ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Normal Distribution and Questionnaire Based Assessment of Risk in Software Reusability

[1]Iyapparaja. M and [2]Dr.Sureshkumar. S

[1]*Research scholar, Anna University, Chennai, Tamil nadu, INDIA.*
[2]*Principal, Vivekanandha College of Technology for Women, Tiruchengode, Affiliated By Anna University, Tamilnadu, INDIA.*

ARTICLE INFO

Article history:
Received 12 March 2015
Accepted 28 April 2015
Available online 5 May 2015

Keywords:
Software reusability, Risk assessment, Test case generation, Normal distribution

ABSTRACT

Background: Analysis of risk is a process that includes relief of lapses amid different methods of software programming development. Early distinguishing proof of risk ought to abbreviate the product improvement finishing time. It will likewise give proficient arrangements. **Objective:** A complete risk examination alongside Identification, observing and assessment is certain to guarantee high quality software programming development. **Results:** In this paper focusing two different risk assessment approaches namely Questionnaires bases risk assessment (QBRA) and Normal distribution risk assessment (NDRA), those helps to identify and analyze the severity level of risk while reusing software component from existing software reusable components repository (SRCR). **Conclusion:** Output of each approaches will be considered with different assumptions and discuss how the severity level of risk is assessed.

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: Iyapparaja. M, and Dr.Sureshkumar. S., Normal Distribution and Questionnaire Based Assessment of Risk in Software Reusability. ***Aust. J. Basic & Appl. Sci.,*** *9(11): 160-166, 2015*

INTRODUCTION

software component reuse is developing strategies that give a confirmation to quicken advancement time of programming venture furthermore steadily decreases the creating cost regarding scrutinize and fixing the pre-fabricated component in to overall software project.

Software reuse conveys the noteworthy risk all through the software project development. Risk is an element that includes questionable danger and can block the improvement complete software project development (Ahdieh Sadat Khatavakhotan and Siew Hock Ow ,2012). Accordingly chance recognizable proof procedures and classifying risk over the few periods of the segment based programming advancement are great critical so that the seriousness of the risk can be lessened.

The accessibility of examining the software reusable component can't end up being a productive one. Subsequently the conceivable results ought to be sensible to withstand its improvement (Ana Lúcia Figueiredo Facin *et al* ,2014)(Anil Kumar Tripathi and Manjari Gupta,2006). The risk assessment approaches endow with the expanded number of the conceivable results with further investigation risk and its decrease (Foo S. W. and Muruganantham A, 2000) (Sebastian Spaeth *et al*, 2008).

II. A. Assessment of Extracted Function Or Package:

A model has been developed and implemented in member function or package using object oriented programming language C++ and Java (Nancy Bazilchuk and Parastoo Mohagheghi,2005)(Neelamadhab Padhy *et al*, 2015) (Ritika Chaudhary and Ram Chatterjee, 2014). Once it extracts the function or package from existing software project and verifies the chance of risk to generate while integrating into present software project development (Charu Singh *et al*, 2014).

Extracted module may chance to appear in three varieties like,

1. Exact Match
2. Partial Match
3. Rejected Match

1) Exact Match (EM):

Identified source code reusable module looks exactly same as the module which the developer expects for current software project based upon the attributes fixed in scrutinize process.

2) Partial match (PM):

Corresponding Author: Iyapparaja. M, Research scholar, Anna University, Chennai, Tamil nadu, INDIA.

To modify the argument description and if require adjust very few syntax and semantic description of Identified source code reusable module will help to adapt the usability of present software project development.

3) Rejected Match (RM):

The Identified source code reusable module entirely differs from expected source code module description given by developer.

In software project development identification of reusable source code will stick by these three varieties as illustrated in following Table 1.

Table 1: Identifying max() reusable module

Program for finding maximum between two numbers	Different possibilities of reusable modules

#include<iostream.h>
#include<conio.h>
int max(int,int);/*Declare reusable module*/
void main()
{
int a,b,c;
clrscr();
cout<<"enter the value for a & b";
cin>>a>>b;
{
Try
{
/* check the parameters of reusable module*/
if(a==0&&b==0)
throw exception1;
else if(a==b)
throw exception2;
}
catch(exception1)
{
cout<<"Give non zero value for a&b";
}
catch(exception2)
{
cout<<" Give distinct value for a&b";
}
else
c=max(a,b); /*Calling expected reusable module*/
cout<<"maximum in"<<a<<"and"<<b<<"is"<<c;
getch();
}
	/*Exact match reusable module*/
	1. int max(int x, int y)
	{
	if(x>y)
	return x;
	else
	return y;
	}
	/*partial match reusable module*/
	2. void max(int p, int q)
	{
	if(p<q)
	cout<<"p is big";
	else
	cout<<"q is big";
	}
	
	/* Rejected match reusable modules*/
	3. Void maximum(int a,int b,int c)
	{
	if(a>b&&a>c)
	cout<<"a is big";
	else if(b>c)
	cout<<"b is big";
	else
	cout<<"c is big";
	}

Above program in Table 1, tries to reuse the module for its task completion but while reusing there may be a chance for getting above three specified varieties software reusable module as discussed in above section II. Calling of any improper reusable module for proposed software project may chance to produce risk factor while software project execution (Firesmith D.G.,1993)(R. Jayasudha and S. Subramanian, 2014). In general there are several different types of risk are consider in software project development(Md. Iftekharul Alam Efat *et al*, 2014).

1. Business risk
2. Scheduled risk
3. Technical risk
4. Environment risk
5. Operational risk
6. System complexity

.

Among these, in this paper concentrates technical risk because which helps to focus various factors like,

i. Changing requirement
ii. Complex to implement
iii. Difficult project integration
iv. Lack of individual strength
v. No advanced technology is available

Due to above listed factors, the severity level of risk is classified into three categories Cosmetic risk, Grievous risk and Pessimistic risk(Han Van Loon, 2007). The presence of risk in software component reusability is assessed by using following two approaches, namely,

1. Questionnaire based risk assessment (QBRA)
2. Normal distribution risk assessment (NDRA)

The complete data flow of risk analysis is depicted in following Figure.1

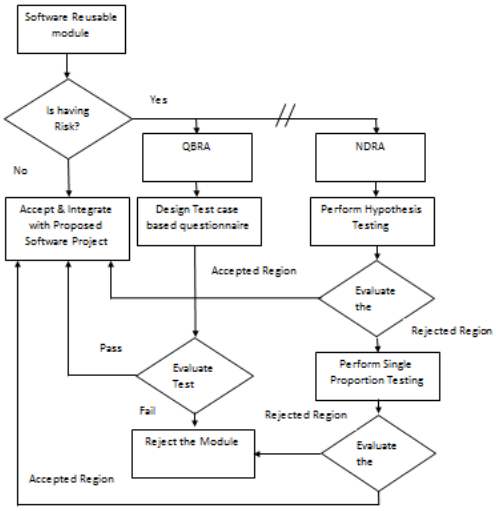

Fig. 1: Data flow diagram for Risk Assessment based QBRA & NDRA

III. A. Questionnaire Based Risk Assessment:

The primary possible amount of risks in software reusable component can be measured by answering the different questionnaires in the following test cases. It helps to identify the risk associated with reusable component before integrate into present software project. The following sets of questionnaires are framed in the form of test cases as shown in Table 2., and those need to be focused for complete assessment.

Table 2: Questionnaires Bases Test Case Assessment

Test case_id	Questionnaires	Expected Result	Actual Result	Result
TCQ_01	Examine whether the identified suitable reusable software module (in chapter 4) in existing repository is capable to adapt with present software language domain.	Adopt to present domain	As Expected	Pass
TCQ_02	Validate the identified module name is same as the developer expected module for current software project development.	Match with expected module name	As Expected	Pass
TCQ_03	Whether all arguments are declared as function parameters or local variables to the function.	Declared as function parameters	As Expected	Pass
TCQ_04	Check whether the count of specified arguments are available within the module either as an parameters of function or as local variables even the extracted module name as same as specified module.	Count of arguments are as expected	As Expected	Pass
TCQ_05	Are the specified arguments either generic or non-generic one?	Non-generic one	As Expected	Pass
TCQ_06	Check whether the extracted module meet its specified usage threshold value as specified in scrutinize process (in section 4.3.2)	Meet its UT value	As Expected	Pass
TCQ_07	Examine whether the individual strength of reusable module by validating coupling & cohesion metrics (in chapter 5)	Met all coupling & cohesion metrics	As Expected	Pass
TCQ_08	Check the stability level of the reused software module.	Module in stable	As Expected	Pass
TCQ_09	Is the reused module producing any unsatisfied outcome and which may cause failure?	Satisfied outcome	As Expected	Pass
TCQ_10	When or which moment the software reusable module arose the failure?	Not arise any failure	As Expected	Pass
TCQ_11	What is the severity level (Grievous risk or Pessimistic risk or cosmetic risk) of that risk?	Cosmetic risk	As Expected	Pass

These various set of questionnaires are framed in the form of test cases in above table 2., and get the answer for it as many times as required by the developer using Junit testing tool, which help to analyze the severity level of risk and document the output against present software project development.

IV.A. Normal Distribution Assessment:

Normal Distribution is the process of finding the probability of risk, undesirable event in a large group of quantity or in an augmentation of information. It is practically impossible to calculate and qualify all modules in given specified time.

Australian Journal of Basic and Applied Sciences, 9(11) May 2015, Pages: 160-166

To ascertain vast measure of such modules, Normal Distribution is being utilized as a part of different quantitative study. Despite the fact that different distributions of computations are accessible, to get a reusable module from substantial amount of SRC Repository, normal distribution is opted. In this research, Normal distribution is connected to examine risk seriousness level in software reusable code.

Samples n - numbers of modules are extracted from software reuse component repository and those numbers of modules are validated by normal distribution method to find the occurrence of risk in those modules.

All risked modules are being neglected and remaining adaptable modules are allowed to integrate with present software project development.

The equation for normal distribution is,

$$Z = \frac{\overline{X} - \mu}{\sigma / \sqrt{n}} \quad (1)$$

$$\overline{X} = \sum_{i=0}^{n} \frac{C_i}{n} \quad (2)$$

Where,

Z = Normal Distribution for reused software module having risk

$\overline{X}$ = Mean of sample reused software modules

μ = Mean of the population reused software modules in SRC Repository

σ = Standard Deviation of population reused software modules

n = Total number of reusable modules in the sample

C_i = i^{th} value of Selected sample reusable modules

The equation (1) gives the normal distribution where Z is the normal distribution of sample acceptable reused software modules. Sample modules are selected from 'n' number of modules to asses for its risk and mean of those sample modules at Software reusable component Repository (SRCR) can be found by using the equation (2).

A statistical hypothesis test is a strategy for settling on choices utilizing information, whether from a controlled examination or an observational study (not controlled). In statistics, an outcome is called statistically significant on the off chance that it is unrealistic to have happened by chance alone, as per a foreordained threshold probability, the significance level. In probability, these choices are quite often made utilizing null-hypothesis tests.

Hypothesis testing permits utilizing sample data to test a case around a population, for example, testing whether population mean equivalents sample mean. Normal distribution has following different terminologies like,

i. Sampling Distribution:

Consider all possible samples of size n drawn from a given population at random and calculate mean values of these samples. If it groups these different means according to their frequencies, the frequency distribution so formed is called sampling distribution.

The statistic is itself a random variety. Its probability distribution is often called sampling distribution. It can be thought of as below:

All possible samples of given expected module attributes are taken from the population and for each sample, the statistic is calculated. The values of the statistic form its sampling distribution.

ii. Statistical Hypothesis:

In endeavoring to achieve choices about population on the premise of sample perceptions, it makes suspicions about population, which are not so much genuine, are called statistical hypotheses.

iii. Null Hypothesis (H_0):

The hypothesis tested for conceivable dismissal under the presumption that it is genuine typically called null hypothesis. The null hypothesis is a theory which mirrors no change or a distinction. It is typically signified by H_0.

iv. Alternate Hypothesis (H_1):

The alternative hypothesis is the announcement which mirrors the circumstance foreseen to be remedy if the null hypothesis isn't right. It is typically indicated by H_1.

v. Region of Acceptance:

The set of values of the test statistic for which can be fail to reject the null hypothesis.

vi. Significance Level of a Test (α):

It is the probability level below which the null hypothesis is rejected. Generally, 10%, 5% and 1% level of significance are used.

vii. Critical Region or Region of Rejection:

The critical region of a test of statistical hypothesis is that region of the normal curve which corresponds to the rejection of null hypothesis. The shaded portion in the following figure is the critical region which corresponds to 5% level of significance.

viii. Critical Values or Significant Values:

The sample values of the statistic beyond which the null hypothesis will be rejected are called critical values or significant values. The tabulated value of Z for each tail test with different set of level of significance is illustrated in following Table 3.

Table 3: Significant Values

Type of Test	Level of Significance (LoS)		
	0.01 (or) 1% LoS	0.5 (or) 5% LoS	0.1 (or) 10% LoS
One tailed test	2.33	1.645	1.28
Two tailed test	2.58	1.96	1.645

ix. One Tailed and Two Tailed Tests:

Two tails of the sampling distribution of the normal curve are used, the relevant test is called two tailed test.

The alternative hypothesis H_1: $\mu_1 ! = \mu_2$ is taken in two tailed test for H_0:$\mu_1=\mu_2$. When only one tail of the sampling distribution of the normal curve is used, the test is described as one tail test. For H_0: $\mu 1= \mu 2$, the formulated alternative hypothesis is either H_1: $\mu 1> \mu 2$ or H_1:$\mu 1<\mu 2$ (R. Pannerselvam, 2009).

1) Procedure for Testing a Hypothesis:

i. Formulate Null Hypothesis (H_0) and Alternate Hypothesis (H_1).

ii. Choose the level of significance either 1%,5% or 10%.

iii. Compute the test statistic Z, using the data available in the problem

iv. Pick out the critical value at using Z.

v. Draw conclusion: if $|Z|<Z\alpha$, accept H_0 at α % level, Otherwise reject H_0 at α % level means accept H_1.

Assumption 1:

Consider the situation in which the n- number of reusable software components are extracted from SRC Repository which following normal distribution. The expected mean μ of the modules extracted at the particular time interval having a risk is 100 modules. The standard Deviation for extracting reusable modules in distinct project available in SRC Repository is 25 modules. To identify the modules which have risk a random of 20 samples are taken. For these, the mean of the extracted reusable software modules from sample is found to be 115 modules. To check whether the extracted reusable modules of distinct or same project are not having risk at a significant level of rejection which is 0.05(5%).

Let X is a random variable indicating the module risk, which follows a mean of 100 numbers of modules with standard deviation (σ) of 16 modules

Then,

μ =100 modules.

σ =25 modules

Sample size, n = 20

Sample mean, $\overline{X}$ = 115

Step 1: Formulate the Hypothesis:

H0: μ=100, Null Hypothesis: extracted reusable modules of distinct or same project are not having risk.

H1:μ > 100, Alternative hypothesis: extracted reusable modules of distinct or same project are having risk

Step 2: Choose Level of Significance:

Level of Significance as (5%) so $\alpha = 0.05$.

Step 3: Compute test statistics:

Test Statistic: $$Z = \frac{\overline{X} - \mu}{\sigma / \sqrt{n}}$$

$$= \frac{115 - 100}{25 / \sqrt{20}}$$

$$= 15 / 5.59$$

$$Z = 2.683$$

Step 4: Pick Tabulated value of Z:

From the Table 2 for one tailed test value of Z = 1.645.

Step 5: Conclusion:

Since |Z|=2.683>1.645, H0 is Rejected at 5% level of significance,

The calculated Z value (2.683) is more than tabulated value of $Z\alpha$ (1.64) and it falls in rejection region. Hence the formulated Null hypothesis is wrong. It has to accept the alternate Hypothesis which is same as extracted reusable modules of distinct or same project are having risk.

$\therefore$ Significance level α (0.05) < Z (Normal Distribution for risk in extracted reusable modules). So the Normal distribution curve for the calculated value is laid down rejected risky region as shown in Figure 2.

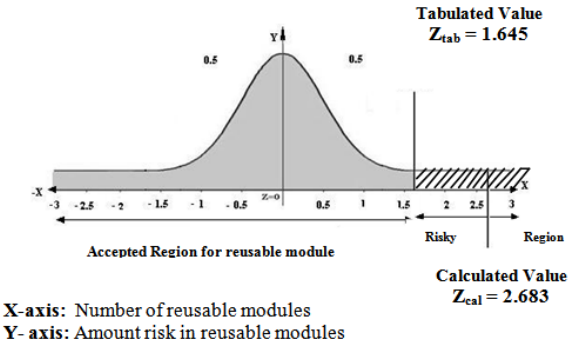

Fig. 2: Risky reusable modules Rejection Region

Next step is to check whether the risk is caused by more number of extractions given by the developer or due to the problem exist in extracted reused modules. In order to achieve this, single proportion test is used.

2) Single Proportion Test:

Single proportion test utilizes the information about the extracted modules and check whether those modules have predefined risk.

If x is the number of modules possessing a certain attribute in a sample of n number of modules, then the sample proportion p = x/n. Consider a sample of size n with proportion P taken from a population. Let P be the population proportion. To test whether,

(i) The difference between sample proportion P and population proportion p is significant or not.

(ii) The sample has been chosen from the population, then proceed as follows.

Let the null hypothesis be H_0: p=P i.e., p has a specified value. The alternative hypothesis is H_1: p! = P

The equation (3) which describes the normal test is

$$Z = \frac{P - p}{\sqrt{p(1-p)/n}} \qquad (3)$$

If α = 0.05 is the level of significance, then compare the calculated Z with table value 1.96.

|Z|<1.96, H_0 is accepted. Otherwise it is rejected. In general, |Z|<3, H_0 is accepted, Otherwise H_0 it is rejected.

Assumption 2:

Consider the following example, in a Java based applications, a sample of 100 Modules are drawn. In that, assume 87 modules are having risk and remaining are authentic modules. In general, both risky modules and authentic modules are equally distributed in the particular java application at 5% (0.05) level of significance.

$$Z = \frac{P - p}{\sqrt{p(1-p)/n}}$$

Z=Normal test for Proportion.

P= Proportion of Population

p=Sample Proportion of risky modules

n=Total no of samples reused modules from the population

H0 : p=1/2, **Null Hypothesis:** Risky modules and authentic modules are equally distributed

H1 : p>1/2 , **Alternate Hypothesis:** Risky modules and authentic modules are not equally distributed that means Large number of risky modules in that group

Level of Significance Fixed as 5%

From the Table 2, for one tailed test value of Z = 1.645.

$$Z = \frac{P - p}{\sqrt{p(1-p)/n}}$$

$$Z = \frac{\left(\frac{87}{100}\right) - \left(\frac{1}{2}\right)}{\sqrt{\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)/100}}$$

$$Z = \frac{0.87 - 0.5}{\sqrt{0.25/100}}$$

$$Z = \frac{0.37}{\sqrt{0.0025}}$$

$$Z = \frac{0.37}{0.05}$$

Here, |Z|=8.4>1.645, Z (Calculated value)>Z (Tabulated value), Hence, H_0 is rejected and H_1 is

accepted that means Alternate Hypothesis: Risky modules and authentic modules are not equally distributed that means large number of risky modules in that group in java application at the time of testing.

As per result in normal distribution curve the reusable modules will be stick with risky region as shown in figure 3.

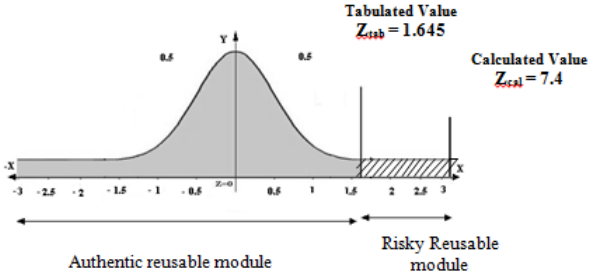

Fig. 3: Identifying reusable module having risk

Once the risky module is identified, those modules will be neglected and only forward the authentic modules for further integration in present software project development.

Conclusion:

Reusing the software code from existing SRC Repository and analyzing its strength through various coupling and cohesion metrics provides the exact reusable module to the developer. But in order to strengthen the software project development, It became mandatory for analyzing the risk in software reusable component in order to reduce the possible number of imminent risks in proposed software project. In this chapter use two different risk assessment approaches for identifying the presence or absence of risk in extracted reusable module.

REFERENCES

Ahdieh Sadat Khatavakhotan and Siew Hock Ow, 2012.Rethinking the Mitigation Phase in Software Risk Management Process: A Case Study,IEEE Fourth International Conference on Computational Intelligence, Modelling and Simulation, pp: 381-386.

Ana Lúcia Figueiredo Facin *et al*, 2014. Knowledge Management to Deal with Risk in the Process of Software Development: A Case Study, IEEE Proceedings of PICMET '14: Infrastructure and Service Integration, pp: 2673-2681.

Anil Kumar Tripathi and Manjari Gupta, 2006. Risk analysis in Reuse Oriented Software Development, International Journal Information Technology and Management, 5(1): 52-65.

Charu Singh *et al*, 2014. An Estimation of Software Reusability using Fuzzy Logic Technique, IEEE International Conference on Signal Propagation and Computer Technology (ICSPCT). pp: 250-256.

Firesmith, D.G., 1993. Testing Object Oriented Software, Proceedings of Eleventh International Conference on Technology of Object Oriented Languages and Systems, pp: 406-427.

Foo, S.W. and A. Muruganantham, 2000. Software Risk Assessment Model. Proc. of the 2000 IEEE International Conference on Management of Innovation and Technology, 2: 536-544.

Han Van Loon, 2007. A Management Methodology to Reduce Risk and Improve Quality, IT Professional, 9(6): 30-35.

Md. Iftekharul Alam Efat *et al*, 2014. Feature Prioritization for Analyzing and Enhancing Software Reusability, IEEE 3rd International Conference On Informatics, Electronics & Vision.

Nancy Bazilchuk and Parastoo Mohagheghi, 2005. The Advantages of Reused Software Components, IEEE Transactions on Software Engineering, 23(9): 556-565.

Neelamadhab Padhy *et al*, 2015. A Systematic Literature Review of an Object Oriented Metric: Reusability, IEEE International Conference on Computational Intelligence & Networks, pp: 190-191.

Ritika Chaudhary and Ram Chatterjee, 2014. Reusability in AOSD – The Aptness, Assessment and Analysis, IEEE International Conference on Reliability, Optimization and Information Technology-ICROIT2014, pp: 34-39.

Jayasudha, R. and S. Subramanian, 2014. Ontology Based Software Storage Repository to Enhance Software Reuse, Aust. J.Basic & Appl. Sci., 8(16): 413-418.

Pannerselvam, R., 2009. Research Methodology, PHI Learning Private Limited

Sebastian Spaeth *et al*, 2008.A Lightweight Model of Component Reuse: A Study of Software Packages in Debian GNU/Linux, SSRN.