Software Reliability Qualitative Evaluation Based on Modified Delphi Hierarchy Process

Peng Cao[1,a], Ziqiang Luo[2,b]*

[1]Dept. of Mathematics and Physics, Qiongtai Teachers College, Haikou, 571100, China

[2]College of Information Science and Technology, Hainan Normal University, Haikou, 571158, China

[a]lanyuan97@126.com, [b]306003057@qq.com

*Corresponding Author

Keywords: Software Reliability, Qualitative Evaluation, Delphi Hierarchy Process, Cloud Model.

Abstract. An approach for software reliability qualitative evaluation by using a modified Delphi hierarchy process based on cloud model was proposed. When the weights of the factors affecting software reliability are determined, we can obtain software reliability qualitative rules using standard weighted association rule mining algorithm. Finally software reliability qualitative evaluation can be achieved through uncertain reasoning based on cloud model.

Introduction

Traditional software reliability models are often founded on failure data from the software testing or the actual running phase, ignoring the influence of various uncertainty factors in the software development process. If there is very few data or no information related to software failure, we can evaluate software reliability indirectly by means of investigating some uncertain factors which determine the level of software reliability in the software development process. Abundance of uncertain information in software reliability engineering requires a formal symbol system to portray. Concepts and the overall characteristics of things are perceived through language, not by precise calculation of some values. In the objective world, there is a very prevalent relationship between the uncertainty inherent in natural language and fuzzy. Cloud model[1] can be used to describe language atoms, then just to reflect this relationship.

For comprehensive analysis of the relationship between several uncertain factors affecting software reliability and software reliability level, we will first determine the weighs of the factors, then we can obtain software reliability qualitative rules using standard weighted association rule mining algorithm[2]. Thus software reliability qualitative evaluation can be achieved through uncertainty reasoning based on cloud model.

Uncertain Factors Affecting Software Reliability

Software development is a very complex dynamic process including people, development tools and application background. Researchers[3,4] realized that the implied useful knowledge, which can be digged out from the various factors affecting the software reliability, software application features and software failure data in the software development process, may be used for qualitative and quantitative software reliability assessment process. Thus more credible and accurate software evaluation results can be obtained. Because software reliability is subject to common effect of numerous uncertainties factors, we need to analyze comprehensively such factors to assess. The degree of the factors impacting on the software reliability need to be determined by means of engineering experience and subjective judgment, inevitably bearing subjectivity and uncertainty. Cloud model theory is suitable for processing the information with uncertain attributes.

According to the survey of existing research and analysis, there are 32 uncertainty factors affecting software reliability during the entire software development process. These factors constitute a qualitative evaluation index system for software reliability. Here mainly introduce the six most important factors.

1) Program Complexity, abbreviated as PC: the more complex the software is, in the development and maintenance process the more resources are consumed, and the greater possibility of errors are introduced in the design. The program size, function modules themselves and the connections between modules are the main factors determining software complexity.

2) Programmer Skills, abbreviated as PS: programmer skills mainly depend on the programmer's level, number of developed software products, the scale of the developed software and the experience of developing software. In General, the programmers’ skills directly affect software reliability. Due to personal programming experience, work stress and personal qualities, programmers often make errors in the program modules.

3) Testing Efforts, abbreviated as TEF: the number of test cases, test workload and test time reflect the testing efforts.

4) Testing Coverage, abbreviated as TC: the proportion of source code covered by test cases.

5) Testing Environment, abbreviated as TE: in order to found more software bugs in the testing phase, testing environment should be as close as possible to the actual operating environment.

6) Frequency of Specification Change, abbreviated as FSC: because users have new requirements, or developers do not properly understand the user's needs, specification needs to add or correct during the period of software development. This will certainly affect the subsequent software development. The more the number of changes or the later change occurs, the greater the impact on reliability.

Because the factors affecting software reliability, which are taken from the software development process at all stages, are closely related with software reliability levels, can truly reflect the software reliability from different angles. But various factors have primary and secondary points, so you need to assigning weights to highlight the main factors.

Delphi Hierarchy Process Based on Cloud Model

Delphi hierarchy process based on cloud model[5,6], DHP for short, synthesizing a number of experts’ opinion on many factors affecting software reliability, cleaning up and selecting factors, and assigning weights, can implement visualized heuristic optimization n of the complex data objects.

By using of DHP method, we can select factors affecting software reliability and calculate initial weights with Delphi method, and then use the analytic hierarchy process, known as AHP, to get hierarchical factors and to determine the final weights.

DHP is briefly introduced as follows: first, the issues to be assessed and the necessary background material in the form of communication may be sent to experts, then the replies should be synthesized and concluded based on cloud model. Once again back to the experts for further advice, so repeatedly 2-4 times, until we get satisfactory results. Second, calculate the relative weights of each level, and calculate the combined weights of the elements. Finally, the total weight can be calculated.

Establishment and Transformation of Related Databases

For some factors affecting software reliability, we can not get the real data, but with qualitative description language, such as "programmer skills are ordinary", "testing method is good", and so on. These language values can be represented appropriately by the cloud model in the domain of [0,1], which can be converted to a numeric representation through a forward cloud generator. Thus, by some typical case studies, we can get numeric databases about factors and software reliability, as shown in Table 1.

Table 1. Databases about factors and software reliability

PC	PS	TEF	TC	TE	FSC	⋯	Software Reliability
0.01	0.96	0.98	0.99	0.99	0.01	⋯	0.99
0.2	0.89	0.97	0.88	0.87	0.03	⋯	0.77
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

Table 2. Soft partition for the attribute values table

PC	PS	TEF	TC	TE	FSC	⋯	Software Reliability	Count/%
very low	very good	a great many	very large	very close	very few	⋯	very high	4
low	very good	many	large	close	very few	⋯	high	9
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

Table 3. Boolean database from the converted category type attribute table 2

PC					PS					Software Reliability				
very low	low	medium	high	very high	very poor	poor	average	good	very good	very low	low	medium	high	very high
1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

Obviously, using raw numeric data to data mining, it will be hard to find useful rules. Therefore, firstly, adopt cloud transformation method for discretization of continuous attributes, and raise the basic concepts to the appropriate concept hierarchy, then make a soft partition to the original data set by maximum determination principle according to the gained concept sets. The six most important factors using qualitative language values may be constructed with evaluation concept sets including five grades, such as follows:

PC:{ very low, low, medium, high, very high};

PS:{very poor, poor, average, good, very good};

TEF: {very few, few, average, many, a great many};

TC:{very small, small, medium, large, very large};

TE:{ not very close, not close, medium, close, very close};

FSC: {very few, few, average, many, a great many}.

Other factors affecting software reliability can be similarly constructed. Of course, we can get more coarse or fine division according to the specific situation.

Table 2 shows the soft partition for original attribute values based on concept sets. Because of the uncertainties of soft partition based on cloud model, the percentage of the value attribute table is slightly different in different times, we can calculate the mean result for multiple soft partitions. Clearly, the amount of data is substantially reduced, and the intrinsic correlation between attributes is also highlighted after the soft partition. According to the requirement of the association rule mining algorithm, the category type attribute Table 2 can be converted to a Boolean database as shown in Table 3.

Standard Weighted Association Rules Mining Based on Cloud Model

When the weighs of the factors affecting software reliability are gained, for a given level of soft support threshold and soft confidence threshold, we can get a standard weighted association rules list with "software reliability" as rule consequent using standard weighted association rules algorithm mining based on cloud model.

If the conjunction of "PC", "PS" and "TEF" is the rule antecedent, "software reliability" as the rule consequent, the following six qualitative rules are assumed to be produced on the above conceptual granularity:

If PC is "very low", PS is "very good" and TEF is "a great many", then software reliability is "very high".

If PC is "low", PS is "very good" and TEF is "a great many", then software reliability is "high".

If PC is "medium", PS is "very good" and TEF is "a great many", then software reliability is "high";

If PC is "medium", PS is "good" and TEF is "many", then software reliability is "medium".

If PC is "high", PS is "average" and TEF is "a great many", then software reliability is "low".

If PC is "very high", PS is "average" and TEF is "many", then "software reliability is "very low".

These rules reflect the relationships between qualitative concepts such as software complexity, programmer skills, testing efforts and software reliability (also available for other software reliability indexes).

We might as well set [0,1] as the three factors' domain *U*, cloud representations of the given five qualitative concepts in the domain of software complexity are as follows:

$$C_{A_{11}} = C(0, 0.06, 0.0006), C_{A_{12}} = C(0.25, 0.06, 0.0006), C_{A_{13}} = C(0.5, 0.06, 0.0006),$$

$$C_{A_{14}} = C(0.75, 0.06, 0.0006), C_{A_{15}} = C(1, 0.06, 0.0006).$$

Cloud representations of the given five qualitative concepts in the domain of PS are as follows:

$$C_{A_{21}} = \begin{cases} 1 & x \in [0, 0.6] \\ C(0.6, 0.06, 0.0006) & else \end{cases}, C_{A_{22}} = C(0.7, 0.06, 0.0006),\ C_{A_{23}} = C(0.8, 0.06, 0.0006),$$

$$C_{A_{24}} = C(0.9, 0.06, 0.0006), C_{A_{25}} = C(1, 0.06, 0.0006).$$

Cloud representations of the given five qualitative concepts in the domain of TEF are as follows:

$$C_{A_{31}} = \begin{cases} 1 & x \in [0, 0.9] \\ C(0.9, 0.06, 0.0006) & else \end{cases}, C_{A_{32}} = C(0.99, 0.006, 6\times10^{-5}), C_{A_{33}} = C(0.999, 6\times10^{-4}, 6\times10^{-6}),$$

$$C_{A_{34}} = C(0.9999, 6\times10^{-5}, 6\times10^{-7}), C_{A_{35}} = C(1, 6\times10^{-5}, 6\times10^{-7}),$$

where A_{11}, A_{21} and A_{31} are descending half-clouds, A_{15}, A_{25} and A_{35} are ascending half-clouds.

Software Reliability Qualitative Evaluation

The summary steps of software reliability qualitative evaluation method are as follows:

1) clean up, select and assign weights for the 32 factors affecting software reliability, using DHP method based on cloud model;

2) generate some association rules about the factors and software reliability, using standard weighted association rule mining method based on cloud model;

3) given a new determined input, activate the corresponding rules based on these qualitative rules, then generate uncertainty output from the rule generator based on cloud model.

Based on multi-conditions and multi-rules reasoning mechanism, firstly, activate these six qualitative rules using the input values through quantitative analysis; secondly, synthesize the above generated clouds using geometric cloud technology; finally, the expected value *Ex* of the generated geometric cloud *C*(*Ex*, *En*, *He*) can be set as the quantitative assessment of software reliability. Reference to the software reliability evaluation conception set, the quantitative evaluation of the software cloud reliability is actually a virtual cloud in software reliability domain [0, 1]. With the input of expectation *Ex*, *X* condition cloud generators may individually output the certain degrees corresponding to the different evaluation concepts. Finally, according to maximum determination principle for membership concept may judge the software reliability.

Acknowledgments

This work is financially supported by National Nature and Science Foundation of China (61463012), Natural Science Foundation of Hainan Province (614233, 114013, 613161, 613162), and Qiongtai Teachers College Scientific Research Project (qtky201304).

References

[1] L. Deyi and D. Yi: *Artificial Intelligence with Uncertainty*, National Defense Industry Press, Beijing(2005)

[2] D. Yi: *Research and Application of Association Rules in Data Mining*, PhD Thesis for PLA University of Science and Technology, Nanjing (2000)

[3] Z. Xuemei and H. Pham: The Journal of Systems and Software, Vol.50 (2000), p.43

[4] W.Tiejiang and L.Meng: Computer Engineering and Applications, Vol.38(2002) , p.23

[5] W. Shuliang: *Data Field and Cloud Model Based Spatial Data Mining and Knowledge Discovery*, PhD Thesis for Wuhan University, Wuhan(2002)

[6] L. Deren, W. Shuliang and L. Deyi: *Spatial Data Mining Theories and Applications*, Science Press, Beijing(2006)

[7] L. Ziqiang and Z. Guangwei: Journal of Frontiers of Computer Science and Technology, Vol.1(2007),p.234