Quality management in optimizing software product profile

IOP Conf. Series: Materials Science and Engineering **666** (2019) 012062 doi:10.1088/1757-899X/666/1/012062

Quality management in optimizing software product profile

A. K. Moskalev[1], K. S. Lugovaya[2], M. Y. Kharitonova[3]
[1]Siberian Federal University, Krasnoyarsk,660041, Russia
[2]"Alfateam" Web Studio, Krasnoyarsk,660028, Russia
[3]Institute of Chemistry & Chemical Technology, Federal Research Center Krasnoyarsk Science Center, Siberian Branch of the Russian Academy of Sciences (ICCT SB RAS), Krasnoyarsk, Russia

E-mail: ak_moskalev@mail.ru

Abstract. The article investigates the quality of a software product sold by "Svyazcom Ltd.", a developer in the field of mobile communications and Internet, and created at various times by several programmers. It is shown that the application of the computer analysis permits to verify 86 indicators relevant to the quality parameters in the shell CppDepend, which meets the requirements of the assigned task most adequately. The analysis resulted in an optimized product profile. Proposals are presented for updating the quality management mechanisms in an innovative IT company taking into account the obtained values of quality parameters.

Keywords: quality of software product, profile of the best program, quality system in IT company, updating the matrix of responsibility.

1. Introduction

The market for the modern software industry is characterized by a very high degree of competition, and for a successful work any IT company must develop, introduce and maintain software of the appropriate quality quickly. As a consequence, the company needs to create adequate solutions to improve the quality system management process. Statistics show that the improvement of software development processes in successful companies leads to a significant increase in productivity and quality with an average return on investment of up to 8:1 [1]. In this regard, research and practice show the inevitability of developing objective indicators to assess the company's ability to produce programs with the required quality characteristics, confirmed by a conformity certificate.

The need to create new software solutions is caused by several reasons, one of which is the creation of software in the field of automation of industrial or commercial projects. Software development on an industrial scale is a high-tech process that should provide a unified scheme for the work of a large number of developers and ensure that the system meets the customer's expectations [1-2]. It is known that the quality of software is a combination of properties that characterize the ability of software to meet the user's needs in accordance with the specified purpose [2,3,4]. The problem of detecting and eliminating errors is exacerbated due to the increased complexity of the tasks solved by programmers; in case of failing to detect and prevent errors there may happen fatal accidents in information systems concerning critical objects or processes [5].

2. Materials and methods

Special metrics are used to measure software quality. Software metrics is a measure that allows calculating the numerical value of some property of the software or its specifications [1]. At present, several hundred metrics are used around the world. However, the pursuit of their universality coupled

with ignoring the scope of the software being developed reduces the effectiveness of their use significantly. When choosing metrics, it must be remembered that the metric shall enable tracking the trend of changes and make sense both for the customer and for the performer.

The control of the code by a supervisor other than its developer can be done manually. Evaluation of the code includes the compliance of the code with the requirements for structuring and dividing into modules, the completeness and quality of documents accompanying the code, including the documentation of the headers of software modules, function prototypes and data structures, comments on the implementation of essential operations, the compliance of algorithms in the source code with software documentation. This kind of control is a long and superficial approach, which is unable to calculate the metrics, whereas the use of methods and tools for automated source code analysis simplifies this task.

There are a large number of commercial and free static code analyzers such as CandC ++ CodeCounter, SourceMonitor, Reflector.CodeMetrics, FxCop, CppDepend, ParasoftC / C ++ test, etc. The list of languages, to which static code analyzers are applied, is also quite extensive (C, C ++, C #, Java, Ada, Fortran, Perl, Ruby, ...). Details can be found in the monograph [1] or on the websites of organizations producing the corresponding software product [6-10]. It is important to understand that there is no perfect tool for testing programs; one needs to find the one that suits the specific needs, programming language and assigned tasks best.

Therefore, the purpose of the work, the results of which are presented in this publication, is to select a computer program for automated analysis of the quality of a software product and to update the responsibility matrix in an IT company based on the main metrics.

"Svyazcom Ltd." is a developer of innovative solutions and services in the field of mobile communications and the Internet, as well, it is a developer of VAS platform solutions and specialized software for mobile operators and service providers [11-15].

The company's activities cover two areas:

- development and distribution of mobile content (pictures, music, Java games, videos, themes, etc.);
- development of VAS platform solutions and specialized software for mobile operators and service providers.

Our ABC analysis of major projects of the organization showed that the projects of group A occupy a bigger share among all projects (43%) and account for 79% of all revenues. Group C projects in percentage terms have a small share (20%) compared to the rest, which means that the company has practically no unimportant projects, which make up 5% only. All projects contribute greatly to the company's total revenue. The most profitable and cost-effective project is the Galaxy project. Now the project, according to its creators, has become a mobile social network, while the "Galaxy Dating" project was initially developed with the focus on mobile phone users. Today, the number of registered users of the project has exceeded 10,000,000.

3. Results

To analyze the quality of software products produced by "Svyazcom Ltd." we needed to select the most appropriate code analysis tool. CppDepend was chosen, as it is perfectly suitable for the purpose of the work and is a unique and extensive software tool containing 86 metrics.

The quality assessment of the software code created in "Svyazcom Ltd." was carried out on the basis of three projects written at different times by different programmers, though having the common goal. A set of software code modules was written to implement the "Svyazcom.SmartConnect" platform.

"Svyazcom.SmartConnect" platform provides a range of services to stimulate voice traffic in cases of problems that interfere with the connection of subscribers (subscriber unavailability, subscriber being busy, subscriber having not noticed the call, lack of money on the caller's account, etc.). The "Svyazcom.SmartConnect" platform accepts calls from subscribers and performs their processing in accordance with the parameters of the call forwarding, settings in the profiles both of the calling subscriber and the receiver of the call.

The standard range of services includes:

- MissedCallsAlert (MCA), which sends SMS alerts about missed calls and the availability for call of a previously unavailable subscriber;
- SmartVoiceMail (VM), which gives an opportunity to leave voice messages to an unavailable subscriber, which can be listened to later.

Analysis of software quality over time, i.e. analysis of projects similar in their functions, but made at different times by different programmers, showed that the quality of written program modules improves simplifying their integration into other projects.

The authors made the quality analysis of the program code in one of the main modules of the platform "Svyazkom.Smart Connect", created at different times by several programmers for "Svyazkom Ltd.", the developer in the field of mobile communication and Internet.

A visualized representation of the code metrics is a matrix, which, depending on the type of metric, shows the scale of all the project components in comparison. CppDepend uses TreeMap, a tree-structured data mapping method with the help of nested rectangles. This method is used to display the project hierarchy. When one clicks on a certain area of the matrix, the structure of the project, whether it is a type, method or class (it depends on the choice of display parameters) is illuminated with a different light on the entire matrix, in accordance with this light activation by other structures.

Below are screenshots of the matrices of the three discussed programs analysed with CppDepend

The first project was created four years ago and was unstable. Figure 1 shows the relative volumes of each module and the methods and types of the embedded program, which is indicated by the cellular structure. The comparison was made based on the number of lines.

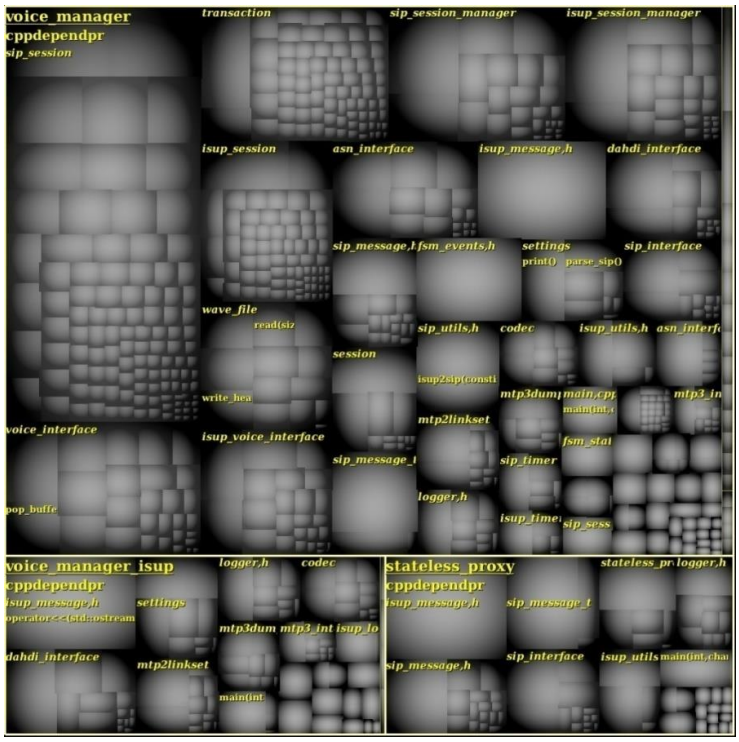

Figure 1. Project 1 matrix.

This matrix clearly demonstrates the absolute and relative volumes of all program structures. The distribution of volumes is homogeneous. It must be borne in mind that too many small structures lead to an added complexity of understanding the program code and increase the likelihood of errors.

In the same way we investigated the second project, which was created two years ago by the same programmer. The project matrix is presented in Figure 2.

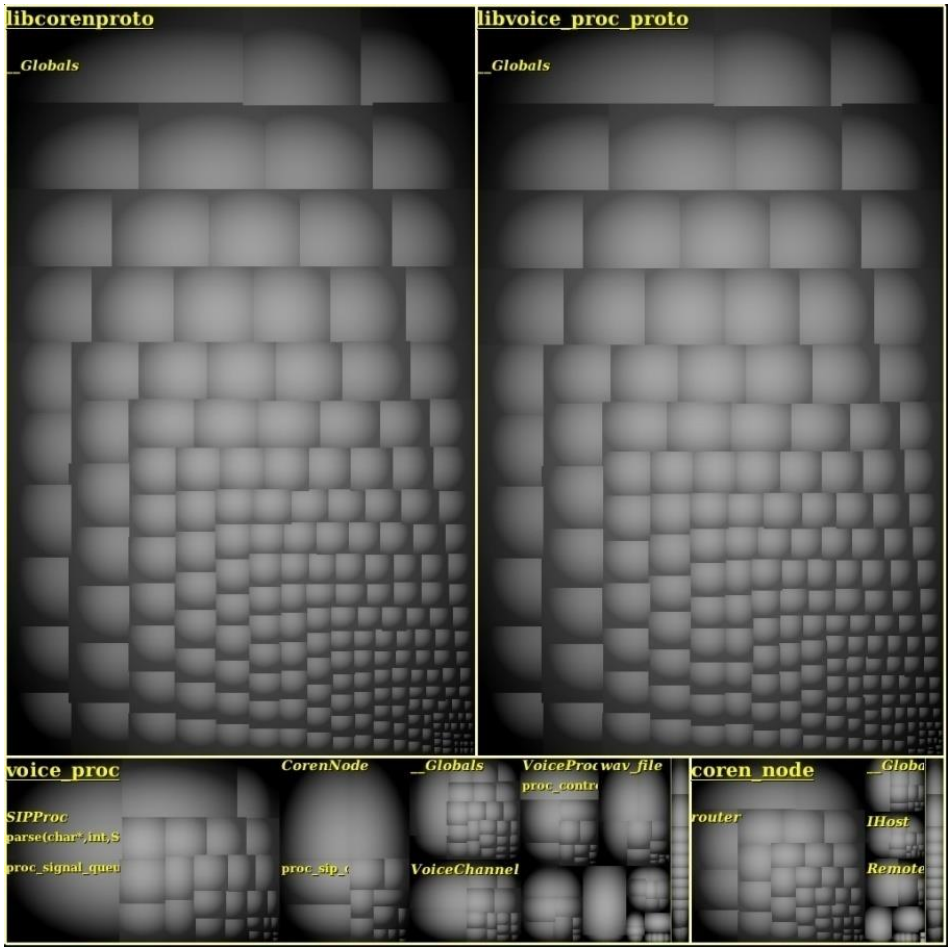

Figure 2. Project 2 matrix.

This matrix demonstrates the absolute and relative volumes of all program structures in a clear form. The distribution of volumes is heterogeneous: libcorenproto and libvoice_proc_prot have much larger volumes than other structures.

Comparing the two projects, we can conclude that the first project consists of a larger number of small structures than the second.

The third project, created six months ago by another programmer, is different in both the quality of the code and the logical structure. The main project metrics are presented in Figure 3.

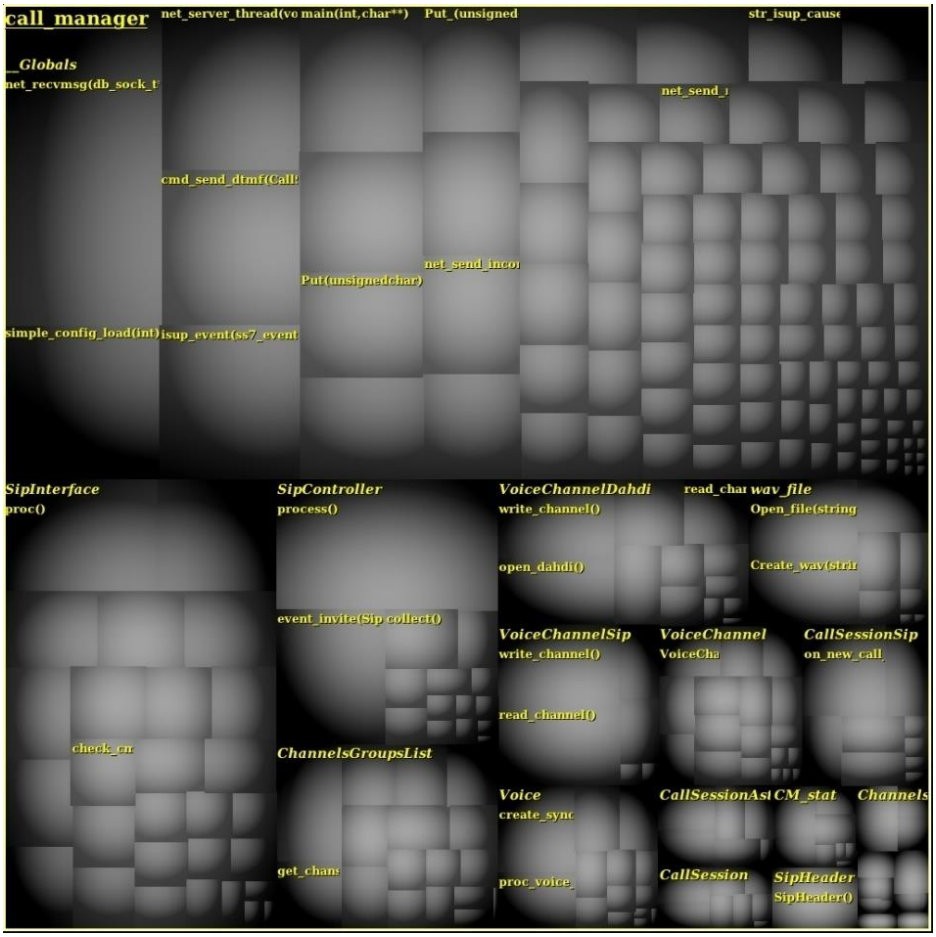

Figure 3. Project 3 matrix

This matrix clearly demonstrates the absolute and relative volumes of all program structures. The distribution of volumes is heterogeneous.

In the quality section of the code, that is in the generated CppDepend report, this project has one critical and twelve average warnings.

Having compared the three projects created with one common goal and analyzed the reports submitted by CppDepend, we compiled a competitive profile of all the analyzed projects. This competitive profile is presented in table 1.

Table 1. Competitive profile for three project.

Parameters	First project (items)	Second project (items)	Third project (items)
Total number of lines	8 834	21 742	5 061
Number of modules in the project	3	4	1
Types with too large volume	1 (1370 lines)	3 (1275;810;593 lines)	1 (725 lines)
Too complicated methods	4 (average:24,25)	29 (average:51.83)	4 (average:39.75)
Quick review of the refactoring method	116	216	49
Method with too large volume	76 (average:59,62)	116 (average:101.78)	40 (72.1)
Methods with a large number of parameters	-	3 (9;9)	2 (19;11)
Too complicated methods	25	70	11
Poorly commented methods	average89	211	47
Methods with too many variables	3 (18;18;18)	58 (average:25.9)	8 (average:19.11)
Types with a large number of methods	9 (average:47)	2 (40;28)	2 (28;24)
Types with a large number of fields	2 (35;37)	3 (35;29;28)	2 (31;29)
Types with poor cohesion	7	5	3

4. Discussion

Thus, the analysis of the quality of the program code in these three projects showed that the project No.3 was of the highest quality in terms of program metrics. This is proved by the competitive profile given in Table 1. Its code is more concise, the structure of the program is simple, and the code is commented within the normal range. The programmer adhered to small volumes, methods and types, too. This program is more stable than others, and therefore more manageable. The third project is most suitable for integration into other systems, as it can be easily improved and modernized according to the customer's demand.

In accordance with the task of developing unified documentation systems for QMS of the ISO 9001 standard, specialists in "Svyazcom Ltd." developed and wrote a quality manual and mandatory documented procedures (hereinafter referred to as DP): DP "Documentation management"; DP "Records management"; DP "Internal audit"; DP "Management of non-conforming products"; DP "Preventive and corrective actions"; DP "Human Resource management"; DP "Participation in tenders"; DP "Design and development of software"; DP "Technical support"; DP "Quality plan"; DP "Choice of the supplier".

Undoubtedly, there is some relationship between the metric parameters of the program code under examination, the documented procedures of the quality management system and the responsible participants in the project. This relationship can be represented as a three-dimensional matrix. One of the sections is a standard procedure, which is the responsibility matrix in accordance with ISO 9001.

The next dimension of documented procedures is metrics. However, not every documented procedure is directly related to code metrics. Table 2 below shows the relationship between the main software product metrics and documentary procedures.

Table 2. Metrics dependence on documented procedures.

Metrics / DP	Number of lines	% of comment coverage	Cyclomatic complexity	Centripetal and centrifugal cohesion (Stability)	Number of methods per class, tree depth, number of descendants	Number of fields and functions in the program code structure
Design and development						
Internal audit						
Management of non-conforming products						
Preventive and corrective actions						

Coloured area shows the importance of metrics and their verification at certain stages of the software life cycle. It is not advisable to check and control the quality of all metrics at each stage, but in order to correct and improve the quality of the program code writing, it is necessary to take them all into account at certain stages, as shown in colour in table 2.

The next dimension of the metric in the matrix shows those responsible for the software quality management. At certain aforesaid stages of the life cycle the project participants make their own assessment of the quality of the program code, depending on the degree of compliance with the certain metric. For example, the value of cyclomatic complexity shows how complex the program is and how it can be estimated in a point system. An example of such scoring is shown in table 3.

Table 3. Example of a point estimate for cyclomatic program complexity.

Complexity of the program	Error probability	Scoring
1 – 10	Simple function, errors are unlikely	5
11 – 20	More complex, average probability of errors	4
21 – 50	Complex, errors are likely	3
51 and over	Not tested (risk is very high)	2

For a better perception of the software code, the software package should be broken into blocks with a small number of lines which can be estimated in a similar way (Table 4).

Table 4. The number of lines in the structure.

N	The number of lines in the program structure	Scoring
1	less than 500	5
2	499-800	4
3	799-1000	3
4	over 1000	2

The instability of the program, which results from centripetal and centrifugal cohesion, can be estimated by the same principle.

For each project participant, one or another metric has a greater or lesser value. As a consequence, the scores of all participants will be different. Having estimated the latter, we can find the quality coefficient of the program code using the formula 1:

$$K = \frac{1}{E * L} \sum_{e=1}^{E} \sum_{l=1}^{L} k_{el} \tag{1}$$

where E – number of reliable people;
L – number of metrics;
k_{el} – estimates made by those responsible for compliance or non-compliance with certain metrics.

5. Conclusion

Hence, now it is possible to sum up some work results. Analysis of software quality in dynamics, i.e. analysis of the projects, similar in their functional purpose but differing in time of creation and their creators, showed that the quality of written program modules improves with the simultaneous simplified integration of them into other projects. This is evidenced by a competitive profile compiled for three projects, which demonstrates that the project No. 3 turned out to be the best from the point of view of quality of program metrics. Its code is more concise, the structure of the program is simple, and the code is commented within the normal range (19%). The cyclomatic complexity of the program methods equals 56, with the 568 types, which is better than the other programs in terms of parameters. For that reason, this program is more stable than the others, and therefore more manageable. The third project is the most suitable for integration into other systems, as it can be easily improved and modernized according to the customer's demand.

The CppDepend software allowed us to evaluate the quality of code for large volumes, to build a management and structure flow graph and, finally, to generate a report.

For the quality management system, the introduction of such a program would enable improving the quality of the code, reducing the time costs for the development of subsequent modules for other complexes. Within this program, it is possible to track code design and code writing against dates and schedules, which would help project managers to do their work related to tracking project duration without bothering programmers. It is proposed to introduce CppDepend as a design tool suitable for development environment for all programmers.

Metric synthesis showed that the quantity and quality of a program code depend on the logical and mental abilities of a programmer, their experience in developing software products and the degree of knowledge of a programming language. This provided us with a means to establish dependencies between program code quality metrics, mandatory documented procedures and those responsible for creating a software package at each stage of the life cycle. In the end, we found it feasible to offer "Svyazcom Ltd." to use in their QMS three-dimensional quality management matrix.

The three-dimensional matrix had the following dimensions:

- a dimension of documented procedures – metrics. The importance of metrics and their verification at certain stages of the software life cycle were shown.

- a dimension of documented procedures – people responsible. We specified a place of software checking and assessing the software quality in the product life cycle.

- a dimension of metrics – responsible. At certain aforesaid stages of the life cycle the project participants make their own assessment of the quality of the program code, depending on the degree of compliance with the certain metric. The combination of these estimates made it possible to find the quality factor of the program code, which is a rough estimate in its own way, be it a five point system or a hundred point system. The sum of metric values was ultimately reduced to a single numerical indicator, namely the quality coefficient of the program code.

The quality of the program code shows how stable and reliable the program is, the professional level of the programmers working in the company, and the speed and reliability of the company as a whole.

6. References

[1] Chernikov B V 2012 *Management of Quality Software.* (Moscow: Forum) 240 p.
[2] GOST ISO 9004-2010 2010 *National standard of the Russian Federation. Management to achieve sustainable success of the organization. Approach based on quality management. Introduced on November 23, 2010* (Moscow: Rosstandart) 47 p.
[3] Rostova, O., Shirokova, S., Sokolitsyna, N. Management of project for automation of investment control at industrial enterprise (2019) IOP Conference Series: Materials Science and Engineering, 497 (1), art. no. 012017, DOI: 10.1088/1757-899X/497/1/012017
[4] GOST ISO 9000-2011 2011 *Interstate standard. Quality management systems. Requirements. Introduced on December 12, 2011* (Moscow: Rosstandart), 32 p.
[5] GOST ISO 9000-2011 2011 *Interstate standard. Quality management systems. Key notions and vocabulary. Introduced on December 12, 2011* (Moscow: Rosstandart), 31 p.
[6] Bolsunovskaya, M., Shirokova, S., Loginova, A., Uspenskij, M. The development and application of non-standard approach to the management of a pilot project (2019) IOP Conference Series: Materials Science and Engineering, 497 (1), art. no. 012024, DOI: 10.1088/1757-899X/497/1/012024
[7] Lipaev V V 2011 *Design and manufacture of complex custom software products* (Moscow: SINTEG), 408 p.
[8] C and C++ Code Counter. Available: http://cccc.sourceforge.net (Accessed 11.02.2019.)
[9] Source monitor**.** Available: www.campwoodsw.com/sourcemonitor.html (Accessed 15.03.2019.)
[10] Habrahabr. Available: http://habrahabr.ru/post/111524/ (Accessed 15.04.2019)
[11] HomeCodeMetrics. Available: https://reflectoraddins.codeplex.com/wikipage?title=CodeMetrics&referringTitle= (Accessed 20.04.2019)
[12] Developer Network. Available: https://msdn.microsoft.com/en-us/library/bb429476.aspx / (Accessed 25.04.2019)
[13] Our products. “Svyazcom Ltd.” official website. Available: http://www.svyazcom.ru/ru/products/. (Accessed 10.05.2019)
[14] Klochkov, Y. 2018 Conflicts between quality management methods. 2017 International Conference on Infocom Technologies and Unmanned Systems: Trends and Future Directions, ICTUS 2017, Volume 2018-January, 34-36. DOI: 10.1109/ICTUS.2017.8285970
[15] Sokolov, A.A., Arkin, P.A. Planning of a project for development and introduction of an automated industrial engineering (2017) Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements, SCM 2017, art. no. 7970717, pp. 765-767, DOI: 10.1109/SCM.2017.7970717