Int. j. inf. tecnol. (March 2022) 14(2):1041–1050
https://doi.org/10.1007/s41870-019-00293-y

ORIGINAL RESEARCH

Open source software: analysis of available reliability models keeping security in the forefront

Shiva Tyagi[1] · **Devendra Kumar**[2] · **Sachin Kumar**[1]

Received: 22 April 2018 / Accepted: 4 February 2019 / Published online: 14 February 2019

Abstract The relatively new Open Source Software (OSS) concept has brought about a tremendous stir in the arena of software development and related fields. But one cannot deny the fact that any kind of revolution demands multiple level of checks and balances to make sure things do not go out of hand. All the utilities and the pluses or the advantages of the OSS might get nullified in case its reliability estimation being not carried out in an appropriate and required manner. Presently, there are a large number of models available for the purpose, proposed by esteemed authors the world over. An effort has been made to review maximum possible and compare their strengths and limitations keeping them side by side. The focus of different models/methods has been different. Some depend upon aspects like maturity, durability, and strategy of the firm. While there are few, that relies on functional aspects of the assessment process. The comparison of the different methodologies may be made in different ways with different attributes in focus. This paper shall try and simplify the task of the users in choosing the right model with right approach for the specific required Open Source Software. The comparison has been compiled and put in the form of a table to make the understanding simpler and future scope has been added in the end to invite suggestions, recommendations and further studies on the subject.

Keywords Models review · OSS · OSS evaluation methods · Reliability testing · Software reliability modeling

1 Introduction

Out of the wide array of functional and non functional requirements of software, reliability happens to be the most important of all. It is a herculean task to precisely estimate or gauge how reliable a Component Based Software Systems (CBSSs) is. In this, not all components contribute to summing up the reliability. The number of times any component is used is different and hence the reliability changes from one component to another. Different approaches have been proposed by a large number of authors, the World over, to estimate and focus on the CBSSs and its reliability. The various approaches cover a wide spectrum ranging from focus on reliability of component to laying more and more emphasis on glue code reliability. While there are a large number of approaches that involve mathematical solution, it cannot be ignored that the basic fact stating that the reliability phenomena involves real world and as a matter of fact it actually is a function of real time life like problems and their solutions on day to day basis [1, 2]. Soft computing techniques can get us the solution to those problems, the answer to which is both unpredictable and uncertain [3, 4]. These techniques take the details from the past and capture the ongoing pattern in the form of data. When the data available with us is enormous, and the new data sources are available, the major challenge faced in today's organizations is to establish a method wherein decision making and

✉ Shiva Tyagi
tyagishiva1@gmail.com

Devendra Kumar
devmochan@yahoo.co.in

Sachin Kumar
imsachingupta@rediffmail.com

[1] Ajay Kumar Garg Engineering College, Ghaziabad, UP, India

[2] College of Engineering, Roorkee, Uttarakhand, India

organizational processes are converted into data that can be utilized to take better decisions.

The reliability estimation is all the more required to be extended to OSS, which is a new paradigm to develop the software by a community and not by an individual or a specific team of developers. The very idea of OSS based products/projects have revolutionized the field of software development. This has taken a new dimension since the commercialization of the software industry has taken place [5, 6]. Keeping the quality utmost, while choosing amongst the various open source products, is the safest option. To validate the software quality, quality models are put to use. This comparison will be done based on certain selected attributes.

2 Software requirements

As a software user, generally there are two major requirements. They are reliability and availability. They are the basic requirements for software for a layman. The importance of reliability and availability will be different for different circumstances. Reliability is of utmost importance when the non performance in respect of the software has higher stakes. Similarly, availability will be of prime importance to the user when the product downtime will be of greatest importance [7]. So it is seen that reliability and availability are both equally crucial and it is a matter of the circumstances that which one takes the back seat and which one places itself in the driver's seat.

Reliability is extremely difficult to define. Reliability is a function depending on time and availability both [8]. However, if someone chooses to define it mathematically, it can simply be represented in a crude and simplistic manner as under:

$$\text{Reliability} = \{1 - \text{Probability of Failure}\}.$$

So, in simple terms, if a specified environment is made available, and a limited and fixed amount of time is decided to analyse, the reliability in respect of software may be defined as the probable outputs of failure free performance. The probability of failure needs to be worked on and thereafter if we subtract it from unity, we shall be able to arrive on the figure of reliability. In a given environment and in specific given conditions, the calculation of reliability will be done simply based on the chances of the software crashing. The percentage of the reliability may simply be calculated based on the above procedure and the probability related data. Software applications are becoming complex by the day and this leads to the importance being gained by the reusability factor. The importance being given to the reusability forms the origin of Component Based Software System (CBSS) based applications. Another thing that is very close to it is the Component Based Software Engineering (CBSE). It is the branch of engineering which is dependent on the software reuse. Putting it simply, it concerns the software developing from the Commercially Off The Shelf (COTS) components [9]. The components in it are assembled in a manner which is interoperable. To achieve the desired level of reliability is an extremely difficult task even with pretested, high quality software components with a high degree of trust. So, to elaborate the reliability factor in a component based application in a detailed and thorough manner, there exist various methods which have overall been categorised into the under mentioned groups:

- System level reliability estimation.
- Component based reliability estimation.

In the system level estimation, we calculate reliability for the whole system. But in case of component based estimation, the application reliability is calculated depending on the function of reliability of separate components and even to the extent of their interconnection matrix. Among these, the fuzzy inference system (FIS) and the adaptive neuro fuzzy inference system (ANFIS) proposed in various research papers have been able to solve the problem of reliability estimation to a great extent. The ANFIS has a marked advantage over the FIS as it combines fuzzy logic with the capabilities to learn of various neural networks [10].

3 Various models used for quality evaluation in OSS

Though this paper will discuss the concept of reliability estimation in detail with respect to various proposals made by various authors in different journals and conferences, the focus of the authors of this paper will be to study and analyse the various models used and proposed world over in the field of evaluation of quality factor in OSS. Open source projects targeting similar type of applications are very common these days. So to decide which one to choose is a tricky option. Open source quality estimation models have gained a wide acceptance because the earlier models or the traditional models of estimation of quality, have failed miserably to provide an exact estimation of certain specific and unique features in terms of OSS. The authors of this paper will focus on comparing the signature characteristic features along with the strengths and shortcomings of various available models proposed till now.

In this age of OSS, we find the OSS all around us and are being utilised and worked on at a day to day basis. They are around us in the form of operating systems (FreeBSD, Linux, Solaris), middleware/database technologies like

MySQL and Apache Web server, and even as web browsers like Mozilla Firefox. The list is vast and the applications are all over. Most of the OSS are of very high quality as viewed by program designers, code writers and even the end users. As per the definition given by the Institute of Electrical and Electronics Engineers (IEEE), the quality is defined as the level to which a system along with a process or a component caters for the user satisfaction. The quality in terms of software is measured by different quality models.

There exists two major ways in which one can work out the quality factor in software products. The quality model forms the main approach that analyses and assesses the standard of quality for software. In this, a specific set of attributes/metrics are worked on separately to optimally find out the quality by making it quantifiable or through a mathematical concept. But such models miss out on covering OSS at times. For this reason, quality models for OSS are put to use and there has been a lot of work/study that has gone into comparing the different available quality models [11]. The available quality models are compared in terms of unique strengths and limitations and characteristic features in this paper. The base paper taken for this study is the paper authored by Misra, Adewumi, and Omoregbe. The authors of this paper shall try and work up on the available study of the different OSS models as proposed by the authors above. Different models proposed since the time of evolution of open source product methodology are Cap Gemini Open Source maturity model, QSOS, Open BRR, SQO-OSS, OMM, QualOSS, OpenBQR, QualiPSO, MOSST, OITOS, FOCSE. In the subsequent sections of this paper, a detailed review of all the above collated models will be carried out and the best and ideal option out of the lot will be arrived at.

4 Methodology implemented for analysis of various models

The strength and weaknesses of all different models for estimation of reliability proposed since the beginning of OSS concept have been thoroughly studied by the authors and co-authors of this very paper. A large number of research papers were studied for understanding and analysing the different features and for understanding the scope of different models available online. A vast array of publications were retrieved to examine each model closely in order to identify their features. Based on the study, the results were tabulated and the comparison carried out. The tabular comparison of various models is given in the succeeding sections of this paper [1, 12]. The comparison matrix has been separately prepared for the readers to see for themselves the differences in features and has also been given subsequently in this paper. In one of the sections of the paper, ISO 9126 base model for all the models have been studied in detail giving all the features, limitations and scope to enable the readers to analyse for themselves the progress and developments in various proposed models since the time of evolution of OSS.

5 The positives and negatives of various models

As mentioned earlier, the base paper referred during writing of this paper was studied in detail and to build up on the recommendations and suggestions in the given paper, the authors have gone into each and every detail that could be extracted from the papers and from different websites on the internet [13, 14]. The table and its format has been picked up from the base paper authored by Adewumi et al. [1, 12]. Initial few models and their studies have been ascertained by the authors and after due verification, certain readings in the following table has been utilised. Balance of the models, which had not been proposed by the time the above mentioned paper was written, have been reviewed by the authors of this paper. The detailed analysis of the features and the shortcomings of different models that have been proposed since the evolution of this concept are as follows.

5.1 Cap gemini open source maturity model

This model was proposed in year 2003. Its strength lies in considering the application indicators and products as part of it. Also, regular updation can be carried out in it with the help of feedback from the users. If the limitations have to be highlighted, then one can say that it is a practical assessment model that deals with two axes on two levels.

5.2 QSOS

Suggested in year 2004, this model consists of four iterative stages. Those stages are definition and evaluation followed by selection and qualification. It is supported by a special tools termed as O3S. Allowing objectivity and enabling traceable evaluation of OSS is the main feature of this model that gets listed as its strength. Discussing the shortcomings, the recent documentation such as version 1.7 is expected to be translated to English for enabling wider usage, making is a tedious process.

5.3 Open BRR

The model was proposed in year 2005. Discussing the special features and strengths of this particular model, it enables accelerated software evaluation and this process is

carried out in a systematic manner. The decisive capability of the user is improved and the level of confidence in the selected OSS increases many folds in this case. This particular model is open and it is customizable and is easily applicable in any business situation. Mentioning about the limitations and shortcomings, the website of the model has not shown any improvements for a long time. It is a one page website with just a few links given to any noteworthy resource materials. The model has not solved the purpose of suggesting a vendor-free and federated clearing house of important data on OSS packages.

5.4 SQO–OSS

This is a model that has a form of hierarchy in its basic design and the evaluation of source code and community process is pretty efficient in it. All the metric values are calculated on its own and it even offers the synergy and the inter relation of the metric values to the predefined quality profiles. Its design reduces user interaction thereby leading to reduction of the subjectivity. This particular model offers a base for the formation of the new matrices. Coming on to the limitations, the main and only limitation that this model offers is the limited user interaction. For evaluation purpose, the user interaction needs to be enhanced to a certain level.

5.5 Open source maturity model

It offers tree level scale and simplicity to the OSS and its tools are available for the evaluation purpose. But this model has not been industrially validated. Once the industrial validation is over, necessary feedback will be gathered and this model will be extensively put to use.

5.6 QualOSS

The robustness of this model is its strength. Apart from this, the factor of evolvability adds another feather in its cap. The quality measurement is also automated in this model which in turn leads to reduction in subjectivity. This may be termed as its limitation as due to reduced subjectivity, there is more dependence on quality.

5.7 OpenBQR

It is the open business quality rating method and a form based web application which involves tool support. This assessment model is quite practical as it involves three levels of details and a flexible scoring model. The only shortcoming of this model is that it does not provide complete and a balanced evaluation.

5.8 QualiPSO

It supports the complete evaluation in a balanced manner of the software, addressing technical, economic, customer and growth/evolution issues. But the issues with this model are that it relies heavily on the product documentation. It is pretty tedious to implement. Code quality is also not tested effectively by this model of software reliability. Only a few sub qualities are present.

5.9 ISO 9126

All the existing models find their base in this model. Over a period of time they have all evolved from it. A total of 6 different quality characteristics are defined, both internal as well as external. They are the ones starting from functionality, moving onto reliability, covering up the performance, catering for the efficiency, and looking after the maintainability and portability aspects. As mentioned earlier, it is the basic model and it is this factor only that is its shortcoming. Being basic, it does not consider certain quality attributes specific to OSS.

5.10 ISO 25010

The key features of ISO 25010 are all derived from ISO 9126. They are just add-ons to the ones mentioned in ISO 9126 above. Those features include functionality, suitability, reliability, efficiency, performance, operability, security, maintainability and transferability. Mentioning about the limitations of this model, this can be utilised as the standard model for OSS only with the factors considering product quality and quality in use. The unique characteristic of OSS like community are not addressed in it.

5.11 E-OSS

Popularly termed as the easiest OSS model, it came into existence in year 2015. The best part about this model is that it offers adaptability for small and medium business and offers interoperability. The software is yet to be exploited thoroughly so the limitations are still being worked out.

6 Base model for all the existing models

The ISO 9126 model is the base model based on whom various latest models for quality estimation in the present day are based. This model was proposed with six characteristics or factors and all these are subdivided into sub characteristics [15]. The characteristics are manifested

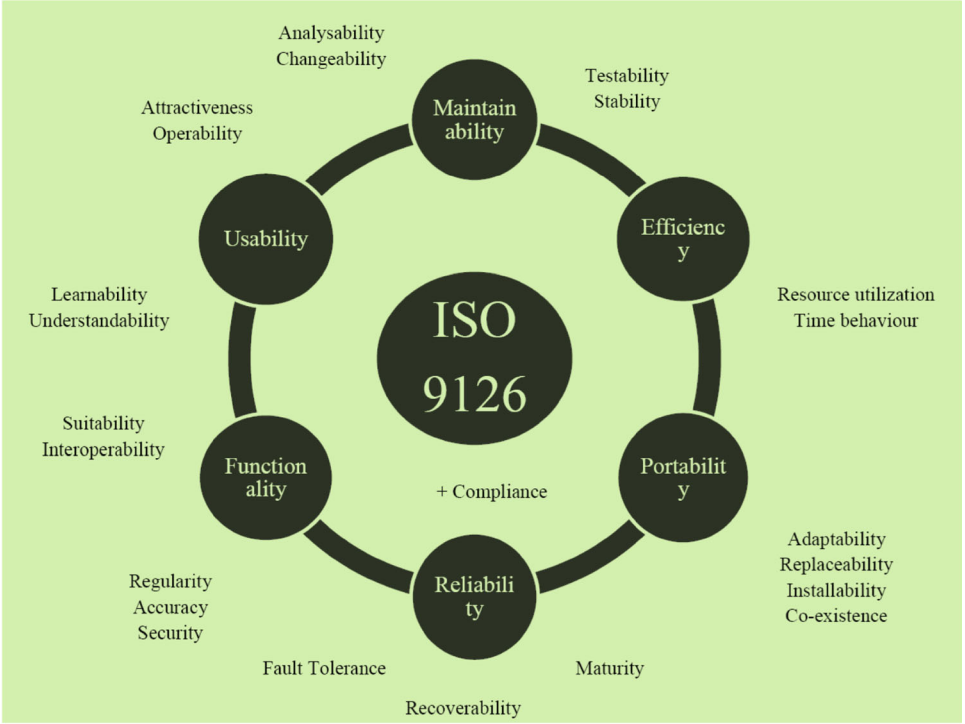

Fig. 1 The ISO 9126 model

externally when the software is used as a consequence of internal software attributes. The various characteristics and sub characteristics that complete and explain ISO 9126 are as given in the subsequent paragraphs in Fig. 1 [15] and Fig. 2 [16].

6.1 Functionality

Simplest definition says that it is a set of those attributes that have direct effect on the basic existence of the set of functions and their thorough specific attributes covered in detail. The different functions that are being covered are those that satisfy the implied needs.

Accuracy, suitability, security, compliance, interoperability.

6.2 Reliability

The keyword here is the assurance or the promise of a desired specific performance standard. It is the group of those attributes that bears direct effect on the performance of software and its ability to maintain it in the given conditions for a specific, accurate and desired period of time.

Fault tolerance, maturity, compliance, recoverability, reliability.

6.3 Usability

In this, the set of attributes involved are those that has effect on the effort required for making full use of the software or in other words exploiting it fully and on the individual study of such use, by a specific or implied group of users.

Learnability, understandability, operability, usability compliance, attractiveness.

6.4 Efficiency

A set of qualities that has a bang on impact on the interrelationship between the level of performance of the software and the specific amount of resources used, under a given set of stated conditions.

Efficiency compliance, time behavior, resource utilization.

6.5 Maintainability

It involves a list of those attributes that has a direct bearing on the efforts needed to make required modifications.

Changeability, analyzability, maintainability compliance, stability, testability.

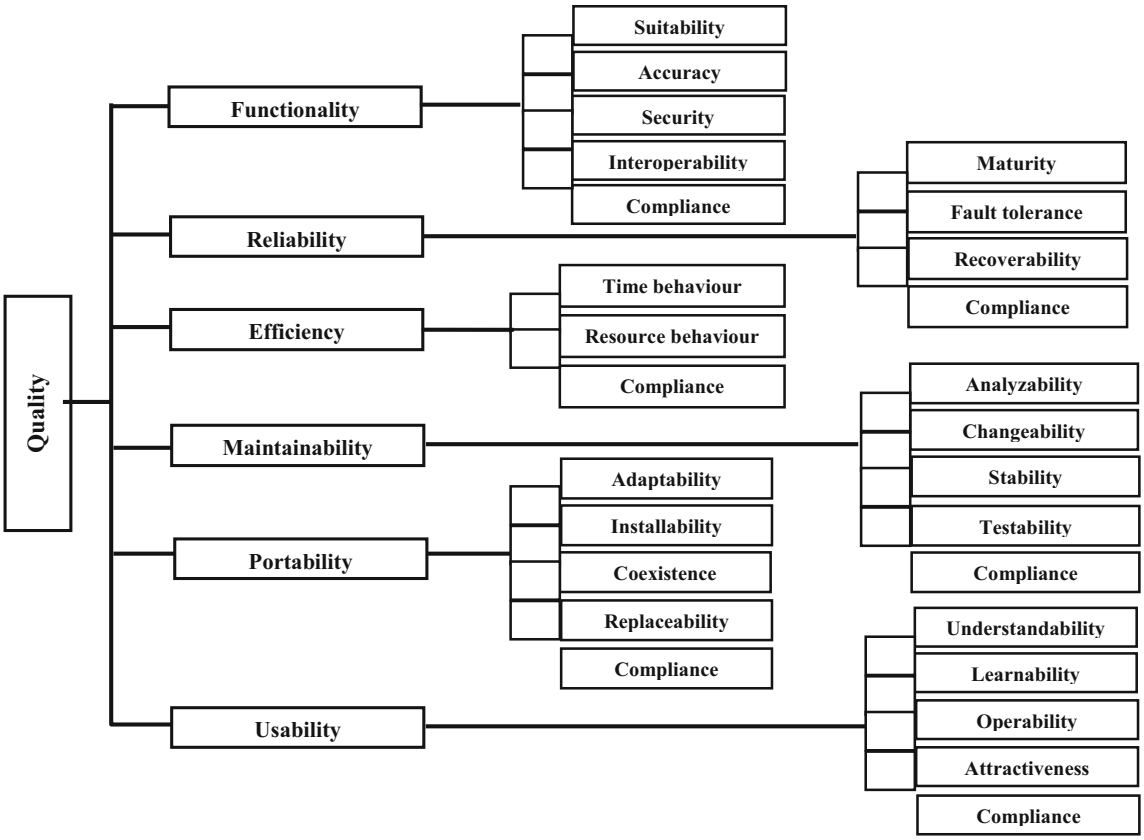

Fig. 2 The simplified ISO 9126 quality model

6.6 Portability

As the name suggests, it is a set of those characteristics that permits the software to be shifted or transferred from one working environment to another.

Installability, adaptability, portability compliance, co existence, replaceability.

The origin of the base model, i.e. ISO 9126 is from the ISO 9000 standard which is the most important standard for quality assurance. The same has also evolved from a series of other standards that were prevalent prior to it coming into use and being the base model for all the subsequent models. The initial models that were existing prior to ISO 9126 were McCall Model, Boehm Model, FURPS Model, Dromey Model, BBN Model and Star Model. A brief comparison between the models proposed prior to the ISO 9126 model has been carried out in Table 1 [12, 17]. The shortcomings of all these models were accounted for in the ISO 9126 Model thereby leading it to be the base model for all the models that were proposed after it.

7 Comparing the quality models

Based on the features and limitations given in Sect. 6 above, the comparison has been carried out between all the models and the same has been tabulated [1] as given in Table 2. A simple comparison was carried out between all the models keeping the basic requirement of ease of understanding in mind. To make it simpler, the basic questions were considered. The following were the criteria that were considered for the purpose of comparing the different models:

- If the published results were available online or not?
- Origin of the model
- If the tool support is available or not?

Comparative study of all the quality models is as given in Table 1. The results compiled and tabulated shows different features and operating requirements of the various models. It depends entirely on the user as to which model is chosen by the user depending on the extent of usage and the required quality of the OSS for the user. The basic and the simplistic ones may be utilised for carrying out the quality modeling in cases where complex details are not involved. However, the latest and the advanced versions of the quality models may be involved wherein OSS has got

Table 1 Comparison of different available software quality models proposed prior to ISO 9126

Quality model	Structure	Number of levels	Relationship	Disadvantages	Advantages
McCall	Hierarchical	Two	Many to many	Overlapping of components	Having evaluation criteria
Boehm	Hierarchical	Three	Many to many	Lack of criteria	Including factors related to hardware
FURPS	Hierarchical	Two	One to many	Not considering portability	Separating functional and non-functional requirements
Dromey	Hierarchical	Two	One to many	Incomprehensiveness	Applicable to different systems
ISO	Hierarchical	Three	One to many	Generality	Having evaluation criteria Separating internal and external quality
Star	Non-hierarchical	–	Many to many	Lack of criteria	Considering different viewpoints
BBN	Non-hierarchical	–	Many to many	Lack of criteria	Having weighted quality factors

Table 2 Comparison of quality models

Criteria model	Is the published result available online?	What is the origin of the model	Is the tool support available?
Cap Gemini Open Source Maturity Model	Yes	ISO/IEC 9126 quality model	No
QSOS	Yes	ISO/IEC 9126 quality model	Yes
OpenBRR	Yes	ISO/IEC 9126 quality model; Cap Gemini Open Source Maturity Model; Navica Open Source Maturity Model	No
SQO-OSS	Yes	ISO/IEC 9126	Yes
OMM	Yes	Capability Maturity Model	Yes
QualOSS	Yes	Cap Gemini Open Source Maturity Model; QSOS; OpenBRR	Yes
OpenBQR	Yes	QSOS; OpenBRR	Yes
QualiPSO	Yes	ISO/IEC 9126; OpenBQR; OpenBRR	Yes
ISO 9126	Yes	Basic Model	No
ISO 25010	Yes	ISO 9126	No

the complexity level which is not manageable with the earlier versions [12]. Also, wherever the stakes are higher for the user, he has the option to choose from the variety of the above mentioned models for the specific OSS he is dealing with [17].

8 Attributes of OSS with respect to various quality models

The authors of this paper have elaborated on different attributes of different quality models that have been discussed so far in this paper. The quality models have been elaborated in two different time frames. Those that have been proposed before the basic ISO 9126 model have been covered separately and those that have evolved from the ISO 9126 have been covered with the different setup in this paper. In this section of the paper, the authors will try and give out the attributes of Open Source Software for various quality models. The same are being highlighted as below.

8.1 McCall

Efficiency, flexibility, interoperability, maintainability, reliability, usability.

8.2 Boehm

Efficiency, portability, reliability, testability, understandability.

8.3 FURPS

Adaptability, functionality, performance, reliability, supportability, usability.

8.4 Dromey

Efficiency, functionality, maintainability, portability, reliability, reusability, usability.

8.5 ISO 9126

Accuracy, changeability, efficiency, functionality, installability, maintainability, portability, reliability, stability, suitability, usability.

8.6 ISO 25010

Accuracy, adaptability, changeability, efficiency, functionality, integrity, installability, interoperability, maintainability, modifiability, portability, reliability, reusability, stability, suitability, transferability, usability.

8.7 Cap gemini open source maturity model

Accuracy, efficiency, functionality, maintainability, maturity, reliability, usability.

8.8 QSOS

Adoption, ease of use, ergonomics, exploitability, intrinsic durability, maturity, technical adaptability.

8.9 Open BRR

Accuracy, changeability, efficiency, installability, maintainability, reliability, stability, suitability.

8.10 SQO-OSS

Maintainability, reliability, security, changeability, stability, testability, maturity, effectiveness.

8.11 OMM

Availability, functionality, interoperability, maintainability, security, stability, quality.

8.12 QualOSS

Accuracy, adoption, ease of use, efficiency, exploitability, functionality, maintainability, maturity, reliability, technical adaptability, usability.

8.13 Open BQR

Adaptability, availability, cost, functional adequacy, quality, repeatability, simplicity.

8.14 QualiPSO

Cost Effectiveness, customer satisfaction, developer quality, functionality, interoperability, reliability, security.

9 Comparison in the backdrop of security of information

After having analysed all the security models based on the ease with which they can be referred, the online availability of the published results, their origin from the specific base model and simply the availability of tool support, there is an important need to carry out the analysis based on the security. Security can be ensured in a large number of ways. Cryptography is one such way. Cryptography is the term used when the level of security is very high and any kind of loss of data to any unauthorized personnel is not acceptable at any cost. The codes are encrypted and the access to the key to the encryption is made available to only the authorized individual or the agency. This way, the security of the classified information or the important data can be ensured and one can use the open source software as per one's own will without any fear of the security being compromised. The very concept of the OSS is such that one has limited choice in restricting the availability of the software from anyone to everyone. However, it is still pertinent to discuss the cryptographic security aspect in terms of the different reliability estimation models for OSS that have been discussed earlier in this paper. There are a wide variety of other security related Do's and Don'ts that are a must when one is dealing with OSS. A few of them have been enlisted below.

- A detailed analysis is required in respect of the firm that desires to utilise OSS and the same should be carried out based on the security policy of the firm. If the firm's security policy does not permit the use of OSS, the idea to use OSS should be immediately dropped.
- A repository is required to be created and maintained at a firm's level that may store data pertaining to the OSS utilised and the results therein. This will enable the ease of employing the OSS in future while studying its success rate.
- The data regarding the vulnerability and related information must be mapped and regularly updated in the repository.

- One must also ensure that the OSS, if at all being used in one's firm, is of the latest version and is up to date to ensure all bugs have been dealt with.
- If for some reason, the firm has to use an outdated or abandoned OSS, a process needs to be employed to diminish the risks. At the same time the firm should be prepared to accept the risks likely to occur due to the use of outdated OSS.
- An internal evaluation of the security aspects of the use of OSS should be carried out to ascertain the security risks of the OSS being used in the firm. In case of non availability of the experts at the firm, the professional experts on the domain may be consulted.
- A plan for security assessment for the OSS being used at the moment and the OSS that are likely to be used in near future should be incorporated in projects being taken on by the firm.

10 Conclusion

In this paper, the authors have been able to identify and list out various models that are existing as on date to check the quality since the idea of the OSS came into existence. The differences in terms of various features, strengths, their shortcomings, their base, the availability of online published results and the availability of the tool support have been studied, explored and tabulated to make it fully simpler for the readers to choose the best one for them. A large number of publications that have been referred for this purpose have been listed separately and the authors and editors of all such publication are being rendered with a vote of thanks from the authors of this paper. The origins of different models differ from traditional basic software quality models such as (ISO/IEC 9126), to the mix of traditional and contemporary models, to purely contemporary models and even from capability maturity models. Their original base model determines the list attributes and qualities they possess. This can assist the users to identify the best suited model they need to utilize to make sure they make the right choice in terms of selecting the exact quality models that they desire for the OSS. The availability of the tool support is another factor that can assist the users in making the right decision in terms of selecting the correct quality model which is best suited for them [2]. This gains all the more importance due to the reason that the open source product are gaining importance day by day and people are using them and improving them on daily basis [7]. This leads to the gaining of importance by the quality models and to understand the quality models easily, one has to start right from the base model. The scope of this paper has been limited to comparing the existing models on the topic and no new model has been proposed.

11 Future scope

This paper serves as the roadmap for the research scholars and authors across the World to step into the arena of the quality models of an OSS. Though the subject and the topic is very relevant in modern times, still it is found that there is only limited knowledge available online regarding this particular subject. This paper provides the consolidated list of quality models with all the features that affect the choosing the right one for oneself by the user. The tabulated form of comparison with all the basic tenets is a must for all the research scholars who are keen to pursue this particular topic for their research studies. The authors of this paper wish all of them all the very best for their future endeavors.

References

1. Adewumi A, Misra S, Omoregbe N (2013) A review of models for evaluating quality in open source software. Int Conf Electron Eng Comput Sci 4:88–92
2. Waserman AI, Pal M, Chan C (2006) Business readiness rating for open source software. In: Proceedings of the workshop evaluation in frameworks for OSS
3. Tyagi K, Sharma A (2012) A rule-based approach for estimating the reliability of component-based systems. J Adv Eng Softw 54:24–29
4. Kuwata Y, Takeda K, Miura H (2014) A study on maturity model of open source software community to estimate the quality of products. In: 18th international conference on knowledge-based and intelligent information & engineering systems—KES
5. Park J, Baik J (2015) Improving software reliability prediction through multi criteria based dynamic model selection and combination. J Syst Softw 101:236–244
6. Amin A, Colman A (2013) An approach to software reliability prediction based on time series modelling. J Syst Softw 86(7):1923–1932
7. Dimov A, Punnekkat S (2010) Fuzzy reliability model for component-based software systems. In: 36th EUROMICRO conference on software engineering and advanced applications, Sofia, pp 39–46
8. Fagerholm F (2017) Measuring and tracking quality factors in free and open source software projects. University of Helsinki, Department of Computer Science, Master's Thesis Report
9. Goswami V, Acharya YB (2009) Method for reliability estimation of COTS components based software systems. In: International symposium on software reliability engineering
10. Tyagi S, Kumar D, Kumar S (2017) Understanding the nittigrities of software reliability and its testing procedures: a different approach. J Inf Optim Sci 38(6):971–988
11. Ortega F, Coca P, Cortazar DI (2010) QualOSS: Quality of Open Source Software

12. Adewumi A, Misra S, Omoregbe N, Crawford B, Soto R (2016) A systematic literature review of open source software quality assessment models. SpringerPlus 5(1):1936
13. Del Bianco V, Lavazza L, Morasca S, Taibi D (2009) Quality of open source software: the QualiPSo trustworthiness model. In: IFIP international federation for information processing
14. Jaiswal GP, Giri RN, Sharma D (2015) Comparative analysis on reliability estimation techniques of component based software system. IJCST 3(2):240–246
15. Djouab R, Bari M (2016) An ISO 9126 based quality model for the e-learning system. In: International journal of information and education technology
16. Behkamal B, Akbari MK (2009) Customizing ISO 9126 quality model for evaluation of B2B applications. Inf Softw Technol 51(3):599–609
17. Tyagi K, Sharma A (2014) An adaptive neuro fuzzy model for estimating the reliability of component-based software systems. Appl Comput Inf 10:38–51