Journal of KOSSE. (2018. 12) Vol. 14, No. 2 pp. 49-57
DOI: https://doi.org/10.14248/JKOSSE.2018.14.2.049
www.kosse.or.kr

ISSN (print) : 1738-480X
ISSN (online) : 2288-3592

Software Reliability of Safety Critical FPGA-based System using System Engineering Approach

Satrio Pradana, Jae Cheon Jung*

Department of Nuclear Power Plant Engineering, KEPCO International Nuclear Graduate School

Abstract : The main objective of this paper is come up with methodology approach for FPGA-based system in verification and validation lifecycle regarding software reliability using system engineering approach. The steps of both reverse engineering and re-engineering are carried out to implement an FPGA-based of safety critical system in Nuclear Power Plant. The reverse engineering methodology is applied to elicit the requirements of the system as well as gain understanding of the current life cycle and V&V activities of FPGA based-system. The re-engineering method is carried out to get a new methodology approach of software reliability, particularly Software Reliability Growth Model. For measure the software reliability of a given FPGA-based system, the following steps are executed as; requirements definition and measurement, evaluation of candidate reliability model, and the validation of the selected system. As conclusion, a new methodology approach for software reliability measurement using software reliability growth model is developed.

Key Words : Software Reliability, Software Reliability Growth Model, Reverse Engineering, Re-engineering, V&V Activities, FPGA-based system, Software Reliability Measurement

Received: November 9, 2018 / **Revised**: December 21, 2018 / **Accepted**: January 7, 2019

* Corresponding author : Jae Cheon Jung, jcjung@kings.ac.kr

This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Software reliability is defined as the probability of failure-free software operation in a defined environment for a specified period of time. A failure is the departure of software behaviour from user requirements. [1] This dynamic phenomenon has to be distinguished from the static fault (or bug) in the software code, which causes the failure occurrence as soon as it is activated during program execution.

Since software does not deprecate like hardware, the reliability of software stays constant over time if no changes are made to the code or to the environmental conditions including the user behaviour. However, if each time after a failure has been experienced the underlying fault is detected and perfectly fixed, then the reliability of software will increase with time. Typically this is the situation during the testing phase.

It has become important issue for instrumentation and control systems in nuclear power plant. Particularly when using safety critical software, various methods like formal verification and validation play critical roles demonstrating compliance with several regulatory requirements.

In this paper, we proposed a methodology for assessing software reliability in safety critical software with FPGA based system. Since FPGA is a strong candidate to replace software system in nuclear power plant.

2. Software Reliability

2.1 Software Reliability Growth Model

Software reliability growth model (SRGM) is one of the methods that are used to quantify software failure rates and demand failure probabilities of digital system. SRGMs are Time-based methods that use test data to estimate software failure rates that, in turn, are employed to determine whether a particular software can be released, by demonstrating that its failure rate meets the desired level.

A large number of software reliability growth models (SRGMs) have been developed over the years[2-6], and each model has its strengths and limitations[7-9]. None is generally superior to the others, because all are based on assumed empirical formulas that are not applicable to all situations. There are different assumptions in different SRGMs on how the failure rate decreases with time; that is, the models specify different empirical formulas, and use test data to estimate their parameters. The empirical formulas and test data are employed to decide if the failure rate objective has been reached.

IEEE Standard 1633-2008 [10] grouped SRGMs into three high-level categories, exponential Non-Homogeneous Poisson Process (NHPP); non-exponential NHPP; and Bayesian; and briefly described many of the models.

2.2 Verification & Validation in FPGA

Verification and validation (V&V) processes are used to determine whether the development products of a given activity conform to the requirements of that activity and whether the product satisfies its intended use and user needs. According to IEEE 1012, [11] V&V life cycle process requirements are specified for different integrity levels. The scope of V&V processes encompasses systems, software, and hardware, and it includes their interfaces. V&V processes include the analysis, evaluation,

review, inspection, assessment, and testing of products.

As a software, FPGA need to be verified and validated. First stage is requirements definition phase (requirement phase), verification validation activities in accordance with the V&V Plan shall be performed. In addition, the independent review shall perform V&V activities. This phase states the minimum requirements for the V&V Plan, which shall specify the details of the V&V activities.

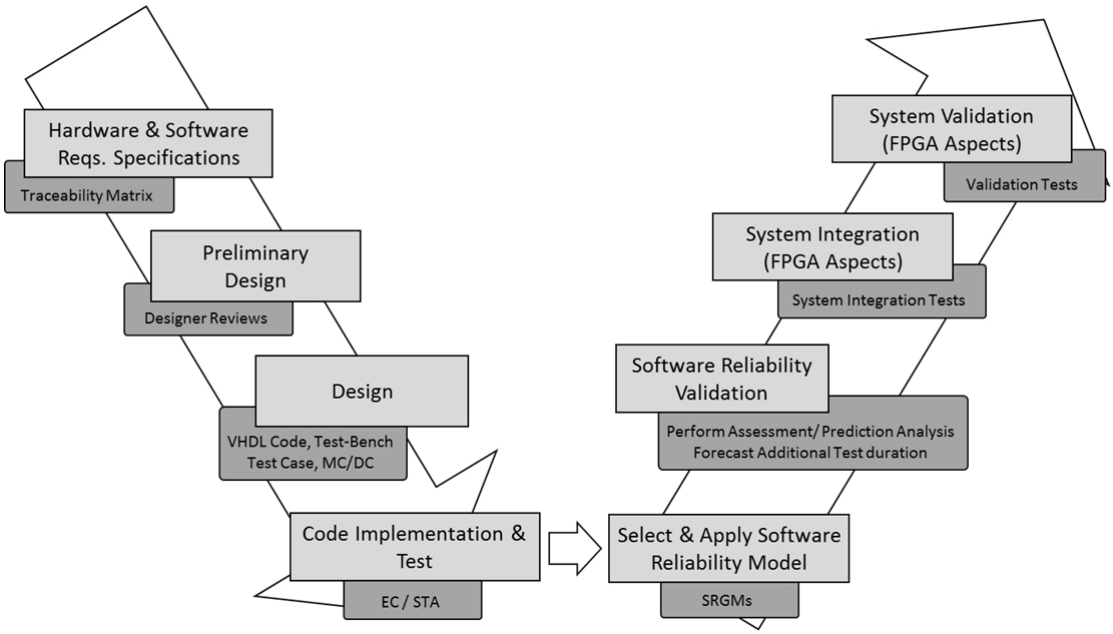
In the design phase, the FPGA Design Specifications including the Software Design Description (SDD) for each FPGA shall be produced. Because FPGAs are independent from each other; most of the activities in this phase and the implementation phase can be performed independently. The development activities in the implementation and integration phase are divided into the following steps such as: VHDL Source Coding, FPGA Implementation, and FPGA Validation. In between, shall be included process of measuring software reliability.

3. Methodology

As mentioned above, FPGA need a specific verification and validation activities. We propose a methodology for verification and validation as shown in Figure 1. An approach that we emphasize are after code implementation & testing. In order to perform the reliability analysis, Software Reliability Growth Model are applied.

3.1 Requirements definition (hardware, software, subsystems)

The requirements of definition are functional, performance and safety requirements; diagnostics; human factors; interfaces with external components; and communication requirements. When implementing the above requirements, V&V



[Figure 1] Verification and validation V model for software critical safety FPGA-based system

can ensure that: requirements are consistent with those specified in higher level documentation; every possible branch of the logic that implements the above requirements results in the desired outcome (White Box testing); and inputs result in the desired outputs independently of implementation details (Black Box testing).

Specifically for software requirement, this approach are emphasizes more in software part than hardware. It will be explained in the next section.

3.1.1 Requirements Specifications

As mention in IAEA Nuclear Energy Series No.NP-T-3.17 Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants. That gives some guidelines to define the requirements for FPGA as follow[12]:

1. Functional and timing requirements for FPGA circuit;
2. Electrical and logical interfaces;
3. Environmental conditions and corresponding qualification requirements;
4. Quality of service requirements,
5. O&M requirements,

3.1.2 Software Requirement

The purpose of Software Requirements is to establish the requirements of the software elements of the system. As a result, the requirements allocated to the software elements of the system and their interfaces are defined; it analyzed for correctness and testability; the impact on the operating environment are understood; consistency and traceability; prioritization for implementing the software requirements is defined; approved and updated as needed.

3.2 Design

V&V includes testing in a simulated environment. Requirements are translated into an architecture involving software and hardware components.

- Product Architecture Design: functionality allocated to hardware and software modules of the product.
- Software Architecture Design: software functionality allocated to the different modules and developed. The output of this activity is commented VHDL code and associated design description.

3.3 Implementation

Translation of the chosen architecture and functionality assigned to the different software and hardware modules into database structures, communication protocols and related machine executable representations.

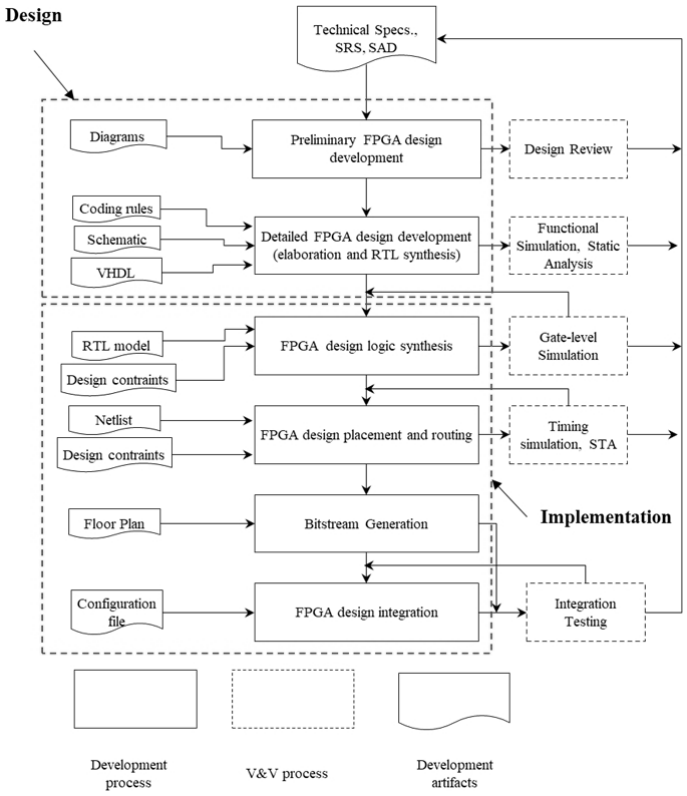
Figure 2 showing integration between the FPGA development process and the V&V activities.

3.4 Software Reliability Measurement

After finishing implementation and testing process, it is the time to measure the code reliability using the failure rate coming from applying the testing methods. This process is important as a step for developing a high-quality software.

3.4.1 Requirements on the testing data

The output from the implementation and testing phase shall be collected and gathered in a way to properly analyze a data sample for reliability



[Figure 2] FPGA design, development and V&V processes

purposes, the data must meet some prerequisites. There are two forms of data that are suitable for statistical analysis of reliability:

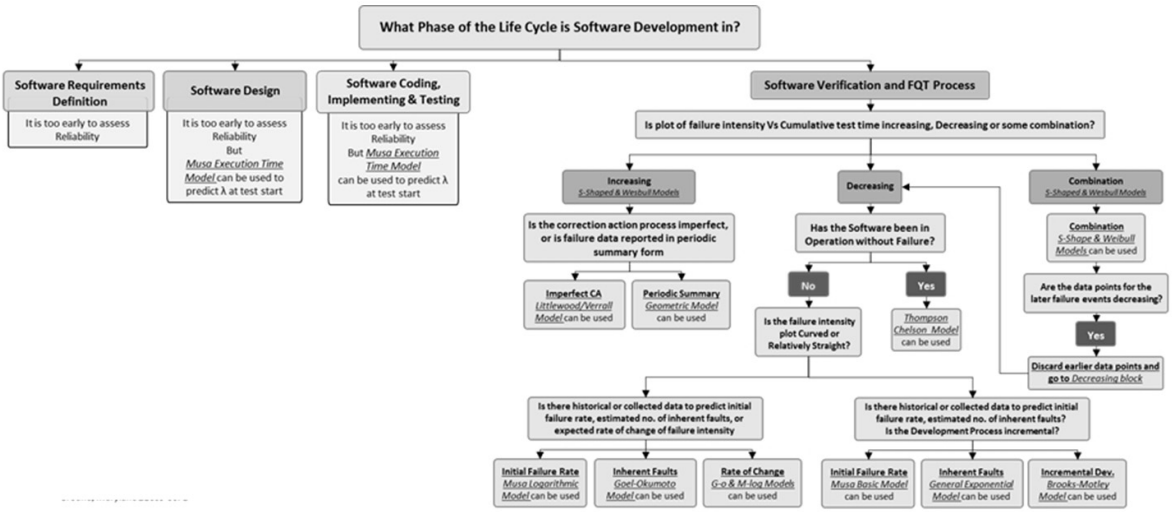
1. Event data, Time Between Failure or failure reports. This data should contain date and time information, information about the kind and severity, and downtime of each failure.
2. Time series data, or number of failures per time interval. There should be information about the time periods that these cover, and the time intervals should preferably be of uniform length.

3.4.2 Reliability Measurement

This process includes measures that are intended to discover and correct faults resulting from these errors, including reviews, audits, screening. Managing these errors involves describing the errors, classifying the severity and criticality of their effects, and modeling the effects of the remaining faults in the delivered product, and thereby working with designers to reduce their number of errors and their criticality.

Because there are many of potential models available, it is not easy to select which model may be most appropriate for a specific situation so an approach has been set as a guidance on model selection based on the following constraints:[13]

1. Failure profiles (failure intensity trend)
2. Maturity of software (what phase of its life cycle is the software in)
3. Characteristics of software development (how are failure modes detected/mitigated)



[Figure 3] Selection during life cycle phase [15]

4. Characteristics of software test
5. Existing metrics and data

SR models can both assess and predict reliability. The former deals with measuring past and current reliabilities. The latter provides forecasts of future reliability. The word "prediction" is not intended to be used in the common dictionary sense of foretelling future events, particularly failures, but instead as an estimate of the probabilities of future events. Both assessment and prediction need good data if they are to yield good forecasts. Good data implies accuracy (that data is accurately recorded at the time the events occurred) and pertinence (that data relates to an environment that is equivalent to the environment for which the forecast is to be valid).

In Figure 3, some of models are recommended to be used in different phase of software development lifecycle. It depends on which phase are we used the software reliability method. In this approach, testing the reliability of software can be assessed after code implementation and testing.

3.4.3 Evaluation and comparison criteria for reliability models

In other way called methods of model validation and comparison. To measure the software reliability one or more software reliability models should be selected and applied. Model selection criteria and guidance on the appropriateness of one model over another depending on certain situations is useful. From reviewing many journal papers, the applicable standards and making a survey for software reliability measurements toolkits an approach has been set and it include the following:

- Choosing Software Reliability Growth Model (SRGMs) as a quantitative software reliability methods (QSRMs).
- Select Toolkit to apply the SRGMs models

There are several ways in which a models goodness can be evaluated and these ways is listed as follow [14]:

1. Predictive validity: This is the capability of the model to predict future failure behaviour during either the test or the operational phases from present and past

failure behaviour in the respective phase.
2. Capability: The ability of the model to estimate with satisfactory accuracy quantities needed by software managers, engineers, and users in planning and managing software development projects or controlling change in operational software systems.
3. Quality of assumptions: If an assumption made by a model can be tested, the degree to which it is supported by actual data; if it is not possible to test an assumption, its plausibility from the viewpoint of logical consistency and software engineering experience. Also the clarity and explicitness of an assumption should be judged.
4. Applicability: Means the usefulness of the model across different software products (size, structure, and function), different development environments, different operational environments, and different life cycle phases.
5. Simplicity: The simplicity and inexpensiveness of collecting the data that is required to particularize the model.

3.4.4 Software Reliability tools

The Several software reliability tools are available to apply one or more of the software reliability model to a development effort and to determine the applicability of a particular model to a set of failure data. A major issue in modelling software reliability lies in the ease-of-use of currently available tools.

Following tasks are handled by the SRE tools:

1. Collecting failure and test time information
2. Calculating estimates of model parameters using information available.
3. Testing to fit a model against the collected information.
4. Selecting a model to make predictions of remaining faults, time to test, etc.
5. Applying the model

3.4.5 Software Reliability Validation

Once the data has been collected and the model has been selected and parameters are estimated, the analyst is ready to perform the appropriate analysis. This analysis should be to assess the current reliability of the software, predict the number of failures remaining in the code, or predict a test completion date.

As achievement of the reliability target approaches, the adherence of the actual data to the model predictions should be reviewed and the model corrected, if necessary.

Additional test duration should be predicted if the initial and objective failure intensities and the parameters of the model are known using software reliability tools.

3.5 System Integration (FPGA Aspects)

The process of system integration is the combining of verified FPGA hardware components into subsystems and finally into the complete system. This process consists of two kinds of activities:

a) System integration: assembling and interconnecting verified FPGA hardware in order to build the intermediate and final targets.
b) Integrated system verification: verifying that the components comply with their design specification, are capable of operating together, and comply with their interface requirements.

3.6 System Validation (FPGA Aspects)

The objective of this phase is to test the integrated system to demonstrate compliance with the functional, performance and interface specifications. Testing shall be performed to validate the system and its software, programming and configuration data to be in accordance with the system requirements. Validation shall comprise tests performed on the system in the final assembly configuration including the final version of the software and other programming data. System validation demonstrate that the system meets its system specification and its system requirements specification.

4. Conclusion and future work

Software reliability growth model for software critical safety which is based on FPGA hardware and VHDL software is developed in this work using the system engineering approach. The proposed methodology approach is conducted through the concept of verification and validation using V-model. The requirements, selecting criteria, and applying software reliability model are also proposed.

This paper shows the steps of both reverse engineering and re-engineering activities carried out to estimate reliability in software critical safety system on FPGA-based. The selecting process model and tools were reviewed. The reverse engineering methodology is applied to elicit the requirements of the system as well as gain understanding of the current life cycle and V&V activities of FPGA based-system. The re-engineering method is carried out to get a new methodology approach of software reliability, particularly Software Reliability Growth Model. For measure the software reliability of a given FPGA-based system, the following steps are executed as; requirements definition and measurement, evaluation of candidate reliability model, and the validation of the selected system. Emphasis in reverse and re-engineering aspect.

The future work will focus on the testing proposed methodology for software critical safety system by simulating software reliability tools and verifying the design measures according of performance requirements design such us, reliability analysis, safety integration and also coverage of code.

Acknowledgements

This work was supported by the 2018 Research fund of KEPCO International Nuclear Graduate School (KINGS), Republic of Korea.

References

1. C. A. Asad, M. I. Ullah, M. J. Rehman “An Approach for Software Reliability Model Selection,” in Proc. 28th Annual International Computer Software and Applications Conference (COMPSAC’04), Hong Kong, 2004.
2. Ajeet Kumar Pandey & N. K. Goyal, “A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics”, International Journal of Electronics Engineering, 1(2), 2009, pp. 239-245.
3. J. D. Musa and K. Okumoto, “A Logarithmic Poisson Execution Time Model for Software Reliability Measurement”, Bell Laboratories, Whippany, N. J. 07981.

4. Xiaolin Teng and Hoang Pham, "A New Methodology for Predicting Software Reliability in the Random Field Environments, IEEE Transactions On Reliability, Vol. 55, No. 3, September 2006.
5. BAI Cheng-Gang, Jiang Chang-Hai, & CAI Kai Yuan, "A Reliability Improvement Predictive Approach to Software Testing with Bayesian Method‖, Proceedings of the 29th Chinese Control Conference, July 29-31, 2010, Beijing, China.
6. Xiaolin Teng, & Hoang Pham, "A Software Reliability Growth Model for N-Version Programming Systems", IEEE Transactions on Reliability, Vol. 51, No. 3, September 2002.
7. Chin-Yu Huang, Michael R. Lyu & Sy Yen Kuo, "A Unified Scheme of Some Non-homogenous Poisson Process Models for Software Reliability Estimation," IEEE Transactions on Software Engineering, Vol. 29, No. 3, Page 261-270, March 2003.
8. Roger C. Cheung, "A User-Oriented Software Reliability Model‖, IEEE Transactions on Software Engineering, Vol. Se-6, No. 2, Page 118-126, March 1980.
9. Hoang Pham, "An Imperfect-debugging Fault-detection Dependent-parameter Software, International Journal of Automation and Computing, 04(4), October 2007, 325-328.
10. IEEE, "IEEE Recommended Practice on Software Reliability," IEEE Std 1633, 2008.
11. IEEE, "IEEE Standard for System and Software Verification and Validation" IEEE Std 1012, 2016.
12. IAEA, "Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants," International Atomic Energy Agency, Vienna, IAEA Nuclear Energy Series No. NP-T-3.17, 2016.
13. CSIAC Handbook of Software Reliability and Security Testing, 2014.
14. A. Iannino, J. D. Musa, & K. Okumoto, "Criteria for software reliability model comparisons," ACM SIGSOFT Software Engineering Notes, Vol. 8, No. 3 (July 1983), pp. 12-16.
15. US ARMY, Technical Report, NO. TR-2011-24, Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland 21005-5071, August 2011.