TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 63, Issue Special, October, 2020

SOFTWARE ENGINEERING PERFORMANCE AND QUALITY ASSESSMENT BY TRIZ

Kangrok LEE, Jaemin SHIM, Jinyong KIM

Abstract: *This paper shows the innovative concept of how software engineering activity performance and quality assessment can be done. To be successful in delivering the software services/products, we should have some kind of measurements about the ongoing productivity and the current status. To achieve this, we have used the substance-field model and 76 inventive standard of TRIZ. Also, we have described how to visualize the activities through dashboards by KPIs (Key Performance Indexes) to be able to take some important timely decisions that help deliver the services/products on time. All this was done gathering the complexity of each individual activity and time information from various information sources. In this context, we must insist that there are some specialties that impact software engineering, particularly internet advertisement services. Therefore, we would like to clarify the necessity of having the TRIZ approach in empowering the productivity of our engineers in software engineering for software quality assessment.*

Key words: *Software Engineering, Technical Debt, Substance Field Modeling, 76 Inventive Standards, Productivity, KPI (Key Performance Index) Dashboard.*

1. INTRODUCTION

In order to provide a comprehensive snapshot of our current engineering projects, it is required to have an introduction of a new approach, as well as tools to gather the time series data about the past from various sources, which helps us to identify the future trends in order to make accurate decisions. For assessing performance and quality it is mandatory to have some kind of measurements to manage and gauge the activities. [1] First of all, quality is everyone's responsibility and it should be followed through in each activity. This means that comprehensive plans should be devised to show the exact responsibilities and decisions of various services. Procedures should be defined to check their conformance with the development plans. Countermeasures about 3-tier problems should be suggested in the event of discrepancies between performance and specification, despite already having developed several research projects and international standards about software quality and measurement.[2] We can neither incorporate nor adjust them into our software quality assurance and performance measurement process because of their outdated nature to the current trends. Firstly, we have a definite need to make up a new set of guidelines and standards for the online business for development and maintenance of internet advertisement services and technical engineering. Secondly, we should also verify the performance index, measure it, and track for advertisement services. All the need performance indexes should be explained in a more detailed analytical way, having a high degree of confidence and being able to propose a KPI dashboard to cover all of them.

2. SOFTWARE ENGINEERING BY TRIZ

For so long, there have been many research papers and books about software and TRIZ. Not quite as big and impactful, but very imperative, is the survey of concurrent relationships between software and TRIZ. In [3] the author promised to formulate a domain-specific matrix by re-architecting successful architectures. Despite finding the correspondences between software

architectures and TRIZ, he does not seem able to interpret the whole contradiction matrix into software architecture terms. Darrel Mann has published a paper [4] in the TRIZ Journal, which serves as a summary of his forthcoming book "TRIZ for Software Engineers" [5]. It states that there are 7 pillars of different levels that detail the starting point of software engineering concerning TRIZ. He has concluded that this could create tools for software engineers in order for them to do a better job with regard to TRIZ tools for the emergence of definitions and methodology.

3. MODEL AND ANALYSIS FOR PERFORMANCE AND QUALITY ASSESSMENT

In this article's second chapter we have discovered that it is a difficult task to perfectly apply TRIZ in the software's domain and architecture. However, some have already done the necessary research to understand the relationship between TRIZ tools and software knowledge and have concluded that it is better to have the methodology and tools to track and assess the software engineering activities. Thus, we need to make a new model and analyze our challenges with regards to software performance and quality assessment. As follows, we have tried to describe that there are new approaches and analysis for our challenging problems with substance field model and 76 inventive standards. Firstly, we will analyze our system, its functions and features, as seen below.

Table 1

Example Description of System, Functions and Features

System	Function	Feature (Parameter)
Campaign Management	Manage	Accuracy
Data Pipeline	Process	Speed
Advertisement Delivery	Deliver	Time
Tracking	Track	Precision
Aggregation	Aggregate	Accuracy

In Table 1, it is stated that advertisement systems are generally operating, functions are being executed and features will be changed from the current status to the desired one.

3.1 Substance Field Model and 76 inventive standards

In this chapter, we will explain that we have used the substance field model and 76 inventive standards to decide the strict rule of checking our software quality and performance. In general, competency is highly likely to be related to effectiveness and can result in better performance, but not always, because all the works cannot always be assigned to the right competency with consistency. So, when it comes to fair and meaningful performance, we need to clarify three things – competency, works, and performance. Besides, there always have been misconceptions about activities such as "simple problems like software bug", technical debts, etc., a non-typical problem like a too complex root cause, misunderstandings that take place due to human interactions. That is why we need to make more abstractions for the definitions of our problems with the substance field model, in order to simplify our typical problems. Separately, it is possible to make our problem model typical and simple with the substance field model and 76 inventive standards. We do this so that we can understand their cause and effect the right way in order to improve it further. In the Fig. 1, we can assume that substances 1, 2 are our advertisement system and software. Fields in our domain are considered to be: development, implementation, test, etc. All fields can affect our software. There may be several types of work and tools, but there may be invisible and impractical things that are affected by our engineers.

Fig. 1. Substance Field Model

In addition, 76 inventive standards are composed of the following 5 classes. We consider we should have a strategic direction regarding our substance field model.

- Class 1: Synthesis and decomposition of substance fields. These solutions improve the useful interaction effect or eliminate a harmful function. Basic and initial scheme for using 76 inventive standards (13 standards)
- Class 2: Evolution of the substance field model. These solutions push the system further towards ideality by suggesting other controlled fields or advanced technology and software such as AI, deep learning, etc. (23 standards)
- Class 3: Transitions towards supersystem or microlevel. These standards will guide us towards multi-systems and miniaturization. (6 standards)
- Class 4: Measurement and detection. These patterns improve the measurement and detection issues in the system. Most important is to define our performance and quality assessment for our system and software. (17 standards)
- Class 5: Guidance and Helpers. The last set of standards will introduce fields and substances in order to obtain better results. (17 standards)

3.2 76 Inventive Standards and Analysis for Software Performance and Quality Assessment

When using all 76 inventive standards, we could not apply the measurements, nor could we detect problems in a particular class, such as class 3, class 5. We should focus on class 4, especially in chapter 3.1, in order to dive deeper into more statements and rules in our problems and situations. Group 4-1 explained the detour. Instead of measuring and detecting problems, it states we should change the system without the need for measurement and detection. Only one alternative is copied when it is impossible to change the system without measurement and detection. Finally, we apply for the modification of the measurement problem in successive detection. All detections must be made only with a certain degree of correctness. Therefore, it is possible to detect them with the method of two consecutive detections, although some problems should be detected with continuous values. Therefore, it is necessary to introduce MTBF (average failure time) and MTTR (average repair time) to check the quality and performance of the software. Group 4-2 is represented as a synthesis of a measurement system. This is described in Figure 2 and we have realized that the problem situations are similar to our problems for software performance and quality assessment. Although the substance and the field are slightly different from the system and software in the software field, it has been found that there are many interactions and new flows of technology and development between many systems and software. In Figure 2, it is better to introduce other substances that are easy to measure and detect in our system because it is difficult to measure and detect for the existing system. Therefore, we created the KPI (Key Performance Index) dashboard to facilitate the detection and measurement of our systems or software for operating the advertising service. To check our productivity, we introduce the productivity and transparency of our systems and software, as seen in figure 3.

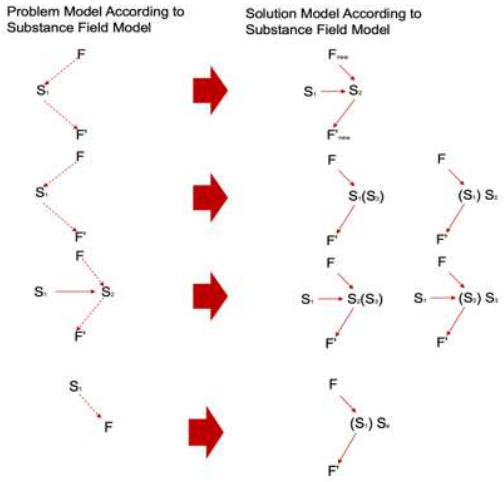

Fig. 2. Class 4: Measurement and Detection Diagrams

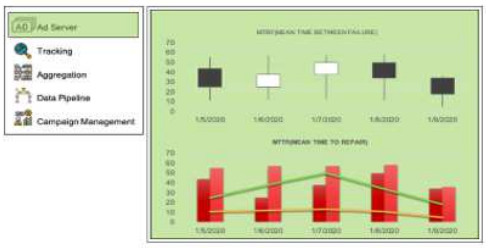

Fig. 3. KPI (Key Performance Index) Dashboard

4. CONCLUSION

Checking software performance and evaluating quality are basic issues in software engineering. This task depends very much on the experience and knowledge of the engineer. The use of TRIZ substance field model and 76 inventive standards can help direct the measurement and detection of problems in a promising heuristic direction. Therefore, these tools can be seen as an extension of software performance and quality evaluations. This paper insists on the approach of finding the correspondences between the substance field model and software engineering. In addition, we emphasize that it is important to check the quality and measurement of software performance against some rules of 76 inventive standards when we effectively improve our software and systems and estimate the accounting solution at the top of all possible improvements. Although not all 76 inventive standards are useful in software engineering, the remaining models and rules can be useful and understandable in decision making. In fact, some of the inventive standards for the field model found in other fields can also be applied to software engineering. Despite the founding correspondences, it does not seem possible to translate 76 inventive standards into performance and quality assessments of software engineering. In essence, it is very important to use these tools with the level of abstraction required by software engineers. [6] In addition, it is quite promising to find relevant and meaningful measurements and measurements of software performance and quality with these tools.

5. REFERENCES

[1] Robert S. Kaplan: *Conceptual Foundations of the Balanced Scorecard.* Handbooks of Management Accounting Research, volume 3, 1253–1269 (2009).

[2] Consortium for IT Software Quality, 2020. CISQ *Specifications for Automated SoftwareMeasurement. Available* from: https://www.it-cisq.org/standards/index.htm

[3] Daniel Kluender: *TRIZ for software architecture*, TRIZ future conference 2006, Volume 9, 2011, 708 – 713 (2011)

[4] Darell Mann: *TRIZ For Software*, TRIZ Journal, Oct. 2004

[5] Darell Mann: *TRIZ for software engineers*, IFR Press, 2008

[6] Usharani Hareesh Govindarajan, D. Daniel Sheu, Darell Mann: *Review of Systematic Software Innovation Using TRIZ*, Int. J. Systematic Innovation, 5(3), 72-90 (2019)

Performanța ingineriei software și evaluarea calității cu TRIZ

Rezumat: Această lucrare prezintă conceptul inovator al modului în care se poate face performanța activității de inginerie software și evaluarea calității. Pentru a avea succes în furnizarea serviciilor / produselor software, ar trebui să avem un fel de măsurători despre productivitatea continuă și starea actuală. Pentru a realiza acest lucru, am folosit modelul substanță-câmp și 76 standarde inventive ale TRIZ. De asemenea, am descris cum să vizualizăm activitățile prin tablouri de bord, prin KPI (indexuri de performanță cheie) pentru a putea lua unele decizii importante în timp util, care ajută la furnizarea serviciilor / produselor la timp. Toate acestea au fost realizate adunând complexitatea fiecărei activități individuale și informații despre timp din diverse surse de informare. În acest context, trebuie să insistăm că există anumite domenii care afectează ingineria software, în special serviciile de publicitate pe internet. Prin urmare, am dori să clarificăm necesitatea unei abordări TRIZ pentru încurajarea productivității inginerilor noștri din ingineria software pentru evaluarea calității.

Kangrok LEE, Global Ad Technology Supervisory Department of Rakuten Inc., Japan, kangrok.lee@rakuten.com

Jaemin SHIM, Rakuten Asia Pte. Ltd. of Rakuten Inc., Singapore, jaemin.shim@rakuten.com

Jinyong KIM, Global Ad Technology Supervisory Department of Rakuten Inc., Japan, jinyong.kim@rakuten.com