Journal of Software Engineering and Applications, **2012, 5, 395-401**
http://dx.doi.org/10.4236/jsea.2012.56046 Published Online June 2012 (http://www.SciRP.org/journal/jsea)

A Model for Software Product Quality Prediction

Brijendra Singh, Suresh Prasad Kannojia

Department of Computer Science, University of Lucknow, Lucknow, India.
Email: drbri_singh@hotmail.com, spkannojia@gmail.com

Received March 28th, 2012; revised April 25th, 2012; accepted May 2nd, 2012

ABSTRACT

When the expression "Software Quality" is used, we usually think in terms of an excellent software product that fulfills our expectations. These expectations are based on the intended use. Number of models has been proposed for evaluation of software quality based on various characteristics. In this paper quality of software product is defined in terms of basic components as constituent part of any program or software and proposed a software quality prediction model based on basic components. It has been justified with example that if any software quality model uses the tacit knowledge that will be better than any other model in terms of quality.

Keywords: Software Product; Software Quality; Quality Model; Quality Attributes

1. Introduction

If we are to talk intelligently about the quality of thing or the quality of product, we must have in mind a clear picture of what we mean by quality. Quality defined by the various quality gurus in various ways, depending on the user perspective [1-7].

- Dr. Barry Boehm thinks of quality as Achieving high levels of user satisfaction, portability, maintainability, robustness, and fitness for use.
- Phil Crosby has created the definition with the currency because of its publication in his famous book "Quality is free". He states that quality means "conformance to user requirements".
- Tom McCabe, the software complexity specialist, defines quality as "High levels of user satisfaction and low defect levels, often associated with low complexity".
- John Musa of Bell Laboratories states that quality means combination of "low defect level, adherence of software functions to users needs, and high reliability".

Traditionally, the quality of product is defined in terms of its fitness of purpose. Although fitness of the purpose is satisfactory definition of quality for hardware products, but it is not satisfactory for software products.

To give an example of why this is so consider a software product that is functionally correct. That is, it correctly performs all the functions that have been specified in its SRS documents. Even though it may functionally correct, we cannot consider it to be a quality product, if it has an almost unusable user interface. Therefore, the traditional concept of quality as fitness of purpose for software products is not wholly satisfactory.

The concept of software quality is not as easily definable. There are various possible quality characteristics of software, and there is even an international standard for this [8,9].

For each quality characteristics, a set of attributes, which can be measured, is determined such a definition helps in evaluating the quality of software [10].

Effective software quality evaluation and assurance requires models that describe what the software quality is and how can it be traced back to the development process. Two approaches can be followed to ensure software quality. One is focused on direct specification and evaluation of the quality of software product [10], while the other is focused on assuring high quality of process by which the product is developed.

Number of models [5-7,10-12] has been proposed to evaluate the quality of software product, based on various quality characteristics.

McCall quality model [5] attempts to bridge the gap between users and developers by focusing, on a number of software quality factors. The evaluation of software has been done by Boehm's quality model [7,10], uses a given set of attributes and metrics. More recently, model has been developed by Dromey's [11,12], which is focusing on the relationship between the quality attributes and the sub attributes, as well as attempting to connect software product properties with software quality attributes.

As we have seen number of attributes reliability, usability, efficiency, maintainability and portability are common in number of models [5-7,10-12].

This paper proposed, a quality model based on basic components. Basic components have been chosen based on the five quality attributes: reliability, usability, efficiency, maintainability and portability. This model focused on direct specification & evaluation of quality of software product, which satisfies to the user. Tacit knowledge is having very critical role to choose the basic components, that reflect the quality of model, and it has been proved in this paper.

Quality experts and the application of recognized quality references, especially ISO9000 standards and performance excellence models [8,9], have traditionally emphasized explicit knowledge, e.g. documentation, written descriptions and procedures, specifications, arguments, and information records. Tacit knowledge has not been considered as consistency and in the same details and important part of knowledge from the business point of view is tacit. Proposed model has been illustrated with example and shown the importance of tacit knowledge for software product quality prediction.

1.1. Definitions

- Software Reliability: The ability of a program to perform a required function under stated conditions for a stated period of time.
- Usability: Usability is concerned with the quality of the user interface, its design and performance characteristics, ease of reusing software in a different context or the extent that it is reliable, efficient and human-engineered.
- Efficiency: The position taken is that computational and logical redundancy is important factors that affect the efficiency of a program. Ensuring that there is a match between program control structure and data structure also makes a contribution. Efficiency also can be defined in terms of execution efficiency and storage efficiency means use of resources (processor time, storage) or the extent that it fulfills its purpose without waste of resources.
- Maintainability: There is a widely held belief that software, which is very easy to maintain, is software of high quality. If the minimum effort required to locate and fix a fault in the program. It leads to easily maintainable.
- Portability: The effort required to transfer a program from one environment to another or the extent that it can be operated easily and well on computer configuration other than its current one.

1.2. Proposed Model

Proposed model emphasis is to connect the basic component with common quality attributes and product functionality, which yield to predict the quality of s/w product. The basic component is defined as the constituent part of any program or software, such as NLOC, NHDL, NVERd, NinputFUN, NcommL, NLOOP, NoutputFUN, Nclass, Nfunct/CL. The basic components of software product is used as a metrics; on the basis of these metrics one can answer the following general questions of customer/user's:

1) How well can I use it?
2) How easy is it to understand, modify and retest?
3) Can I still use it if I change my environment?

These metrics lead to the quality attribute namely reliability, usability, efficiency, maintainability and portability. In this paper quality of software product is defined in terms of its basic components via which it's constructed, and each component of product is uniquely characterized.

In this model we also uses the conformance to applicable specification & standard that is agreement between user and software product developer. Quality of conformance refers to the extent to which the software product complies with the specifications, standards, and workmanship criteria (tacit knowledge) imposed upon its developer such as:

1) Software must be use for specific purpose for example: calculation of factorial of a number;
2) Input of software must be specific based on user, and software must be tested for input data;
3) Software must produce the correct output as desire by user based on his input data;
4) Software should be User friendly to satisfy user.

The most important quality attributes of a software product can poses is usefulness; *i.e.* the software must satisfy user needs.

Software developer used to claim the various qualities attributes for specific quality of product as—reliability, usability, efficiency, maintainability and portability. **Figure 1** shows the block diagram of proposed model for prediction of quality. With the help of proposed model, it

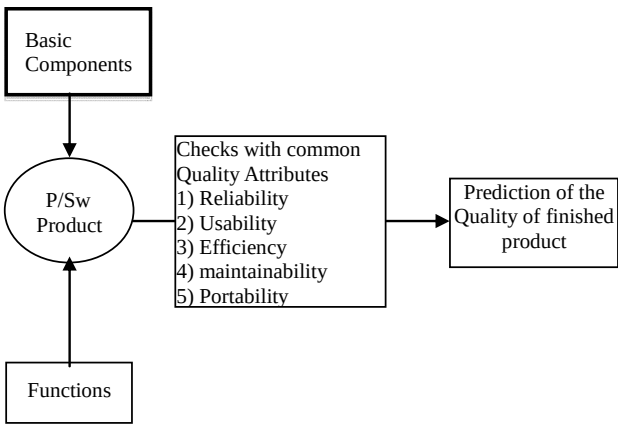

Figure 1. Block diagram of proposed model for prediction of quality.

is possible to predict the quality of conformance based on the agreement between user and software developer. In next section, model has been illustrated and satisfies the various attributes for specific quality of product.

2. Illustration of the Proposed Model

For the illustration purpose we have taken a very simple program, which is developed by the two different programmers (in C, C++ programming language) independently. For example program: calculation of the factorial of a number, the code of taken example is listed in annexure-I and II. Assume finished software product taken as a input developed by the programmer-I and programmer-II separately.

The basic components chosen from the program as shown in annexure-I, and metrics of the basic components has been shown in **Table 1**.

Figure 2 Shows the graphical representation of the **Table 1**, in this graph there are two series line dark and light, the dark lines indicate the basic metrics of the program developed by programmer-I. Similarly light lines indicate that basic components metrics of the program developed by programmer-II, in this graph there are one trends lines corresponding to each series lines. Programmer-II having the tacit knowledge to develop the program but programmer-I have no tacit knowledge. On the basis of these trends lines we are able to predict the quality of software, so program developed by programmer-II is more reliable, usable, efficient, maintainable and portable.

The basic components chosen from the program as shown in annexure-II and metrics of the basic components has been shown in **Table 2**.

Figure 3 Shows the graphical representation of the **Table 2**, in this graph there are two series line dark and light, the dark lines indicate the basic metrics of the program developed by programmer-I. Similarly light lines indicate that basic components metrics of the program developed by programmer-II, in this graph there are one trends lines corresponding to each series lines. Programmer-II having the tacit knowledge to develop the program but programmer-I have no tacit knowledge. On the basis of these trends lines we are able to predict the quality of software, so program developed by programmer-II is more reliable, usable, efficient, maintainable and portable.

On the basis of graphical analysis of the basic components of the product program or s/w, we analyze that:

1) Software Reliability: On the basis of trends lines of the graph as shown in **Figures 2** and **3** shows that program developed by programmer-II is more reliable as compared to the program developed by programmer-I. In ordinary case program developed by programmer-I give the correct out put, but some times its fail to give the correct result where as program developed by programmer-II always gives a correct result.

2) Usability: Compared the programs/software, which is developed for the same purpose by two independent developers. The program/software developed by programmer-I is compact less readable, useful, not easy to handle. It means less users friendly. Because number of code lines is less, number of comment line is less, where as comment is used to give the detail of program how they work, how user can go on to perform the desired task. Usability of software/program developed by programmer-II is higher than programmer-I.

3) Efficiency: Program/software developed by programmer-II program/software developed by programmer-I has no computational and logical redundancy. In this program it has the power to take the range of inputs integer to float. But it is not in case of program/software developed by programmer-I. Therefore program/software developed by programmer-II is more efficient than the program/software developed by programmer-I.

4) Maintainability: Program developed by programmer-I is constructed/coded in compact form in terms of the numbers of lines of code, that's why some times it is very easy to locate and correct the error, but some times

Table 1. The basic components of annexure-1 program is tabulated.

Small program Developed by	NLOC	NHDL	NVERd	NinputFUN	NoutputFUN	NcommL	NLOOP
Programmer-I	21	02	4	01	05	0	01
Programmer-II	34	02	4	01	05	3	01

Table 2. The basic components of annexure-2 object oriented program are tabulated.

Small program Developed by	NLOC	NHDL	NVERd	NinputFUN	NoutputFUN	NcommL	NLOOP	NClass	NFunction/CL
Programmer-I	30	02	03	01	04	0	01	01	02
Programmer-II	50	02	03	01	04	7	01	01	02

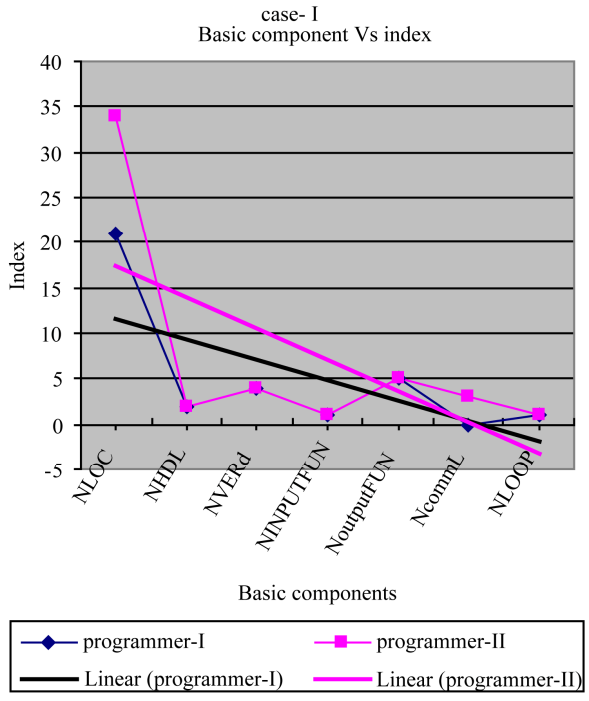

Figure 2. Basic components vs index as per Table 1.

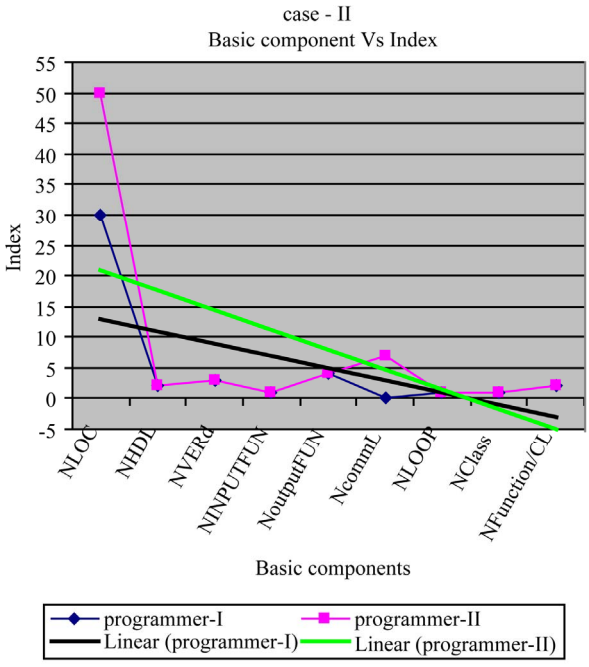

Figure 3. Basic components vs index as per Table 2.

its tedious task. Where as program developed by programmer-II is constructed in detailed form, so normal user can understand easily and easy to modify. It mean easy to maintain. Therefore maintainability of program/ software developed by programmer-II is higher than programmer-I.

5) Portability: The programs *i.e.* as shown in annexure-I & II can run on the MS-DOS or windows environments. Therefore portability of the both program/software is good in the availability of the common environment.

Proposed model satisfies the agreement between the user & software product developer, which has been shown through illustration.

3. Conclusion

We have defined quality of software product in terms of basic components as constituent part of any program or software. Model for software product quality prediction has been proposed. Model has been illustrated with example that the program/software developed by the programmer-II is more reliable, usable, efficient, maintainable and portable. As compared to the program/software developed by the programmer-I. Model has been tested on various example's that shows any program is developed having tacit knowledge, program/software have good quality. It has been illustrated with example in paper & result shows that program/software developed by programmer-II is having better quality than program/ software developed by programmer-I.

REFERENCES

[1] S. Brijendra, "Quality Control & Reliability Analysis," 3rd Edition, Khanna Pub, Delhi, 2011.

[2] N. S. Godbole, "Software Quality Assurance Principles and Practice," Alpha Science International Limited, 2004.

[3] A. K. Rae, H. L. Hausen and P. Robert, "Software Evaluation for Certification: Principles, Practice and Legal Liability," McGraw Hill, New York, 2007.

[4] J. D. Musa, "Software Reliability Engineering," 2nd Edition, Author House, Bloomington, 2004.

[5] J. A. McCall, P. K. Richards and G. F. Walters, "Factors in Software Quality," *National Technical Information Service*, Vol. 1-3, 1977.

[6] B. Kitchenham and S. L. Pfleeger, "Software Quality: The Elusive Target [Special Issues Section]," *IEEE Software*, Vol. 13, No. 1, 1996, pp. 12-21. doi:10.1109/52.476281

[7] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod and M. Merritt, "Characteristics of Software Quality," North-Holland, Amsterdam, 1978.

[8] International Standards Organization, "Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for Their Use," ISO/IEC is 9126, Geneva, 1991.

[9] ISO, International Organization for Standardization, "ISO 9126-1:2001, Software Engineering—Product Quality, Part 1: Quality Model," 2001.

[10] B. W. Boehm, J. R. Brown and M. Lipow, "Quantitative Evaluation of Software Quality," *Proceedings of the 2nd International Conference on Software Engineering*, Los Alamitos, 13-15 October 1976, pp. 592-605.

[11] R. G. Dromey, "Concerning the Chimera [Software Quality]," *IEEE Software*, Vol. 13, No. 1, 1996, pp. 33-43. doi:10.1109/52.476284

[12] R. G. Dromey, "A Model for Software Product Quality," *IEEE Transactions on Software Engineering*, Vol. 21, No. 2, 1995, pp. 146-162. doi:10.1109/32.345830

Notations:

NLOC: Number of lines of code.
NHDL: Number of hidden file used.
NVERd: Number of variable declared.
NinputFUN: Number of input function used for data entry.
NcommL: Number of comment lines.
NLOOP: Number of Loop used (process repetition).
NoutputFUN: Number of out put functions used (for display the intermediate result or final Result).
Nclass: Number of class defined.
Nfunct/CL: Number of function/class.

Annexure-1

Case I: Program developed in C Language for the calculation of factorial of a number.

Program developed by programmer-I

```
#include <stdio.h>
#include <conio.h>
main ( )
{ int n,f,i;
print (“Please Enter the Number”);
scanf(“%d’’, &n);
if (n<0)
{ printf(“Try Again”);
}
else if (n= =1)
{f=1;
printf(“factorial is=%d”,f);
}
else
{ f=1;
for (i=1; i<n; i++)
{ f=f*i;
}
printf(“%d”,f); }
return(0);
}
```

Program developed by programmer – II

```
#include <stdio.h>
#include <conio.h>
main ( )
{
// Declare the variable
int i;
int f,n;
clrscr();
//Take the input value from the user
printf (“Please Enter the Number”);
scanf (“%d’’, &n);
//check the input value whether it is positive or Negative
if (n<0)
{
printf(“factorial is not possible”);
printf(“Try Again”);
}
elseif (n= =1)
{
f=1;
printf(“factorial is=%d”,f);
}
else
{
f=1;
for (i=1; i<n; i++)
{
f=f*i;
}
printf(“%d”,f);
}
return(0);
}
```

Annexure-2

Case II: Program developed in Object Oriented Programming Language (C++) for the calculation of factorial of a number.

Program developed by programmer-I

```
#include <iostream.h>
#include <conio.h>
class factorial
{ private:
int i,n,f;
public:
void enterdata (void)
{ cout<<” Please Enter the Number”;
    cin>>n;
}
void calculation (void)
{ if (n<0)
cout<<”factorial is not possible”;
elseif (n==1)
{ f=1;
cout<<”factorial is=”<<f;
}
else
{f=1;
for (i=1; i<n; i++)
{f=f*i;
}
cout<<”factorial of”<<n<<”=”<<f;
}
}
void main(void)
{   factorial s1,s2;
     s1.enterdata (void);
     s2.calculation (void);
      }
```

Program developed by programmer-II

```
// Include the supporting file
#include <iostream.h>
#include <conio.h>
// Define the class
class factorial
{
private:
int i,f;
int n;
```



```
public:
void enterdata (void);
void calculation (void);
}
// Define the class member functions
void factorial: : enterdata(void)
{
cout<<" Please Enter the Number";
cin>>n;
}
// Do the calculation
void factorial : :calculation (void)
{
// check the number whether it is positive or nega-
tive
if (n<0)
{
cout<<"factorial is not possible";
}
elseif (n==1)
{
f=1
cout<<"factorial of given no. is="<<f;
}
else
{
f=1;
for (i=1; i<n; i++)
{
f=f*i;
}
cout<<"factorial of given no. is="<<f;
}
}
void main (void)
{
// Create the object of the class
factorial m1,m2;
clrscr();
// Calls the class members functions
m1.enterdata (void);
m2.calculation (void);
}
```

