Effectiveness of Software Metrics on Reliability for Safety Critical Real-Time Software

Shobha S. Prabhu and H. L. Shashirekha

October 15, 2020

Effectiveness of software metrics on reliability for safety critical real-time software

Shobha S. Prabhu[1] and H.L. Shashirekha[2]

[1] Corresponding author
[1] Gas Turbine Research Establishment, Defense R&D Organisation, Bangalore, India
shobha.prabhukamath@yahoo.co.in
[2] Department of Computer Science, Mangalore University, Mangalore, India
hlsrekha@gmail.com

Abstract : Safety critical software is a key component of any critical system and whenever there is a failure in this software, the system malfunctions with effect on safety of life or mission. Reliability is one of the quality factors and performance evaluator for this critical software. High reliability is expected of such software in its design, development and maintenance in order to increase the quality of the software for the system. Reliability metrics are derived for this kind of software during the planning phase and enhancement of reliability metrics is achieved during the design & development phase. When the software is being designed, the reliability metrics are taken as the factors of foundation on which the software is built. In this paper, the software metrics which form the basis for proving their effectiveness on reliability for airborne engine control safety critical software are described. Development process based on these metrics not only influenced the reliability enhancement of the critical software but also improved the performance and efficiency of the embedded real-time system.

Keywords - Safety Critical Software, Software Reliability, Reliability Metrics

1 Introduction

An embedded computer consisting of real-time software based on specific requirements with deterministic timing schedule constitutes the safety critical systems. Real-time systems are considered to be functioning precisely when the system output is logically correct along with meeting the strict timelines, regardless of the system load or adverse environmental conditions. Overall, these systems should behave in a predictable way irrespective of unpredictable external or internal stimulus. Embedded software is developed for the functioning of these systems where the failures can critically affect the systems in terms of safety of life or mission. Examples of such software are present in the field of medical and aircraft systems, where defects directly cause loss of life / mission. This kind of software keeps the reliability factors as the key factor during various phases of software development lifecycle such as planning, design, development and testing. Reliability, being an important quality factor of software, plays a vital role in the efficiency of these systems. Software Reliability Management talks about the process of reliability optimization through emphasis on software error prevention, fault detection / removal and use of measurements to maximize reliability irrespective of project constraints [1]. Metrics is a measurement of the level by which a system, component or process possesses a given characteristic / feature. Metrics help in getting a measurement of testability, maintainability, clarity, complexity, consistency, modularity and reliability. The goal of software metrics is to control and efficiently identify the essential parameters that affect software development [2]. In this paper, appropriate metrics are identified, studied and analysed to check their effectiveness on reliability for safety critical real-time software.

As the definition goes, software reliability is the probability of failure-free operation of software (in a system) over a specified time within a specific environment for a specified purpose. According to the above statement, reliability highlights the availability of the system with continued performance of the software with respect to functionality along with external conditions with which it is operated during the complete operational period. This shows that the reliability depends on various measurable quality factors, which could be improved during any / all the stages of software development life cycle to obtain a better performance, functionality and safety [3]. Application of quality attributes at all phases of the development life cycle highlighting error prevention, specifically in the early lifecycle is the foundation of building high reliability software. Metrics are needed at each development phase to measure applicable quality attributes [1], which reflect the characteristics of reliability. The safety critical software is expected to have proactive fault tolerance and correction of identified design errors through the stringent development process while keeping strict vigil on the quality / reliability metrics.

The glimpse of this paper is: Section 2 explains the relevance of reliability and metrics considered for the safety critical software for the case study. Section 3 highlights the motivation obtained as an outcome of the survey carried out on the similar work. Section 4 presents the strategies followed in implementing the above concepts during the design & development of the software lifecycle. Section 5 highlights the effectiveness of metrics on software reliability improvement and thereby on the software quality improvement. Section 6 concludes the methodologies with future scope for expansion of this work.

2 Current Work

Case under study is Safety Critical Software developed for Embedded Real-Time Controller (SACSERC) system which provides engine control and monitoring of propulsion system over the complete flight envelope of the unmanned aerial vehicle [4]. This embedded software features the complete aero engine control functionality along with efficient performance, safety and reliability. Unique features of this software include - execution of the expected functionality as per system requirements, hard real-time behaviour with scheduling methodologies for deterministic timing requirements, optimally well planned software design strategies, complete traceability from system requirements to embedded object code etc. Software of this system being safety critical in nature is expected to have Mean Time To Fail (MTTF) of 1 Lakh hours (continuously running for more than 10 years). This will lead to a failure once in 10^5 hours with a reliability score of more than 0.9. SACSERC being an embedded system, its reliability depends on the reliability of both software and hardware. As the software is being developed for the safety critical system, it is necessary to embed enormous safety and quality features in it. This enforces the software to implement safety and emergency features alongside normal functioning to take care of abnormal events with high reliability. Hence, airborne DO-178C process with Level A (highest level of safety) along with its in-built stringent guidelines is followed in achieving better reliability by minimizing faults / failures [5]. The software reliability attributes are measured so that they can be monitored and enhanced if required. Until now, there do not exist a single good method of measuring software reliability for softwares with varied scope and nature. Sufficient experimental studies are carried out on the SACSERC software to identify the metrics, prove their effectiveness on software reliability and enhancing the reliability by modifying the software through moderating the metrics is discussed in this paper.

Software failures are one of the most dominant causes of failures in today's critical systems. System reliability directly depends on the software reliability and in turn on failure free functioning of the software. Failures are caused by faults which occur during different software life-cycle phases and it is required to eliminate these faults to build a reasonably reliable system. In reality, even though it is not possible to eliminate all the faults in the software, usage of sound software engineering process helps in producing better software in terms of reliability. The top level metrics which are considered for the case study are Mean Time To Fail (MTTF) and Mean Time To Recover (MTTR) which are directly linked with the failure density and recovery time. Failure related incidents and their recovery duration need to be minimized in order to increase the MTTF and decrease the MTTR. Identifying potential areas of problems and rectifying them at early phases prevents ripple effects at later stages and increases MTTF [4]. Hence, the metrics which affect the factors related to MTTF and MTTR are studied for their impact and attempt has been made to moderate them. These metrics are identified through the measurement of structural complexity, inter relationship among the software components and transparency of software modules in the case study. Following are the metrics considered:

- ***Cyclomatic complexity*** - A metrics which gives sneak peek into the design aspect of the software which has indirect influence on testing.
- ***Coupling*** - A metrics which measures modularity as well as dependency factors of software modules.
- ***Percentage of comments*** - a derived metrics which is a measure of clarity and dependent on Lines of Code (LOC) (complete coded lines along with comments excluding the blank lines).

Overall, the intention of this study is to enhance the reliability of the software by moderating these metrics which contribute to MTTF and MTTR through restructuring the software architecture during software development lifecycle.

3 Survey on Related Work

Even though research goes on continuously in the field of software reliability, till now there is no conclusion about a common methodology using which improvement can be done in reliability for all types of software. It is specific to each type of software and is the responsibility of the designer or developers to find a suitable methodology. In this context, the safety critical software which is developed for the airborne system demands reliability enhancement as a continuous effort to establish improvements in the reliability. A number of literatures have been studied to derive suitable methodology for the work presented here.

To improve the quality and reliability of products, projects depend on software metrics for identification of critical areas where problems or failures may occur. Identification of these metrics which help in detection and correction of requirement faults are of utmost importance for prevention of errors at later stages of the life cycle is opined by Linda Rosenberg [1]. Lockhart et. al., [6] highlights the fact that consistent and concise system specification removes design errors prior to implementation which is crucial for developing reliable software systems for safety critical applications. These techniques also include static analysis for the removal of development errors before the release of the software in order to build better reliability into the critical embedded systems.

Most important and sought-after characteristic of software that is accurate and trustworthy is its reliability. Hence the evaluation of software engineering metrics with deterministic quantitative model of failure patterns for reliability enhancement is proposed in [2]. Milena Krasich [7] presents the methodology to improve reliability through early planning and assessment during the software development process. It highlights the fact that faults which have a high impact on reliability are discovered early in design, analysis and test so that reliability improvement can be attempted with identification and correction of potential faults at every phase. Certification standard for avionics software, which provides guidance and data concerning verification based software life-cycle processes is the essence given by Yannick Moy et. al., [8]. QiuFang et. al., [9] have brought out the necessity of achieving higher reliability by reducing faults through keeping eye on the metrics of software product especially in the case of real-time safety critical software.

In a nut shell, the literature survey has revealed the impact of metrics and its relevance to reducing failures in the study of reliability of safety critical real-time software. In the context of airborne software, several areas were focused in terms of evaluating the improvement on reliability in order to reduce the MTTR and increase the MTTF. Keeping in view of the metrics at different software phases, removal of remotely occurring faults becomes prime importance in increasing the reliability.

This paper highlights the software metrics for SACSERC, their effectiveness and the strategies adopted in order to enhance the reliability of the safety critical embedded software.

4 Methodology

The following paragraphs detail the methodology adopted for the above mentioned strategies in the context of SACSERC for avionics application.

Working in tandem with DO-178C based software life cycle processes, the quality and reliability are considered as basic non-functional characteristics in the case study. During the development of software, importance is given to reliable, traceable and error free code by which one can ensure better quality of software. It is necessary to measure and monitor the metrics, so that impact of the same on reliability could be studied. At the same time, these metrics provide the developer with certain details about the structure of software architecture, clarity and testability of modules which are useful during maintenance period. During the development phase, it is also mandatory to expose the software to rigorous static and dynamic analyses, which in turn help in exposing the errors / faults (which otherwise may escape testing techniques) prior to releasing the object code. A combination of these analyses alongside keeping a watch on the software metrics is a better option while focusing on the quality attributes [10] such as Reliability, Maintainability, Testability and Efficiency.

4.1 Cyclomatic Complexity

It is generally accepted that more complex modules are more difficult to understand and have a higher probability of defects than less complex modules. Thus complexity has a direct impact on overall quality specifically on maintainability [1]. Cyclomatic complexity is a metrics which is used to indicate the complexity of a program by directly measuring the number of linearly independent paths through a program's source code. It is a primary measure of the soundness of the module, which also is a strong indicator of testing effort. It is considered as a good testability metrics because it determines the number of test cases needed to test all paths in a software module (test coverage). The high complexity programs contain more errors and detecting them is more difficult through testing [11]. Hence developers who would like to simplify testing often re-write the code for reducing complexity. This also may affect the higher coupling factor. The developer has to work out a suitable and optimum software architecture for reducing complexity as well as reduce coupling together to maintain good cohesion.

The SACSERC was developed as per the software design and coding standard specifically prepared for this software. The value of cyclomatic complexity stated in these standards was as low as 9 initially. During the software design phase, it was found that 9 was too less to accommodate few of the important and complex control function requirements. If the cyclomatic complexity number is increased to 15, it was logically difficult to handle too many test cases to complete the 100% test coverage during normal testing or regression testing. It is also observed that the maintenance becomes bit difficult with more complex module. There needed a tradeoff between the complexity and all other dependant factors in order to have the increased reliability. The restructuring of software architecture was carried out to optimize the complexity, maintenance cost / time and testing / verification efforts.

Fig. 1 depicts the effect of cyclomatic complexity on testability and maintainability metrics which in turn affect the reliability. Exercises are carried out by moderating different software architectures to finalise the cyclomatic complexity. It is observed from the figure that cyclomatic complexity of 9 (initial) has resulted in a higher testing effort and MTTR. To ease the situation, it was decided to go for cyclomatic complexity of 15. This again complicated the testing as well as MTTR as evident from the Fig. 1. With several experiments, the cyclomatic complexity was finalized to 12, which gave the minimum testing effort (of 22 hours) and least MTTR (of 120 minutes) during a demand of minor change in the software.

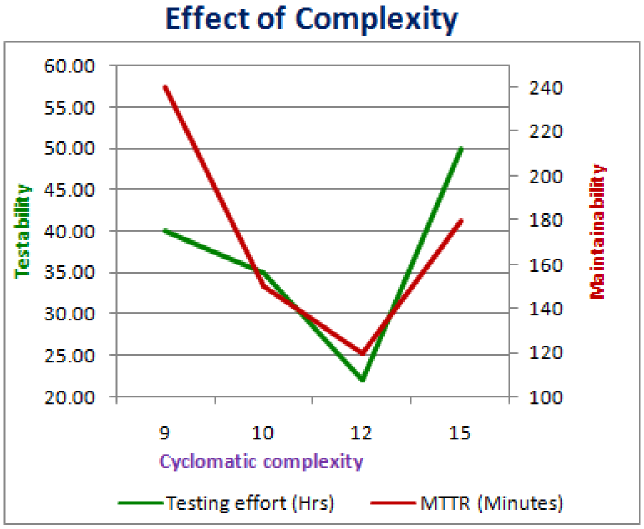

Fig.1. Effect of complexity on reliability factors

4.2 Coupling

Coupling is the degree of interdependence between software modules or the strength of relationships between modules and represented as a measurement of modularity. Low coupling & high cohesion is often a sign of a well-structured software product with a good design and supports the general goals of high readability and maintainability. In order to have the complete structural coverage, two types of couplings are discussed here: *Data Coupling and Control Coupling*. Both these coupling studies are intended to identify any code structure that was not exercised during the requirements-based testing [12]. In DO 178C process, Data and Control Coupling objectives are to provide a measurement and assurance of the correctness of software components, interactions and dependencies between modules [13]. They can indicate issues during integration tests such as data loss across an interface, components adversely affecting each other, sub modules not contributing to desired functionality and problems faced with global data against the intended & designed requirements.

Data Coupling represents the dependence of a software module on data not exclusively under the control of that software component. Control Coupling represents the manner or degree by which one software module influences the execution of another software component [12]. Extensive exercises have been carried out to decide on the coupling percentages, specifically keeping reliability in view during requirement based testing and coverage analysis of the software. For complex software like SACSERC, deciding on the upper limit for coupling study is a major task as its impact is directly felt on the subsequent activities in the software lifecycle such as review, analysis, testing and maintenance. It has been decided to have an upper limit of 20% for data coupling and 15% for control coupling to check if the data / modules are loosely or tightly coupled. These upper limits are finalised on the basis of low software maintenance, high cohesion among modules, low testing effort and low MTTR.

In SACSERC, there are 614 modules and 205 variables. The architecture of the software is designed in such a way that modules need to be loosely coupled with high cohesion of individual modules. Data or control coupling are categorized as given below:

a) Loosely Coupled Data (or Control): If by changing a specific variable (or module), the number of modules getting affected is less than 20% (or 15%) of the total number of modules, the specific data (or control) is said to be Loosely Coupled.

b) Tightly Coupled Data (or Control): If by changing a specific variable (or module), the number of modules getting affected is more than 20% (or 15%) of the total number of modules, the specific data (or control) is said to be Tightly Coupled.

Data coupling categories for all 205 variables for SACSERC software are worked out and the result is depicted in the Fig. 2. From this it can be noticed that 194 variables (data) (95%) are having coupling percentage of less than 10 and coupling percentage of 10-20 is observed in 11 variables (data) (5%). All variables are within the upper limit of data coupling percentage which is 20%. This shows that the dependency of modules on data is well within the limit and changes to the software can be handled with a low MTTR and low testing effort.

Control coupling categories for each of the 614 modules for this software are worked out and the result is depicted in the Fig. 3. From this it can be noticed that 589 modules (96%) are having coupling percentage of less than 10 and 25 modules (4%) are having coupling percentage of 10-15. All modules are within the upper limit of control coupling percentage which is 15%. This shows that the dependency of modules on other modules is well within the upper limit and changes to the software can be handled with a low MTTR and low testing effort.

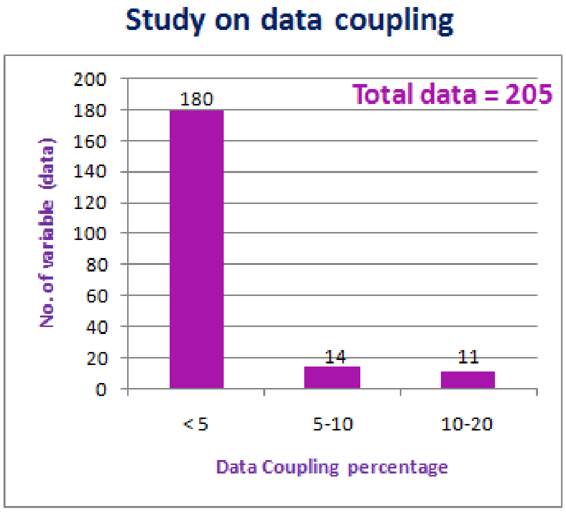

Fig. 2. Effect of data coupling on reliability factors

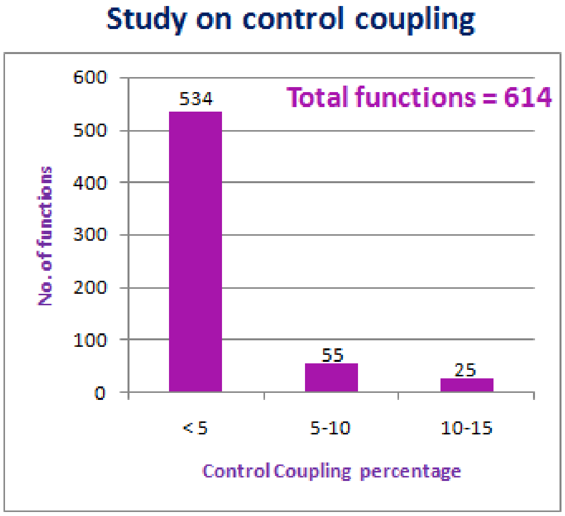

Fig. 3. Effect of control coupling on reliability factors

4.3 Percentage of Comments

This is a measure of clarity which gives the details of each module. SACSERC is developed in C programming language as it is flexible, mature and easily portable. During the analysis of SACSERC, it was found that few metrics are dependent on Lines of Code (LOC). Analysis is carried out on those metrics which are keeping

LOC as their prime factor and showing their efficacy on the software reliability indirectly. One such metrics is percentage of comments in a module. This is computed with the ratio of number of lines of comments to LOC of a module and converted into percentage. Even though there are many definitions of LOC [11], coded lines along with comments excluding the blank lines are considered as the LOC for this case study. Comments are written for declaration part as well as code part of the module separately to give transparency among development team members. Comments are useful in deriving the test cases at the time of verification and clarity during maintenance. The comments are bridging the gap between the developer of the module and other personnel involved in various stages of software development life cycle. The comments in a module play a vital role during structural coverage activity (intends to identify any code structure that was not exercised during the testing) and further till the active life of the software.

During the initial stages of implementation of SACSERC, minimum comments (5-10%) were introduced for all modules as it is assumed that detailed software design gives sufficient clarity. During the verification stage, confusion aroused during the testing and analysis phases due to insufficient and minimum comments. This affected the cost, time and development effort of the project. Hence, detailed and explanatory comments were introduced to eliminate any miscommunication among personnel involved in different phases of development. Further, lower (5%) and upper (40%) limits were introduced for the comments in order to control the optimum and necessary details only to check the completeness and correctness of implementation. Comments played a significant role during the static and dynamic analyses in eliminating hidden errors, in turn increasing MTTF. Detailed comments also helped in reduction of time during the maintenance phase which in turn reduced the MTTR. As code coverage (degree to which the source code has been tested) of 100% is mandatory for safety critical airborne software, optimum comments for a module helped in achieving structural coverage analysis in a shorter duration.

We have a total of 614 modules as seen from Section 4.2. Among them, 12% of modules (73) have below 5% of comments; 57% (350) have 5-20% of comments and 31% (191) have 20-40% of comments. To arrive at this final decision on the percentage of comments, several aspects were considered. Modules with a complex logic may require more comments and short modules with simple logic may not need much description. Some modules are clearly explained in the design phase and hence they may not need more description. All these aspects contributed for the study to decide on the amount of comments needed for each of the modules. Fig. 4 depicts the effect of this percentage of comments on verification and maintenance metrics. It is also observed that as the percentage of comments optimised from 5 to a range of 5-40, coverage increased from 65% to 100%. Meanwhile the MTTR reduced from 300 minutes to 120 minutes. It is derived that, when the clarity is better, the testing becomes easy and it is possible to get the test coverage easily. Similarly, right percentage of comments help in achieving the software maintenance in a shorter time when there is complete clarity. This shows that the percentage of comments is having direct effect on verifiability as well as maintenance and hence contributed immensely in enhancing reliability of the said software.

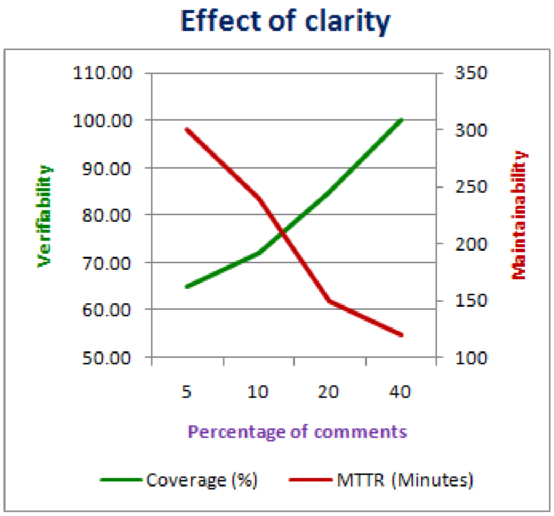

Fig. 4. Effect of clarity on reliability factors

5 Comparative Study

When the reliability considerations are not part of software development process, the metrics are computed for record of statistics rather than utilizing them for software quality improvement. Hence the usage of metrics is never highlighted. Since reliability is part of DO-178C process which is followed in the current case study, measurements of critical features through metrics were made use for this study. The reliability enhancements were indirectly evaluated by measuring, monitoring and improving these metrics during the development lifecycle of this critical software. The above paragraphs show the efforts made in exhibiting the effectiveness of software metrics on reliability for safety critical software. Metrics such as Percentage of comments, Cyclomatic complexity and Coupling were studied and moderated to achieve the improvement in reliability (and also quality) through reduction of MTTR and increase in MTTF. The numbers which are moderated for each of these metrics may not be suitable for all types of software. In particular, even the safety critical software for different applications may not adapt to the same metric values. Normalisation of these metrics and their values is not uniform for different varieties of softwares.

6 Conclusions and Future Work

Reliability is an indicator of quality and performance of safety critical real-time software for an airborne system and is enhanced through some of the software metrics. In this paper, the metrics which are affecting the reliability are studied thoroughly and moderated (either increased or decreased) in order to enhance the reliability of the system significantly. Metrics such as percentage of comments has been considered as a strong clarity metrics which is useful in all phases of development lifecycle. Cyclomatic complexity and coupling, being indicators of testability, complexity, modularity and maintainability are also considered for improvement. Attempts were made to improve these metrics and show their effect on MTTR and MTTF of the software. This paper emphasizes enabling improved safety and reliability into the software by means of study on few affecting metrics and observed their effectiveness.

As a future work, process related metrics such as percentage of bidirectional traceability with object code, compliance to coding standards, absence of dead code, existence of run time errors may be explored and analysed to bring the quality and reliability features of the airborne real-time embedded safety critical software to a improved / higher level. Feasibility study may be attempted to normalise the metrics for a particular type of software, if not for general embedded safety critical software.

Acknowledgements The authors are thankful to Director, Gas Turbine Research Establishment, Defense R&D Organisation, who permitted them to publish this paper. The authors thank all the personnel who are associated with this work.

References

1. Dr. Linda Rosenberg, Ted Hammer, Jack Shaw: Software Metrics and Reliability. In: IEEE International symposium on Software Reliability Engineering, 301-286-0087, 301-286-7475, 301-286-7123 (1998).
2. Kashyap, S., AshishTripathi, Kapil Sharma: Analysis and Ranking of Software Engineering metrics. In: 2nd International IEEE Conference on Computing for Sustainable Global Development (INDIACom), pp. 1654-1659. INSPEC Accession Number: 15109983 (2015).
3. Roger Pressman S.: Software Engineering a practitioners approach. Seventh edition, The Mc-Graw Hill Publications (2011).
4. Shobha Prabhu S., Hem Kapil, Shashirekha, H.L: Safety Critical Embedded Software: Significance and Approach to Reliability. In: 7th International Conference on Advances in Computing, Communication and Informatics, ICACCI-2018, pp. 449-455. DOI: 10.1109/ICACCI.2018.8554566, IEEE Xplore (2018).
5. Guidelines, Standard for: Software Considerations in Airborne Systems and Equipment Certification. RTCA DO-178B/C, Radio Technical Commission for Aeronautics (2011).
6. Lockhart, J., Purdy, C., Wilsey, P.: Formal methods for safety critical system specification. In : IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS), pp.: 201-204, DOI: 10.1109/MWSCAS.2014.6908387 (2014).
7. Milena Krasich: Modeling of SW reliability in early design with planning and measurement of its reliability growth. In: IEEE Conference on Reliability and Maintainability Symposium (RAMS), DOI: 10.1109/RAMS.2015.7105066, INSPEC Accession Number: 15112765 IEEE Xplore (2015).
8. Yannick Moy, Emmanuel Ledinot, Herve Delseny, Virginie Wiels, Benjamin Monate: Testing or Formal Verification: DO-178C Alternatives and Industrial Experience. IEEE Software Issue : May/June 2013,Vol.30, pp. 50-57, DOI : 10.1109/MS.2013.43 (2013).

9. Qiu Fang, Chenxi Zhang, Xin Ye, Jianqi Shi, Xiaoxian Zhang: A new approach for developing safety critical software in automotive Industry. In: 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 64-69. DOI: 10.1109/ICSESS.2014.6933515, INSPEC Accession Number: 14698700 (2014).
10. An article by Amir Ghahrai https://devqa.io/static-analysis-vs-dynamic-analysis-software-testing/ last accessed on 8th January 2017.
11. Yahya Tashtoush, Mohammed Al-Maolegi, Bassam Arkok: The Correlation among Software Complexity Metrics with Case Study. In: International Journal of Advanced Computer Research, pp. 414-419. (ISSN (print): 2249-7277 ISSN (online): 2277-7970)Volume-4 Number-2 Issue-15 (2014).
12. Certification Authorities Software Team (CAST): Position Paper CAST-19 (Rev 2) Clarification of Structural Coverage Analyses of Data Coupling and Control Coupling. Federal Aviation Administration, USA (2004).
13. Maia, T., Souza, M.: A Practical Methodology for DO-178C Data and Control Coupling Objective Compliance. In: International Conference on Software Engineering Research and Practice, SERP'18, pp. 236-240. CSREA Press, ISBN: 1-60132-489-8 (2018).