, incredibly complex language usability etc. This reasoning has led to the usage of lightweight approach to formal methods. Lightweight approach emphasizes on application development completely and formal methods partially, thereby bringing greater benefits at reduced cost. Commercial off-the-shelf tools support application of formal methods to software development during various stages of lifecycle.

Dino Mandrioli [20] adds his modest personal view in support of formal methods and concludes that instead of having pessimism, attempting the formal method during any stage of implementation would bring a partial victory to the developers. The lightweight approach, which fills the gap between the necessities of engineering and the goals of formalisation, provides a good compromise and is found to be the most productive route for this particular software.

The application of formal methods is in the beginning stages in FADEC embedded software. Currently lightweight formal methods are used in improving the reliability by using formal method based software tools such as LDRA, ASTREE, Stack analyser etc. for analyses and verification at different stages. Since formal method based tools are mathematically based and the verification is based on the exhaustive set of mathematically derived test cases, confidence is increasing while building reliability in the software.

V. Comparative Study

When DO-178B is compared with other software standards such as MILSTD-498, DOD-STD-2167A, IEEE/EIA-12207, IEC 61508, it is evident that other standards deal with certain aspects of software development process covered by DO-178B and none of them has been found to provide complete coverage of DO-178B objectives especially safety related guidelines. Our experience with previous similar projects which were developed using DOD-STD-2167A or IEEE-12207, considered safety aspects partially as a part of embedded controller of aero engines and not as part of software standard for safety. Due to this, software was not 100% traceable to the safety related requirements. This led to incremental software versions over a period of several aero engine runs with additional time and cost implications. When the current software is developed using DO-178B, the much expected safety and reliability is covered while defining the requirements itself when the mandatory safety guidelines were followed as a part of the standard. This streamlining process is benefitted the developer as well as user equally in terms of time and cost.

Reliability considerations were not part of older software standards such as DOD-STD-2167A with which our previous projects were developed. Reliability factors are part of DO-178B process which is followed in our current project along with multiple comprehensive reviews. Complete traceability and enhancement of end-to-end quality is observed immensely. Automation process and its qualification devised for the current project brought the software to higher quality with minimum turnaround time and reduced manual errors during software change management. Formal methods based software tools are utilised for verification and analyses in order to improve the confidence on testing.

Safety critical software developed by us for aero engines in previous versions did not consider all of the above reliability enhancing factors. In the current version, the combination of better considerations of process or methods & tools, automation enforced on the current software development contributed greatly to the higher level of system safety and reliability as evident from the previous sections.

VI. Conclusions and Future Work

In this paper, we have highlighted the significance of reliability on the overall performance of the safety critical embedded software for an aero engine application and the experiments to accomplish it by various methods and improvements. Further we discussed the identification of reliability metrics and attempts to improve reliability through software development with standardised process, automation, reviews,

configuration control etc. This paper emphasizes on enabling safety and reliability into the software by adapting highly desirable factors such as process, methods and tools in order to build continued confidence into the safety critical software systems.

As a cost effective first step, we have currently emphasized on lightweight formal methods for verification and analyses on the application oriented area of software. This helped us in achieving a step ahead in level of quality enhancement. Extensive formalisation of specification and verification to produce the entire embedded software in a formal manner is required to be carried out in the future to further increase the level of satisfaction in safety, reliability and thus better quality.

ACKNOWLEDGMENTS

The authors are thankful to Director, Gas Turbine Research Establishment, DRDO who gave permission to publish this paper. The authors also thank all those personnel who are associated with the work related to this paper.

REFERENCES

[1] Roger S. Pressman, Software Engineering a practitioners approach Seventh edition, The McGraw Hill Publications, 2011
[2] B.Githanjali, et al., Full authority digital engine controller for marine gas turbine engine, Proceedings of ASME Turbo Expo 2006
[3] RTCA DO-178, Software Considerations in Airborne Systems and Equipment Certification, RTCA and EUROCAE, 2011
[4] Dr. Linda Rosenberg, et al., Software Metrics and Reliability, BEST PAPER Award in 9th International symposium, November, 1998.
[5] Currit P.A., et al., Certifying the reliability of software, IEEE Transactions on Software Engineering, 1986, (Classic paper)
[6] Lohar Debasmita, et al., Integrating Formal Methods with Testing for Reliability Estimation of Component Based Systems, IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2015
[7] Eldred Nelson, Estimating Software Reliability From Test Data, Published in Micro electronics and Reliability Vol. 17, 1978
[8] Kashyap S., et al., Analysis and Ranking of Software Engineering metrics, 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015
[9] Milena Krasich, Modeling of SW reliability in early design with planning and measurement of its reliability growth, IEEE Conference on Reliability and Maintainability Symposium (RAMS), 2015
[10] Yannick Moy, et al., Testing or Formal Verification: DO-178C Alternatives and Industrial Experience, IEEE Software Issue : May/June 2013
[11] Qiu Fang, et al., A new approach for developing safety critical software in automotive Industry, 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2014
[12] David P. Gilliam, et al., Application of Lightweight Formal Methods to Software Security, 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, IEEE Computer Society 2005
[13] Marilyn Wolf, Embedded Software in Crisis, COMPUTER Magazine Published By The IEEE Computer Society, January 2016
[14] N. Polikarpova, et al., What good are strong specifications?, 35th IEEE International Conference on Software Engineering , 2013
[15] Timothy Wang, et al., An application of a prototype credible auto coding and verification tool-chain, EEE/AIAA 33rd Digital Avionics Systems Conference (DASC), 2014
[16] COCOMO II - Model Definition Manual Version 2.1, 1995 2000 Center for Software Engineering, USC
[17] Handbook for Software Cost Estimation, JPL D-26303, Rev. 0, 2003
[18] Gurpreet Kaur, et al., Software Reliability, Metrics Reliability improvement using agile process, IJISET, Vol 1, Issue 3, May 2014, ISSN 2348-7968
[19] Shobha S. Prabhu, et al., An Automated Change Process for Embedded Controller Software of a Full Authority Digital Engine Control System, International Journal of Computer Applications (0975 8887), 2016
[20] Dino Mandrioli, On the Heroism of really Pursuing Formal Methods, IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering, 2015