선진국방연구
Journal of Advances in Military Studies
2018, Vol. 1, No. 2, 1-12.

Weapon System Software Quality Improvement using Software Reliability Prediction Model

Ryu In Soo*
(MOASOFT)

≪Abstract≫

Most software flaws that occur during the operational phase are often not identified during the development phase. Therefore, to ensure reliability of software during the development phase is very important. This study proposes a predictive model to improve the reliability of the weapon system software during the development phase. Using the RADC model proposed in this study, we calculate the predicted reliability of the software development phase for the software project developed by M company. Experiments comparing before and after model application indicated a significant decrease in the predictive defect density (PDD) and the predictive defect count (PDC). The results of the experiment showed that we could reduce the burden on the operational phase by minimizing the failure in the software development phase.

Keywords : Software Reliability, Software Quality, Prediction Model, RADC Model

* isryu@moasoftware.co.kr

Ⅰ. Introduction

The aspect of the future ware will change to NCE (Network Centric Warfare), in order to implement the concept of joint battlefield operation, weapon systems such as surveillance, reconnaissance, command control and communication will be connected to a network. With the development of the fourth industrial revolution, IOT, Big data, Unmanned, and Automation technologies will make accelerate the development of the weapon system for NCE operations. The technologies associated with the fourth industrial revolution have been applying to weapons systems and it will be an essential of the future defense industry.

The importance of software system reliability is emphasized by increasing the proportion of software in the weapon system. The quality of the software requires an effort to ensure a quality of the process and the product perspective. For the purpose, software developers and organizations are trying to improve software quality through a variety of improvement with the capability maturity models. However, most of the software failures that occur during the operational phase is caused by a lack of identification in the development phase. Thus, systematic software quality management is required from the development phase to minimize the costs of defects correction and the burden in the operational phase. This study proposes the predictive model to improve the software reliability during the development phase and the applicability of the model proposed is evaluated using the comparison test before and after the model application through the case study.

Ⅱ. Literature Review

2.1 Review on the software quality assessments

John D. Musa emphasizes in his book 'Software reliability Engineering more reliable software faster and cheaper(2nd Edition).' the use of software reliability for software quality measurement as follows. "Software reliability engineering is integrally connected and is, in fact, a keystone to total quality management. It is a keystone in the sense that you cannot manage quality without a user-oriented metric of system reliability, and you cannot determine the reliability of software-based systems without a software reliability measure (Musa, 2004)."

The main content mentioned about software quality assurance in ROK DAPA's 'WSS Development and Management manual' is follow.

The quality assurance of WSS is an activity to ensure that the process applied to the software deliverable project complies with the quality requirements specified in the contract and complies with the pre-promised plan. WSS quality assurance is activities that software output and development process adhere to the quality requirements stated in the contract and activities to ensure compliance with the plan. This is divided into output assurance, process assurance and is performed through a review meeting, verification, validation, audit, etc. The software quality assurance methodology is conducted through software review meeting to ensure that the quality goals of the product and process are met, and the project management review meetings and technical review meetings (DAPA, 2017).

The WSS development and management manual focuses on the standards compliance and the quality of the output document during software development. In other words, qualitative quality control is mainly active. Quantitative quality control activities are needed from the beginning of development.

The software engineering body of knowledge describes the general knowledge of the field of software engineering. Each area of knowledge includes a list of basic concepts. There are 15 areas of knowledge, including software requirements, design, implementation, testing, engineering processes, engineering models and methodologies, and software quality is described in the tenth part. The Software Quality Measurement (SQM) is used to support decision making. It improves software quality in a variety of ways and is a useful method in the decision-making process. Software quality is an important factor in how well the quality goals are achieved beyond the behavior of the software (Abran, Moore, Bourque, Dupuis, & Tripp, 2004).

The SQM report provides valuable information in the entire software lifecycle process as well as the development process. Quality measurement uses the following techniques.

Statistically based techniques and statistical tests: Statistical-based techniques and statistical tests enable identification of more problematic parts in a software product (Abran et al., 2004).

Trend analysis: Charts and graphs of trend analysis help decision makers to make decisions about resource allocation.

Reliability models: Prediction technology using the reliability prediction model is useful to estimate the effort and time required for the test or to predict the failure or change in the software.

2.2 Review on the software reliability prediction

2.2.1 Software Reliability Evaluation model classification

The models for Software Reliability growth management are undergoing a lot of research, and they are mathematical models for measuring the reliability of software in specific operating environments.

Before software testing, predicting reliability by considering the development environment of the software to be developed is a reliability prediction model.

Initial reliability predictive models include Rome Lab, COQUALMO, which analyze based on characteristics of products, process-based models such as CMMI, and software architecture-based models that take into account organizational software development capabilities.

2.2.2 Reliability prediction model

The prediction model predicts the reliability from the requirements analysis phase to the software implementation phase and aims at the reliability growth of the software developed through the prediction. The results obtained by the reliability prediction model are the predicted defect density and the predicted defect count. These values represent the degree of integrity of the software.

The Rome Laboratory Prediction Model based on the RL-TR-92-52 standard is an RADC model (Friedman, 1992). This model is designed to predict software reliability of the US Air-Force Computer System, and It includes quantitative measures for developing reliable software during the software development process. Also, it is defined that it is possible to predict software reliability for each development phase. Each phase of development is based on the Defense Department's 'MIL-STD-2167A, Defense System Software Development'.

The Rome Lab Model can predict the initial defect density and current reliability, based on data collected from the software development environment, experience, and activities of the development process.

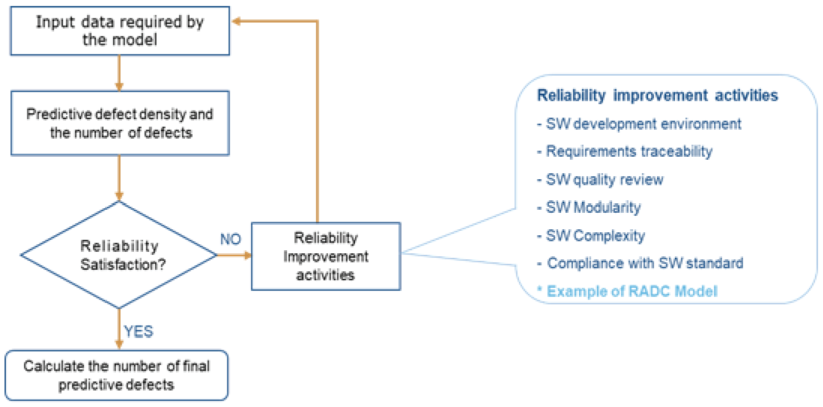
Therefore, the Rome Lab Model is classified as a model based on the product characteristics among the software reliability model classifications. The reliability calculation of this model shows the defect density per LOC(Lines of Code). That is, to predict the initial number of defects, it is possible to calculate the defect density.

In the previous research, only the result of calculating the software reliability was presented, but this paper additionally suggested the software reliability improvement activity.

Ⅲ. Quality Improvement Method on the Software Development Phase

3.1 Software reliability prediction process

The model used in this paper for predicting the reliability of the WSS are the RADC (RL-TR-92-15) model (Friedman, 1992). The process of predicting and improving software reliability activities are shown in figure 1.


Input data required by the model
Predictive defect density and the number of defects
Reliability Satisfaction?
NO
YES
Reliability Improvement activities
Calculate the number of final predictive defects
Reliability improvement activities
- SW development environment
- Requirements traceability
- SW quality review
- SW Modularity
- SW Complexity
- Compliance with SW standard
* Example of RADC Model

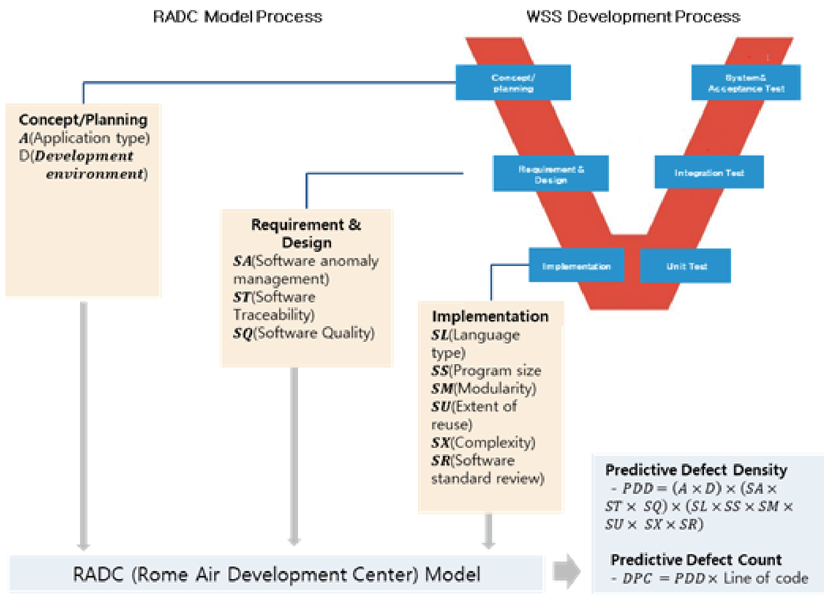
<Figure 1> Software reliability prediction and reliability improvement process

3.2 Reliability prediction using RADC model

This model calculates software defect density for each development phase and predicts the number of defects based on reliable quantitative measures in the software development process (IEEE, 2017; Ryu & Jeong, 2018). The software defect density and the number of defects can be predicted from the early phase of software development as shown in figure 2.

Concept/Planning Phase defect density calculation is A*D. A factor reflects the characteristics of the Application. For example, the application type is airborne, the A Factor value is 0.0128, and the tactical is 0.0078. Thus, it is proposed to quantify the average defect density according to the Application type (Friedman, 1992). D factor is calculated based on a checklist of the characteristics of the current system development Organization and the development environment of the system. The calculation elements are organizational considerations 8, Methods used 9, Documentation 12, Tools Used 9, Test techniques Planned have 6 elements to predict the defect density and number of defects.

SW requirements and design phase reflect the characteristics of the software requirements analysis and design process to predict defect density and defect count.



<Figure 2> RADC Model Process

The defect density calculation of this step is shown as below.

*Defect Density = A*D*S1 (S1: SA (Anomaly Management) * ST(Traceability)* SQ (Quality Review))*

The SA(Anomaly Management) checks the requirements specification and design documentation to determine whether there is a countermeasure for error and exception handling.

ST(Traceability) identifies the mapping relationship from the system requirements specification in the system analysis phase to the detailed design specification.

The SQ(Quality Review) is based on a checklist of quality review activities and faults identified in the review activities found in each review phase.

The implementation phase calculates defect density by reflecting the software characteristics of the software implementation process. The defect density calculation of this step is shown below[5].

*Defect Density = A*D*S1*S2 (S2: SL (Language Type) * SS(Program Size) * SM(Modularity) * SU(Extent of Reuse) * SX(Complexity) * SR(Standard Review))*

SL(Language Type) is a classification according to the type of software implementation. SS(Program Size) is the dimensions of the programs. SM(modularity) is a measure for determining the readability and understanding of the module based on the size of the implementation module. SU(Extent of Reuse) is the ratio of the reusable code. SX(complexity) uses the Cyclomatic complexity as a measure to determine the structural complexity of the implemented module. Standard Review. The SR(Standard Review) is a measure of compliance with standards during implementation.

3.3 Case study on the RADC Model

Using the RADC model, I calculate the predicted reliability of the software development stage for the software project developed by M company. I identified the need for software reliability improvement at the development phase and applied complementary activities.

3.3.1 Concept and planning phase

According to the above table 1, the first prediction defect density measurement results of this step are as follows.

*PDD (Predictive Defect Density) = A*D*

*PDD: 0.0246 = 0.0123*2*

After the reliability improvement activities defect density is as follows.

*PDD: 0.00615=0.0123*0.5*

<Table 1> Concept phase defects density

	Input Data	Factor Value		Reliability Improvement Activities
		Initial Measurement	Measurement after improvement	
System type	- Application Type	0.0123	0.0123	
Development environment	- Organizational Considerations - Methods Used - Documentation - Tools Used - Test Techniques Planned	2	0.5	2 Yes items Increased -Using the Configuration Management Tool -Add a quality assurance plan

3.3.2 Requirement and Design Phase

<Table 2> Requirement and Design Phase Concept phase defects density

	Input Data	Factor Value		Reliability Improvement Activities
		Initial Measurement	Measurement after improvement	
System type	- Application Type	0.0123	0.0123	
Development environment	- Organizational Considerations - Methods Used - Documentation - Tools Used - Test Techniques Planned	0.5	0.5	
Anomaly management	- Action for Error Condition - Identification and recovery of calculation/ operating failures	1.0	0.9	-Add final validation checks for output data, etc. =>4 Yes items Increased yes
Traceability	- Traceability of requirements and designs	1.1	1	Traceability 80% => 95% Increased
Quality Review	- Quality review activities - Found Quality Defects	1.1	1	-Define inputs, processing, and output. etc. =>5 Yes items Increased yes

According to the above table 2, the first prediction defect density measurement results of this step are as follows.

*PDD (Predictive Defect Density) = A*D*(SA*ST*SQ)*

*PDD: 0.0223245 = 0.0123*0.5*(1.0*1.1*1.1)*

After the reliability improvement Activities defect density is as follows.

*PDD: 0.005535 = 0.0123*0.5*(0.9*1*1)*

Because the number of source code lines in this step is unknown, the number of predictive defects can be calculated in the implementation phase.

3.3.3 Implementation Phase

<Table 3> Implementation phase defects density

Input Data		Factor Value		Reliability Improvement Activities
		Initial Measurement	After improvement	
System type	Application Type	0.0123	0.0123	
Development environment	- Organizational Considerations - Methods Used - Documentation - Tools Used - Test Techniques Planned	0.5	0.5	
Anomaly management	- Action for Error Condition - Identification and recovery of calculation /operating failures	0.9	0.9	
Traceability	- Traceability of requirements and designs	1	1	
Quality Review	- Quality review activities - Found Quality Defects	1	1	
language Type	- Assembly Language code line - Higher Order Language code line	1	1	No improvement needs
Program size	- Source Code Line	2	2	No improvement needs
Modularity	- CSCI Unit Modularity	1.21053	0.9120623	SM<200LOC module increase (8)
Reusability	- Extent of reuse	0.1	0.1	No improvement needs
Complexity	- Cyclomatic Complexity	0.9155642	0.67	SX>=20 module decrease (10)
Standard review	- Software standard review	1.5	0.75	Modules With problems decrease (10)

According to the above table 3, the first prediction defect density measurement results of this step are as follows.

*PDD (Predictive Defect Density) = A*D*SA*ST*SQ*(SL*SS*SM*SU*SX*SR)*

PDD: 0.00184=0.0123 2* 1* 1.1* 1.1*(1* 2* 1.21053* 0.1* 0.9155642* 1.5)*

*Predictive Defect Count (PDC) = Defect Density * Source Lines of Code*

*PDC: 64.78 = 0.009897057*35,200*

After the reliability improvement Activities defect density is as follows.

PDD: 0.000507= 0.0123 0.5* 0.9* 1* 1* (1* 2* 0.9120623* 0.1* 0.67* 0.75)*

*PDC: 17.86 = 0.000507*35,200*

The RADC model measures defect density by development phase. Defect density was reduced after identifying and improving reliability factors based on the measurement of defect density. The more these activities are strengthened in the development phase, the fewer defects that move into the testing phase.

The systematic reliability improvement activities in the requirements analysis phase, design phase, and implementation phase can result in a reduction in the test period and personal input.

Ⅳ. Analysis of Effectiveness

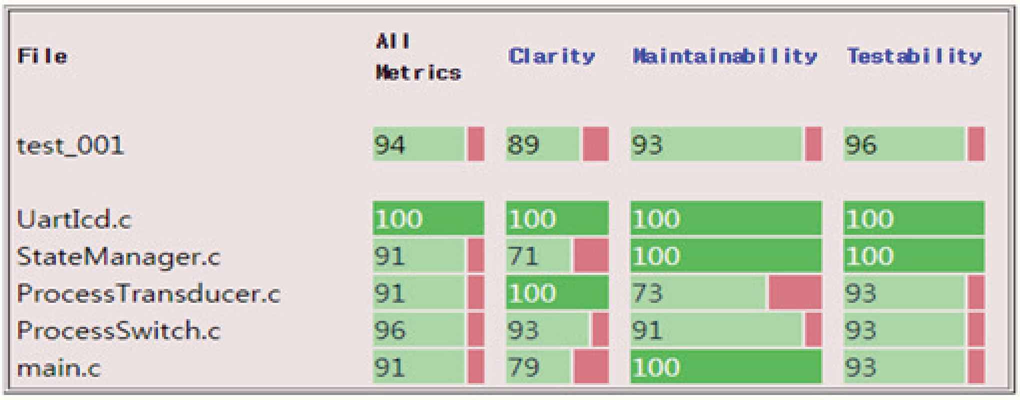
By using the prediction model to calculate defect density and defect counts, improve reliability in the development phase, I reduced the number of defects from 216.48 to 17.86, as shown in table 4. The software test time is expected to reduce due to the reduction in the number of defects. Therefore, Reliability improvement activities should be performed.

<Table 4> RADC model software quality improvement activity result

	Concept Phase	Requirement and Design Phase	Implementation Phase
Defect Density	0.00615	0.005535	0.000507
Defect Counts	216.48	195.01	17.86
SLOC	(35,200)	(35,200)	35,200

Using the LDRA tool, measurements of quality metrics of the project that applied prediction model(Figure 4).

By comparing the quality metrics of the project, I can see that the quality metrics of the project with the quality metrics are very well. The software quality of projects that manage software quality using prediction models that calculate defect density and number of defects is excellent.



File	All Metrics	Clarity	Maintainability	Testability
test_001	94	89	93	96
UartIcd.c	100	100	100	100
StateManager.c	91	71	100	100
ProcessTransducer.c	91	100	73	93
ProcessSwitch.c	96	93	91	93
main.c	91	79	100	93

<Figure 4> Results of Prediction model applied Projects

Ⅴ. Conclusion

I research how to predict software reliability using the RADC model at the software reliability prediction step and suggest reliability improvement activities at software development phase. I measured the software quality on the project, the quality of the project that applied the method presented in this paper was measured well. It can be improved the quality of software by minimizing potential defects at the phase of software development. I think that this research will contribute to software quality improvement of a weapon systems.

References

Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. L. (2004). Guide to the software engineering body of knowledge: 2004 version SWEBOK. IEEE Computer Society.

Defense Acquisition Program Administration (DAPA) (2017). Weapon System Software Development and Management Manual. DAPA.

Friedman, M. (1992). Methodology for Software Reliability Prediction and Assessment. Report RL-TR-92-52, Rome Laboratory 1992 (2 volumes).

IEEE (2017). Recommended Practice on Software Reliability. IEEE Std 1633$^{\text{TM-}}$2016 (Revision of IEEE Std 1633-2008).

Musa, J. D. (2004). *Software reliability engineering: more reliable software, faster and cheaper.* Tata McGraw-Hill Education.

Ryu, I. S., & Jeong, S. J. (2018, May). *A Study on the Weapon System Software Reliability Prediction and Estimation Process at the Software Development Phase.* In International Conference on Computational Science and Its Applications (pp. 205-217). Springer, Cham.

원 고 접 수 일　2018년 11월 28일
원 고 수 정 일　2018년 12월 26일
게 재 확 정 일　2018년 12월 28일