The New Aspect on The Quality of Software with The Use of Knowledge Bases and The Semantic Web

Dalibor Misirača,  
University of Travnik, Travnik, Bosnia and Herzegovina  
&  
Mladen Radivojević  
University of Travnik, Travnik, Bosnia and Herzegovina

Abstract
The quality of the software becomes an exponentially critical factor with the rise of sophistication of the users, the complexity of technology and with increasing competition on the market of software products. In this paper, it is suggested the use of the semantic web and knowledge bases and the corresponding alternative orthodox, expensive and above all boring, ruling methodology which ultimatively demands for the best possible quality of the software product. It is known to everyone who deals with the quality of the software, that it is not important the number of mistakes in the software but the consequences that everyone of them can cause. It is always better to assess what to do and really do it (with the means available) but to insist on the slogan <<quality without compromise>>. The solution we propose will secure to those who deal with the quality of the software one different aspect of it and fast approach to the necessary knowledge updated with the use of Protégé editor open source platform.

The keywords: software quality, the needs of users, quality factors, knowledge bases.

1. INTRODUCTION

Today, information network covers the entire globe, enters every house, every factory and social institution, connects land vehicles, aeroplanes, ships and every individual. At the same time, the computers are equally found in large bank systems, industrial complexes and in directing direct traffic as well as in washing machines, cars and children's toys.

Observing from that perspective, the information, its distribution and processing have become a direct measure of accomplished level of development of any production (and not just production) system and thereby its full quality.

The sadder side of this story, which threatens to become euphoric is the irrefutable fact that the more information and knowledge we produce, the bigger is the importance of the quality of those information and knowledge.

What is interesting is the question of how much is the quality of those information dependent on the quality of the software which produces it?

2. Knowledge Bases and the Semantic Web\(^{[1]}\)

The term semantic web has introduced Tim Berners Lee (2001) as a clear structure to the contents of the Web page. It originated as a necessity of a more efficient finding of certain information and knowledge. It is based on the idea that the information on the web becomes machine readable.\(^{[2]}\) Instead of documents linked with hyperlinks it should use the interrelated data (information) which have a corresponding structure and meaning. In order for the idea of a semantic web to function, computers would have to have an access to the collections of information. It has to ensure the rules for reasoning on data, and to enable the presentation of data, information and knowledge.
Semantic web will enable the computers to find, understand and use the information via web in order to achieve certain goals. It covers plenty of different areas and there is certainly a few people that have completely the same idea on semantic web.[3]

In order for the idea of a semantic web to function, computers would have to have an access to the collections of information. It has to ensure the rules for reasoning on data, and to enable the presentation of data, information and knowledge.

Here we use also the Protégé editor, platform of an open source that will enable the users: reading and storing the basic knowledge related to the software quality. Protégé enables the users the displaying the meanings of terms and relations between that terms. It gives a rich gathering of structures for modelling and activities that encourage creation, visualization and manipulation of knowledge which is represented in different formats.

3. FIVE LEVELS OF MATURITY OF THE SOFTWARE PROCESS CMMM METHOD

CMM Method was developed in the USA for the requirements of the military industry (H-L Hausen, D.Welzel: "Guides to Software Evaluation"). Its objective was to determine the relevant elements which have to exist in the organization of the software engineering process and that will secure that the final result be of the defined quality.

It is based on the collection of the publicly available criteria, which describe the features of maturity of the organizations, that execute the process of software development (software process) and the process of continuous improvement of the existing through the multitude of small, evolutionary steps. CMM Method gives the basis and conducts the systematization of the evolutionary steps in the five levels of maturity and represents the guide post of organizations for improvement of the process and the transition from one level to the other higher level. All the maturity levels determine the real picture for evaluation and assessment of the software process implemented in the organization [4]. They also help the organization to set the priorities in improving their development process of the software [5]. Maturity levels are:

1. Initial: software process is executed from time to time, chaotically. It is defined only a few processes and the success of the project depends on the individual effort.

2. Repeatable: Project directing is based on the monitoring of costs, plan and functionality. Discipline is needed in the conducting of the process based on the repetition of the earlier successful projects.

3. Defined: Engineering activities are documented, standardized and integrated into the software process for the organization. The projects use the coordinated and customized version of the standardized software process of organization for the development and the maintenance of software.

4. Managing: Detailed data are collected about the software process and the product quality. Software processes and products are quantified and controlled.

5. Optimized: Constant improvement of the process is enabled by quantitative feedback from the process and from managing of innovative ideas and technologies.

We will use Protégé, open source platform for presenting the basic knowledge of the software process. (Picture 1 Software processes). In order to be able to follow the contents of the base, we present it in the English language on which it is updated:

The project of software processes Projekt softverskih procesa

Initial - software process is executed from time to time, chaotically. Inicijalni - Softverski proces se izvodi s vremena na vrijeme, haotično - the success of the project depends on the individual effort. Uspjeh projekta zavisi od individualnog napora.

Repeatable - Project directing is based on the monitoring of costs, plan and functionality. Upravljanje projektom zasnovano na praćenju troškova, plana i funkcionalnosti - Discipline is needed in the conducting of the process Potrebna disciplina u obavljanju procesa

Defined - Engineering activities are documented, standardized and integrated Definisan - Inžinjerijske aktivnosti su dokumentovane, standardizovane i integrirane – The projects are tuned for development and maintenance. Projekti usaglašeni za razvoj i održavanje
Managing – The data are collected about the software process and the quality of the product. Software processes are quantified and controlled. Optimized - Constant improvement of the process - enabled by quantitative feedback from the process and from managing.

Figure 1 Software processes

3. DIFFERENT ASPECTS ON THE QUALITY

Quality software is a very simple concept (at least according to the various books which deal with this topic). It is only needed to define your demands and to secure the mechanisms that will guarantee that they will be fulfilled. For this, it is enough to verify if the team of software people is filled and if the enough amount of time is set aside to complete the job. Apart from that, it should be verified if the procedures of quality evaluation are strictly applied in every phase of life cycle, starting from the defining of demands and finally with final testing.

By any means it shouldn’t be forgotten on constantly taking care of convincing the management that the quality has no price and alternative, otherwise all the rest should be simply forgotten.

Unfortunately, in the real world the software quality is not so simple:

• The demands are constantly changing and the desired quality flickers as mirage of a desert oasis.
• There is a constant lack of needed personnel and the significant missed deadlines are the rule rather than the exemption.
• Quality management becomes a synonym for “ad hoc” testing.
• Management is much more interested in realizing quick profit but to absorb into the finesse of software engineering.

It is often thought about the quality as a substance, that is, about something what in principle can be easily estimated by simple look on the pointers of the meter. However, the quality is more
similar to the optical illusion [6]. It is a complex abstraction which is partly made of the one that observes it, partly from the observed and in bigger part of the impression that creates the sole process of observing. The problems that affect the software quality are presented in the picture 2.

At present, when the software quality is in question, in practice is mostly used the following three approaches:

**Aesthetic aspect** - Quality is elegance, indescribable experience of good. This perspective is most frequently supported from the creators of software which describe it with the whole array of different attributes, but they cannot unequivocally describe the essence of the idea.

This approach should be supported because it lifts the moral and satisfaction and the pride of the results of the achieved work. The danger of this approach is that it can become the refuge for perfectionists and those who never did anything.

**Aspect of the manufacturer** – Quality is correctness, compatibility with the demands (specifications). This aspect opens the Pandora’s Box, because it ultimately presents the painful question the ability of creating the quality specification. By doing this, it doesn’t help at all in solving the problems of design or comparing the relative significance (importance) of different types of nonconformities.

**User’s view** – The quality is the possibility of using: simply everything that satisfies the needs of the user. The problem with this approach is that the quality becomes not only the subjective category (as it was with the aesthetic approach) but it is also irretrievably lost of control over the overall business:

- Who are my users?
- How to incorporate their demands and needs into the product that they never saw?
  It can certainly be more than one user of the software product, or the users don’t know to express what they exactly want or what they want is not simply achievable [7].

The danger of this approach is that we are constantly in the vortex of attempts to be serviceable instead of maximally concentrating on creating such a product of which we are going to be certain that it will make the user happy as soon as he saw it.

The fact is that the user’s (internal and external) can never objectively assess the real quality of the product. Perception of the user is conditioned by the adopted value system, knowledge level, previous experience and purpose (point of using). Some of these factors we can study but objectively we cannot control any of them. Some of the mentioned knowledge we have updated by using Protégé editor, an open source platform. (figure 3).
4. IMPOSSIBILITY OF THE EFFECTIVE APPLICATION OF METRICS IN PRACTICE

The measurable quality factors are still fiction and elusive dream. In the Encyclopedia of Software Engineering there is an excellent article about the quality factors of software products and ideas in order to measure those factors. According to this approach the quality management is reduced to evaluation if it is present (and in what measure) in the software product every of the above mentioned quality factors. Thereby, everyone should take into account the three previous mentioned aspects on the quality (aesthetic, the manufacturer’s and the user’s).

However, in practical application the problem is not so simple because (depending on the concrete context) some factors are more significant than the rest. Some factors are mutually conflicted, like, for example, it is the known problem of functionality/transferability. For most of the factors, it will show in practice that there are no objective and payable ways to measure them. Further, there is no simple way that for one time measured factors shape into generally acceptable quality metrics.

5. SUGGESTION OF THE SOLUTION

To briefly summarize what we have achieved until today:

- We have some idea about what is the software quality but we are not in the condition to define that term in precise manner.
- There are corresponding methods for measurement of the software quality, but it cannot be said that any of it is the one.
- Studying and managing of the quality is burdened with:
  - Complexity of the software products
  - Intangibibility of the software product,
  - Sensibility of the software product even for the tiniest mistakes,
  - Interaction with the external components,
  - Interaction with the user.
Ultimately the satisfaction of demands for creating software product, which will be of maximum quality, will in practice show as a very expensive enterprise while on the other hand most of the users are not simply in the condition to recognize the difference between the maximum and “sufficiently good” quality.

This leads us to the three critical questions:

- Which of the quality factors of the software products will show in the concrete situation as useful and acceptable and to what extent?
- How to measure it correctly?
- How to control it adequately?

One of the answers to these questions is to forget the endless theories on measuring quality. Instead of our endless effort to create a universal quality metrics, in order to be able after that to optimize everything according to that measurement standard, we should concentrate on the problems which the quality is trying to solve. If these problems are successfully solved, it is automatically secured “sufficiently good” quality.

For the needs of the insurance of the quality of the software product, the most suitable would be the definition of the quality as an optimal collection of solutions for the given collections of problems. From this perspective the answers to the previously set three critical questions will follow on the basis of:

- The process of understanding problems with which we are faced,
- Iterative studying of the possible solutions and limitations and
- Connecting the chosen solution of the problem with the corresponding processes and existing software standards.

Successful managers, which deal with the software quality, express the important fact that it is not important the number of the mistakes, but the consequences which can produce each of them. [9] The problem is in how the risk can be effectively estimated and controlled. However, it is always better to estimate what should be done and to do it concretely, instead of playing psychological games, like it is for example “six sigma” and the use of the slogan like “Quality without compromise”.

We used Protégé editor for creating a chart – possible solutions.

Figure 4. OntoGraf – one of the suggestions of the solutions
CONCLUSION

The appearance of industrial, technological and informatics revolution has brought to the point that the software products today have become, like fire and water, good servants but bad masters. In biblical times had arisen the slogan of quality system: “Better safe than sorry”, which is very up-to-date for the software quality. The quality means the compliance with the demands. There is no such thing that can be called the economy of compliance – it is always cheaper to do a good job from the first attempt.

This can be called hygiene factors, given that they function in the way similar to the principals of medical hygiene. The objective of the hygiene is to remove the health risks of the human environment. It does not cure. It acts preventively.

The prevention is, certainly, of the most important significance. But it is not the prevention of everything and everyone but the maximum prevention of the proven important problems. In order to be sufficiently good, it is needed with certainty to make differences between the important and unimportant, necessary and unnecessary.

REFERENCES

9. Y.Wang, [1997]: "What the Software Industry Thinks About the Quality System", Proceedings of International Conference on Software Quality - ICSQ ’97, Maribor