Study on Software Reliability Design Criteria Based on Defect Patterns

Fuping ZENG, Aizhen CHEN, Xin TAO
Department of System Engineering
Beihang University
Beijing, 100191, China

Abstract—**Software reliability design criteria, one of the primary means to improve software reliability, could help to summarize experience and lessons learned. Nowadays, more and more organizations are putting emphasis on the collection and utilization of software defects. Software defects are the root causes for software failures. Therefore, software reliability design criteria based on the analysis of the defects could avoid the occurrence of similar defects and improve software quality. This paper investigates the idea of studying software reliability design criteria on the basis of defect patterns. Through the analysis of defect data, we advance a defect classification suitable for each software development phase and the definition of software defect patterns. The software defect patterns in requirement analysis, design and coding phase are listed. One method to solve the problem of how to convert from defect patterns to reliability design criteria is proposed. Subsequently the reliability design criteria in requirement analysis, design and coding phase are expounded by using the method. The reliability design criteria are tested to be practical and valid in application examples.**

Keywords- software reliability design; software reliability design criteria; software defects; software defect patterns; defect prevention

I. INTRODUCTION

At present, the main functions for the majority of equipments and systems are realized by software. Software is applied more and more widely in the industry. Software failures could not only result in serious economic losses, but also cause some disastrous consequences. Therefore, software reliability engineering plays an important role in the successful software production.

There are two main study directions in software reliability engineering. One is software process qualitative analysis, including reliability design and analysis, reliability test and evaluation, the other is software product quantitative analysis, including reliability model and reliability allocation. In the past, study on software reliability engineering focused on quantitative evaluation and analysis, and the study on reliability design and analysis is relatively weak. In recent years, software reliability engineering activities gradually extend into the whole process of software lifecycle[1]. As software reliability engineering activities in the early stage of software lifecycle, reliability design and analysis are paid more and more attentions.

With advantage of low cost and easy to implement, software reliability design criteria are the main contents and means for carrying out reliability design, thus, it has important practical value in engineering. Nowadays the relatively authoritative reliability design criteria standard has been issued for ten years. In this period, more and more valuable experiences and lessons in software engineering practice are accumulated, and reliability design criteria standard has not been updated yet. It is necessary to study reliability design criteria based on these new experiences and lessons.

Software defect is the root cause of software failure and affects software reliability directly[2], Software reliability design is essentially the process of struggling with defects constantly. At present, software defects play a very important role in software development, and more and more software organizations become to pay much attention to defect collection and management. Although lots of defect data have been collected, the utilization of defect data is not enough. This paper studies software reliability design criteria based on defect and defect prevention, and gives reliability design criteria to avoid the occurrence of similar defects. The goal is to improve software reliability.

Firstly, this paper briefly introduces the research on software defect patterns, and addresses defect patterns in requirement analysis, design and coding phase. Then, the method is proposed on how to convert from defect patterns to reliability design criteria, the reliability design criteria in requirement, design and coding phase are expounded. Finally, the application of reliability design criteria are presented and discussed.

II. SOFTWARE DEFECT PATTERNS

A. What is Software Defect Patterns

Patterns are carried out to aim at repeated problems in complex system. In resolving problems, most experienced experts always take into account the previously resolved similar problems, and reuse their essential solutions to solve the problem. The essential solutions that are continuously quoted are usually said patterns. In essence, a pattern is an abstract concept, it is the law found and abstracted from the repeated occurrence events, and it is the experience summed up to solve problems[5]. There should be some patterns in repeated occurrence.

978-1-4244-4905-7/09/$25.00©2009 IEEE

Software defects are a kind of software inherent attributes and they always occurs repeatedly. In other words, software defects have some certain patterns. A software defect pattern is defined as the following in this paper:

A software defect pattern is the abstract description of a particular kind of repeated or similar software defects.

The definition of software defect patterns is applicable to defects introduced from any development phase. We collect the corresponding defect data found in process of software development and test. In addition, the defect data from domain experts and domestic and foreign references are gathered as a supplement. By using the related data analysis technology[6][7], software defect patterns are acquired.

B. Software Requirement Defect Patterns

The research foundation of requirement defect patterns is the defect data which are introduced in software requirement phase.

First of all, from the perspective of software requirement contents, the requirement defects are classified into five categories: functional requirement defects, non-functional requirement defects, interface requirement defects, data requirement defects and others.

Then, by further analysis requirement defect data, software requirement defects are classified into five general requirement defect patterns: requirement missing, requirement inconsistency, requirement redundancy, requirement ambiguity, requirement error. These requirement defect patterns are applicable for each type of requirement defect classification.

C. Software Design Defect Patterns

To obtain software design defect patterns, the foundation of research is the defect data which are injected in design phase.

Firstly, from the perspective of software design contents, the software design defects are classified into five categories: program structure design defect, process design defect, data and database design defect, interface design defect and others.

Then, by further analysis design defect data, software design defects are classified into five general design defect patterns: design missing, design inconsistent with requirement, design redundancy, design complexity over-high, design error. These design defect patterns are applicable for each type of design defect classification.

D. Software Coding Defect Patterns

Software code is the realization of software design. Adopted the prescriptive programming language, all software functions are realized.

Firstly, the software coding defects are classified into seven categories: data defect, calculation defect, logic defect, function defect, exception handling defect, file I/O defect and others. Then we have analyzed the defect data in C language coding, the seven software code defect classifications are further divided into thirty-two coding defect patterns.

1) Data Defect: use of uninitialized variable, variable type mismatch, redefinition with different variable type, null pointer reference, wild pointer, incorrect macro definition, modification of ‘constant pointer’.

2) Calculation Defect: imprecision caused by mixed operation, incorrect bit operation, incorrect comparison operator, array boundary violation, buffer overflow, algorithmic error.

3) Logic Defect: inconsistence with design, no executive statement in branch, constant conditions, incomplete judging condition, overlapping condition, uncontrolled cycle conditions.

4) Function Defect: function redefinition, function defined with empty body, function declaration inconsistent with function definition, real parameter inconsistent with formal parameter, returned type inconsistent with function type.

5) Exception Handling Defect: operate on non-exist files, file type error, incorrect exception handle, and memory application failure.

6) File I/O Defect: no-closure after file open operation, file I/O defect pattern.

7) Others: macro definition defect pattern.

III. Software Reliability Design Criteria Based on Defect Patterns

A. Research Method

How to convert software defect patterns into reliability design criteria? Through studying on the cause analysis and preventive measure of software defect [8][9][10], it is shown that software reliability design criteria are defect preventive measures to some extent. Therefore, this paper adopts the following method to solve the problem.

Firstly, software defect pattern warehouse is established, and the corresponding preventive measures for defect patterns are analyzed. Then, reliability design criteria are obtained by making Cartesian product of defect classification and defect preventive measures in software defect pattern warehouse.

Step 1: Set up software defect pattern warehouse.

The defect pattern warehouse is defined as quaternion group in this paper, which is the set of the defect pattern attributes, including the name of defect pattern, the cause of defect pattern, the consequence of defect pattern, and the preventive measure of defect pattern. It can be defined as $\psi=(N,I,O,P)$. In which:

- N is the name of defect pattern, which is able to identify defect pattern uniquely, and it can‘t be empty value;
- I is the cause set of defect pattern, and it can’t be empty value;
- O is the consequence set of defect pattern, and it can’t be empty value;
- P is the preventive measure set of defect pattern, and it can’t be empty value.

For the above quaternion group, there are the following notes:

- The cause of defect pattern mainly refers to the common causes which lead to a class defects during software development process.
- The consequence of defect pattern refers to the consequences of the failures which are caused by defects without any fault-tolerances.
- Software defect preventive measures are basically the similar for the same type of defect patterns. Different types of software and domain are different in describing forms. In a certain sense, the defect preventive measures are the software reliability design criteria.

It is a long-term process to set up defect pattern warehouse, and need to continuously collect defect data and defect attributes to supplement the information of defect pattern warehouse. The scheme to set up defect pattern warehouse is shown in Figure 1.

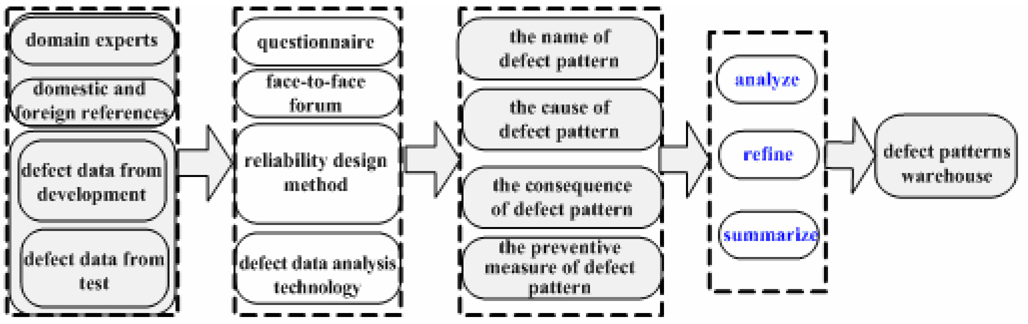
domain experts
domestic and foreign references
defect data from development
defect data from test
questionnaire
face-to-face forum
reliability design method
defect data analysis technology
the name of defect pattern
the cause of defect pattern
the consequence of defect pattern
the preventive measure of defect pattern
analyze
refine
summarize
defect patterns warehouse

Figure 1. Scheme of defect pattern warehouse establishment

Step 2: Carry out Cartesian product operator.

The defect patterns in the requirement and design phase are not specifically proposed for any defect classification, thus they are abstract, and the preventive measures in defect pattern warehouse are also abstract, however the reliability criteria are specific, thus the Cartesian product in collection algebra is introduced to set up the relationship between the reliability design and the preventive measures. The method to make the reliability design criteria in coding phase is different, the coding reliability design criteria are obtained directly from the preventive measures without doing Cartesian product.Cartesian product is defined as follows.

Cartesian product: given a group of sets of D_1, D_2, D_3, ..., D_n , The Cartesian product of n sets $D_1 \times D_2 \times D_3 ... \times D_n$ is the set of all possible orderly Tuple $<d_1, d_2, d_3, …, d_n>$, in which $d_1 \in D_1$, $d_2 \in D_2$, ..., $d_n \in D_n$.

By the Cartesian product definition above, the reliability design criteria in the requirement phase and design phase can be obtained. The reliability design criteria in requirement phase are the result that requirement defect classifications and preventive measures of requirement defect patterns are carried out Cartesian product. The reliability design criteria in design phase are the result that design defect classifications and preventive measures of design defect patterns are carried out Cartesian product.

B. Results

1) Software Requirement Reliability Design Criteria

According to the method mentioned above, taking the defect pattern of requirement missing as an example, this section describes in detail how to deduce the requirement design criteria based on requirement defect patterns.

Step 1: Establish requirement defect pattern warehouse.

Through analyzing the requirement defect pattern attributes, including the name, cause, consequence and prevention, we can establish requirement defect pattern warehouse. Table 1 gives some examples.

TABLE I. SOFTWARE REQUIREMENT DEFECT PATTERN WAREHOUSE EXAMPLES

Requirement Defect Pattern	Causes	Consequences	Preventive Measures
Requirement Missing	Analysts do not fully communicate with users; Omission of certain needs; Analysts have an inadequate consideration of "what should do and what should not do"; Carelessness leads to the omission of certain need.	Software function and performance cannot meet what users need, software has defects.	We should fully communicate with users completely, considering all functions and performances, including the mission scenes or information flow how to use the software to complete what users expect, and must be made clear functional inputs, outputs and data; Besides, must be made clear what software should do and not do, and the exception.

Step 2: Introducing Cartesian product.

There are five defect classifications and five defect patterns in requirement phase. Thus, there are five × five, that is, twenty-five, requirement reliability design criteria. Due to length limitation, we take the requirement missing defect pattern of function requirement as an example, it is showed in Table 2.

TABLE II. THE REQUIREMENT RELIABILITY DESIGN CRITERIA EXAMPLES

Defect Classification	Defect Pattern	Reliability Design Criteria in Requirement Phase
Functional Requirement Defect	Requirement Missing	Criteria 1: Must fully understand each function of software, including the mission scenes or information flow how to use the software to complete what users expect, and describe functional characteristics with text, graphics or mathematical method. Explanation: Must define the content of functional requirements to ensure the integrity, non-ambiguity and consistency of functional requirements, including must be made clear in each functional input and output and deal flow; Must be described all input information about functions; Must be described all output information about functions.
		Criteria 2: When the input has the scope, not only required to list the deal flow inside the scope of the input, but also required to list the deal flow outside the scope of the input.

According to the above method, forty-eight requirement reliability design criteria are proposed based on requirement defect patterns, including functional requirement, non-functional requirement, data requirement, interface requirement and others.

2) *Software Design Reliability Design Criteria*

The method how to deduce the reliability design criteria in design phase is similar with that in the requirement phase. It has two steps. First, design defect pattern warehouse is established. Second, reliability design criteria in design phase are obtained by the Cartesian product of defect classifications and preventive measures. The example of reliability design criteria in design phase is given in Table 3. Forty-three reliability design criteria in design phase are proposed by this method based on the design defect patterns, including program structure design, process design, data and database design, interface design and others.

TABLE III. THE EXAMPLES OF RELIABILITY DESIGN CRITERIA IN DESIGN PHASE

Defect Classification	Defect Pattern	Software Reliability Design Criteria in Design Phase
Program Structure Design Defect	Design Complexity is over high	Criteria 1: Module size should be moderate Explanation: The size of a module should not be too large, preferably written within a page, normally not more than 60 line statements.
		Criteria 2: Depth, width, fan-out and fan-in should be appropriate
		Criteria 3: Strive to reduce the complexity of the module interface
		Criteria 4: The Mccabe complexity must be restricted Explanation: According to experience, the possible coding errors have a great relationship with the high complexity, the high Mccabe complexity indicates the quality of code may be low and difficult to test and maintenance, thus it must be limited, and it is generally not more than 10

3) *Software Coding Reliability Design Criteria*

It has two steps that the method how to deduce the reliability design criteria in coding phase. First, coding defect pattern warehouse is established. Second, reliability design criteria in coding phase are obtained directly from preventive measures. Because the defect patterns in requirement and design phase are abstract, they are not specifically proposed for defect classification. Whereas, the defect patterns in coding phase are proposed specifically for coding defect classification. Thus the coding reliability design criteria are obtained directly from preventive measures without doing Cartesian product. We compiled thirty-three reliability design criteria in coding phase, which are partly shown in Table 4.

IV. APPLICATION EXAMPLE

In order to verify the practicality and efficiency of reliability design criteria based on defect patterns, the optical system control software is selected as the application example. According to the software characteristics, the reliability design criteria which are suitable for this software project are instituted. The selection rates of the software reliability design criteria in requirement analysis, design and coding phase are 71.1%, 83.8% and 100% respectively.

Software developers have a high evaluation on the reliability design criteria. They think the reliability design criteria are what they need urgently in the practical work of reliability design. According the developers, the reliability design criteria are characterized by rich contents, detailed information, clear structure, easy understanding and practicality. At present, the software reliability design criteria have effectively used to guide the software project to perform reliability design in the development phase.Authors and Affiliations

TABLE IV. THE CODING RELIABILITY DESIGN CRITERIA EXAMPLES

Defect Classification	Defect Pattern	Software Reliability Design Criteria
Logical Defect	Inconsistent with design	The judging condition, branch structure, boundary value and so on should be consistent with the design
	Without executive statement in branch	Avoid unnecessary branches; Avoid branch is empty; Avoid to use the structure of "then if" and empty "else" statement; Avoid to use the structure of "else goto" and "else return"
	Condition is always right	Avoid to use the structure that condition is always right
	Incomplete judging condition	Avoid to miss intervals in the judging branch
	Overlapping condition	Avoid to use the overlapping condition; Avoid to exist the two equivalent branches
	Looping condition is beyond the control	Avoid the looping condition is empty; Avoid to put constant as looping judging condition

V. CONCLUSION

This paper studies on the reliability design criteria based on the analysis of software defect data. The method on how to build the reliability design criteria based on defect patterns is proposed. By applying this method, the reliability design criteria have been established. Application example of the method indicates that the reliability design criteria based on defect patterns are more effective in avoiding the recurrence of similar defects and improving software reliability.

The existing reliability design criteria can be further improved and supplemented with the accumulation of the defect data by using the method mentioned above, and thus be used to guide software reliability design more effectively in future.

ACKONWLEDGEMENTS

This work is supported by General Armament Department Advanced Research Project.

REFERENCES

[1] X.Z. Hang, Software Reliability, Safety and Quality Assurance. Beijing: Electronics Industry Press,2002.

[2] Y.Z. Gong, "On software's defects," Journal of Academy of Armored For Engineering, 2003, Vol.17, No.1, pp.60-63.

[3] W.H. Zhang, "Process and method of software defect prevention," Computer Engineering, 2004, Vol.30, pp.23-25.

[4] L.T. Jiang, and G.Z. Xu, "Software fault and software reliability," Computer Simulation, 2004, Vol.21, No.2, pp.141-144.

[5] B. Goldfedder, Using Patterns for Enterprise Development. Beijing: Tsinghua University Press, 2003.

[6] H. Liu, and K.G. Hao, "Method of software defect data analysis and its implementation," Computer Science, 2008, Vol.38, No.5, pp.262-264.

[7] W.G. Han, H.J. Zhou, and L.F. Zhao, "Research on information of software defects," Computer Engineering and Design, 2008, Vol.29, No.13, pp.3381-3383.

[8] K. Adeel, S. Ahmad, and S. Akhtar, "Defect prevention techniques and its usage in requirements gathering - industry practices," IEEE Engineering Sciences and Technology, 2005, Vol.8, pp.1-5.

[9] M. Li, X.Y. He, and A. Sontakke, "Defect prevention: a general framework and its application," IEEE Quality Software, 2006, Vol.10, pp.281-286.

[10] E. Bean, "Defect prevention and detection in software for automated test equipment," IEEE Instrumentation & Measurement Magazine, 2008, Vol.8, pp.16-23.