Recruiting Quality Software Developers

Ayman Hussein Odeh
College of Engineering & Information Technology,
Al Ain University of Science & Technology,
PO Box 64141, Al Ain, UAE
Tel: +971-3-7024-886 E-mail: ayman.odeh@aau.ac.ae

Tariq Rahim Soomro
College of Engineering & Information Technology,
Al Ain University of Science & Technology,
PO Box 64141, Al Ain, UAE
Tel: +971-3-7024-883 E-mail: tariq.soomro@aau.ac.ae

Abstract

Developing quality software is the main concerned of software houses and organizations as well as for clients who are demanding software applications for their business needs. Same way it is also concerned of software houses and organizations to recruit highly qualified software developers for their teams. This paper will introduce "System for Measuring Source Code Quality Assurance (SMSCQA)", which will enable the software houses and organizations to recruit best qualified software developers. Experimental results based on five programmers programs will be analyzed and finally conclusion of these experimental results along with future work will be discussed.

Keywords: Software Quality, Recruiting, Software Developer, SMSCQA.

1. Introduction

Development of computer software has been growing over the last few decades (Kriahnan et al., 2000). On the one hand there are several researcher who dedicated their research efforts to improve the quality of software using standard model based on ISO/IEC 9126 (Odeh, 2012) (Baggen et al., 2012) (Motoei, 1996) (Mitra, 2008). On the other hand software development houses and organizations are striving and concerned to recruit the best quality talent, the one who help them to produce high quality software and understands the quality related attributes. Currently software development houses and organizations are facing great challenge, when they wanted to find and choose the suitable candidate for their software development team. According to (Hamilton et al., 2001) hiring a software developer is just like a buying a car; one has to look for a right place and understands what he/she wanted exactly and then buy his/her dream car; but according to the authors buying a car could be easier than recruiting a quality software developer. There are lots of places / experts, one can take car to them and get certified how this car is? – Quality-wise of course, but what about quality software developers? Mostly software developers are recruited by software houses and organizations based on their academic qualifications and professional experiences, but how to determine their academic qualification and professional experience is enough to develop quality software? In this case, if the software developers didn't meet the criteria, software houses and organizations are providing them training to overcome this deficiency. Training here means investment of time, resources and cost. Second mostly software developers are recruited through some agencies or consultant companies. Again for these agencies and companies, same question arises, how to make sure these software developers are going to produce quality software? Again these agencies and companies are spending their time, resources and cost to train these software developers before handing over them to software houses and organizations. There are other means too exist to recruit, for example, personal referral; add in newspaper; campus recruiting; etc. (Hamilton et al., 2001), but again all these means may not guarantee that the recruited software developer is quality-wise best talent. The only way is to test the prospect candidate by giving them coding tests and there is great need of experts who can manually evaluate the code and finalize the recruiting process among several prospect candidates. Manually evaluation is possible if the candidates are few and the particular field expertise is available. In case the particular field expertise is not available or the candidates are several in numbers the manually evaluation process will be impossible humanly. This research paper proposed the solution of this problem by introducing a mechanism to evaluate prospects candidate through the SMSCQA – System for Measuring Source Code Quality Assurance. The section 2 will explore the proposed system; section 3 will discuss experimental results; finally section 4 will conclude and will discuss future work.

2. Proposed System

The main purpose of the SMSCQA (Odeh, 2012) is to measure the quality of source code based of 9 quality factors and more than 30 source code metrics to represent single value. These 9 quality factors are:

- Maintainability
- Portability
- Reliability
- Reusability
- Audit-ability
- Readability index
- Understandability
- Simplicity
- Testability

The architecture of SMSCQA is based on following 6 components and is shown in figure-1 below (Odeh, 2012):

i. Source Code Reader: the job of this component is to read and load the source code file, which contained tested source code
ii. Source Code Analyzer: the job of this component is to analyze and classify source code lines into separated sub-components corresponding to each code structure
iii. Source Code Metrics Measurement: the job of this component is to use the measure all directly measurable source code metrics, for example, LOC, operators, operands, total LOC, etc.

iv. Quality Factors Measurement: the job of this component is to calculate and retrieve non-directly measurable metrics, grouping them to achieve 9 quality factors and also to calculate a final overall quality result

v. Quality Report Generator: the job of this component is to generate four quality reports, these reports may be used to evaluate prospect candidate for recruiting

vi. Common Quality Standard Database: the job of this component is to keep most common quality standards as the basis for this whole process

The result of this mechanism is evaluated in the form of value of weight and this should be controlled properly depending on the nature of the problem. Different weighted matrix should be used for different types of problems, for example, artificial intelligence software evaluation; business software evaluation; embedded software evaluation; financial software evaluation; military software evaluation; real-time software evaluation; scientific software evaluation; system software evaluation; and others. The weight can be change easily in the system.

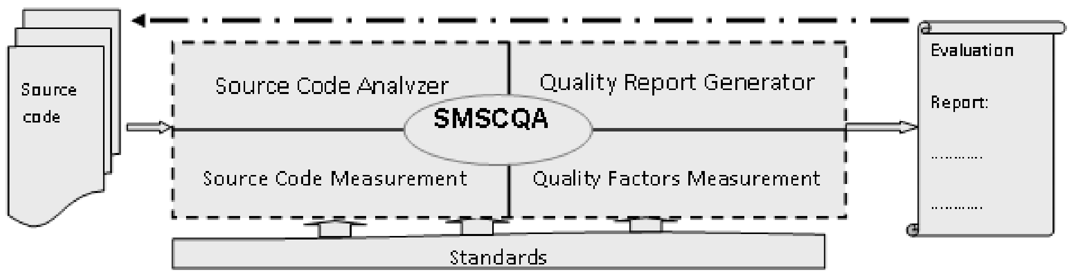

Figure-1: Architecture of SMSCQA

3. Experimental Results

To recruit quality-wise best programmers, organizations need to do some efforts. In this regards SMSCQA is proposed. To find out which programmer is quality-wise better following problem statement was given to 5 equally qualified programmers.

Write a computer program to solve the quadratic equation (second degree) given in the following general form: $ax^2 + bx + c = 0$; find roots in real domain, if there are no solution print out "No solution", finally calculate the multiplication of root1 and root2, if the result is positive number, test this number if it is a prime number then print out "The number is a prime".

The above problem statement was given to five equally qualified programmers. They solved this problem in different languages (i.e. C/C++, Java and VB). Latter on these all programs were converted to VB, as SMSCQA is currently accepting VB based programs only. These programs were tested using the following test cases as shown in table-1, as testing of software is an important process and test cases surely plays an important role in producing high quality software testing (Wen et al., 1999) (Katayama et al., 1999). Here in Table-1, Test case values of a, b and c variables are provided and Programmers P1 to P5 results for each test case is shown here.

Table-1: Test Case Samples

Test Case			Solutions				Programs				
Constants			Roots		No Solution	Is(x1*x2) a prime?	P1	P2	P3	P4	P5
a	b	c	x1	x2							
1	5	4	-4	-1		No	√	√	√	√	√
1	4	4	-2	-2		No	√	√	√	√	√
-1	4	4	4.2	-0.82		No	√	√	√	√	√
-1	-4	-4	-2	-2		No	√	√	√	√	√
5	5	5			No Solution		√	√	√	√	√
5	4	1			No Solution		√	√	√	√	√
1	5	1					√	√	√	√	√
2	4	2	-1			Yes	√	√	√	√	√
0	-2	5	2.5			Yes	X	√	X	√	√
5	0	2			No Solution		√	√	√	√	√
0	0	0			No Solution		X	√	X	√	√

At the next step all programs were tested by the source code quality assurance tool SMSCQA (Odeh, 2012) and result is shown in Table-2. Total 12 software metrics were used for this type of evaluation. Note that in SMSCQA, one can set software metrics other than these, keeping in view type of program/test organization wanted to analyze. Table-3 below shows software quality based on 24 criteria and five programmers performance in this table. Table-4 shows below the SMSCQA 9 quality factors and performance of these five programmers are shown in this table.

Table-2: Source Code Metrics

Source Code Metrics		Program No				
		P1	P2	P3	P4	P5
1	Total Lines	50	43	47	75	35
2	LOC	42	37	39	47	28
3	Comments	3	3	3	22	6
4	Spaces Lines	5	3	5	6	1
5	Variable	3	3	3	9	6

6	Operators	31	28	30	31	17
7	Unique Operators	12	9	10	10	8
8	Operands	63	57	60	68	38
9	Unique Operands	37	31	32	30	21
10	Vocabulary	49	40	42	30	19
11	Length	94	85	90	99	55
12	No Of functions	3	2	2	2	2

Table-3: Software Quality Criteria

No	Software Quality Criteria	Program No				
		P1	P2	P3	P4	P5
1	Halsead's Effort	457	942.5	1210	1692.5	247
2	Average Cyclomatic Complexity	2.5	4	4	4	3
3	Average LOC per module	11	18.5	19.5	23.5	11
4	Percentage comments	0.047	0.066	0.52	0.301	0.185
5	Using Error Handler	0	0	0	0.5	1
6	Software independence	0.875	0.875	0.875	0.875	0.875
7	Hardware independence	0.5	0.5	0.5	0.5	0.5
8	Commented Lines (%)	0.047	0.066	0.052	0.301	0.16
9	Conciseness	0.793	0.822	0.789	0.578	0.642
10	Inline commented per LOC	0	0	0	0.0622	0.138
11	Length of Identifiers	0	0	0	1	1
12	Structural Fan - In	1	1	1	1	1
13	Average Structural Fan - Out	0.25	0.5	0.5	0.5	0.66
14	Internal Documentation	0.084	0.112	0.093	0.513	0.375
15	Degree of Coupling	0.25	0.5	0.25	0.5	0.461
16	C to standard length	1	1	1	0.5	1
17	Variable name convention	0	0	0	0.777	1
18	Control name convention	0	0	0	1	1
19	Goto Usage	0	1	0	0	0
20	Nested conditions	0.416	0.5	0.666	0.333	0.333
21	Dead variables	0	0	0	0	0
22	Dead procedures	0	0	0	0	0
23	Level of abstraction	0.347	0.276	0.262	0.245	0.864
24	accuracy	1	1	1	1	1

Table-4: Software Quality Factors

Software Quality Factors		Program No				
		P1	P2	P3	P4	P5
1	Maintainability Index	0.746	0.699	0.6776	0.7716	0.8410
2	Reliability Index	0.952	0.952	0.95233	0.956	0.997
3	Portability	0.6875	0.6875	0.6875	0.6875	0.6875
4	Understandability	0.3655	0.4375	0.398	0.59	0.6625
5	Reusability	0.491	0.5974	0.5436	0/6776	0.8841
6	Readability	0.449	0.5165	0.513	0.7081	0.7590
7	Audit-ability	0.25	0.25	0.25	0.7592	1
8	Simplicity	0.9351	0.624	0.912	0.995	0.995
9	Testability	0.6756	0.264	0.6374	0.6220	0.8565
	Overall Quality	0.617	0.5589	0.7659	0.8478	**0.9153**

4. Conclusion and Future Work

Right software for right purpose at right cost on right time by right developer is the need of all stakeholders who really cares about software quality. Right software for right purpose can be achieved by collecting software requirements in the form of scope. Right cost on right time also can be achieved by applying proper software management principles and guidelines. Right developer is the only issue left behind and it is the concerned of software houses and software companies along with project managers. This paper considered this issue as challenge and proposed a mechanism to recruit high quality qualified software developer. In this regards, 5 equally qualified programmers were tested and given a task to develop software. The SMSCQA produces the result that programmer number 5 is suitable and he/she understands the importance of quality and while he/she wrote code he/she keep in mind quality concerned. The only limitation of SMSCQA is receiving VB code as an input, but in future authors may update a component, which may accept input source code as C/C++/Java etc.

References

Baggen Robert, Jose Pedro Correia, Katrin Schill, Joost Visser, 2012, Standardized Code quality benchmarking for improving software maintainability, Software Quality Journal, June 2012, Volume 20, Issue 2, pp 287-307, DOI: 10.1007/s11219-011-9144-9, http://link.springer.com/article/10.1007%2Fs11219-011-9144-9?LI=true

Hamilton Marc and Harris Kern, 2001, Recruiting the Best Talent for Software Development Team, URL: http://www.informit.com/articles/article.aspx?p=23770

Katayama Tetsuro, Eisuke Itoh, Zengo Furukawa, Kazuo Ushijima, 1999, Test-case Generation for

Concurrent Programs with the Testing Criteria using Interaction sequences, Sixth Asia Pacific Software Engineering Conference (APSEC '99), http://earth.cs.miyazaki-u.ac.jp/~kat/papers/pdf/apsec99.pdf

Kriahnan, M.S., C.H. Kriebel, Sunder Kekre and tridas Mukhopadhyay, 2000, An Empirical Analysis of Productivity and Quality in Software Products, Management Science, June 2000, Vol. 46, No. 6, DOI: 10.1287/mnsc.46.6.745.11941

Odeh Ayman Hussein, 2012, SMSCQA: System for Measuring Source Code Quality Assurance, International Journal of Computer Applications, Volume 60, No.8, December 2012, http://research.ijcaonline.org/volume60/number8/pxc3884181.pdf

Mitra Amitava, 2008, Fundamentals of Quality Control and Improvement, 3/Edition, ISBN: 978-0-470-22653-7, Wiley 2008

Motoei Azuma, 1996, Software products evaluation system: quality models, metrics and processes – International Standards and Japanese practice, Information and Software Technology, Volume 38, Issue 3, March 1996, Pages 145-154, http://www.sciencedirect.com/science/article/pii/0950584995010696

Wen C. Pai, Chun-Chia Wang and Ding-Rong Jiang, 1999, Selecting Software Testing Criterion based on Complexity Measurement, Tamkang Journal of Science and Engineering, Vol. 2, No. 1, pp. 23-28 (1999), http://www2.tku.edu.tw/~tkjse/2-1/2-1-3.pdf

Tables

Figures