Forward Engineering Completeness for Software by Using Requirements Validation Framework

Nayyar Iqbal[1,2,3], Jun Sang[1,2], Min Gao[1,2], Haibo Hu[1,2], Hong Xiang[1,2]
[1]Key Laboratory of Dependable Service Computing in Cyber Physical Society of Ministry of Education, Chongqing University, Chongqing 400044, China
[2]School of Big Data & Software Engineering, Chongqing University, Chongqing 401331, China
[3]Department of Computer Science, University of Agriculture Faisalabad, Faisalabad 38000, Pakistan
nayyariqbal@cqu.edu.cn, jsang@cqu.edu.cn, gaomin@cqu.edu.cn, haibo.hu@cqu.edu.cn, xianghong@cqu.edu.cn

***Abstract*—In software development environment, software companies usually ignore the user requirements validation process in requirement gathering phase, which results in large number of modifications being required in the software maintenance phase to fulfill the customer requirements. Identification of accurate requirements from user stories and determining the effectiveness of work deliverable of software industry has always been a challenging task. In this paper, a new measurement approach for forward engineering completeness for software was introduced by using requirements validation framework. The forward engineering completeness for software was measured in two steps. In the first step, software component structure was developed in order to find the functional and non-functional requirements rejected by the customers in the requirement validation framework. In the second step, completeness of software from component-based development was determined in which the following parameters, such as functional, non-functional completeness attributes, were considered in the measurement process, and the unadopted attributes of the reuse code were also considered. Quality level for the attributes were assigned based upon the valuation of interior quality of the source code. Therefore, it resulted in the reduction of development time required for the software and the cost required for the software development was also reduced. A case study was incorporated in this research to explain the measurement process of forward engineering completeness. If the forward engineering code is satisfying the quality standards, then the code is in the completeness form. The attributes of code that negates to be used were considered as unadopted attributes.**

Keywords-completeness; forward engineering; functional requirements; non-functional requirements; requirements engineering; validation

I. Introduction

It has been observed that software industries that abused the Requirement Engineering (RE) in the software development process resulted in the project failures [1-6]. Ana-Maria et al. [7] argued that business analysts needed to focus on requirement gathering techniques in a technically responsible way. Question had been raised as to the relationship between the functional and non-functional requirements, with some, such as Vishal and Xiaoqing [8], argued that there was reciprocal relationship. Software quality measurement is one of the most complicated tasks in software design methodology [9]. Quality of the software can be determined by the success of software system for this various parameters, framework and methodologies has been proposed [10]. Forward Engineering as defined by Pressman [11] "In most cases, forward engineering does not simply create a modern equivalent of an older program. Rather, new user and technology requirements are integrated into the reengineering effort. The redeveloped program extends the capabilities of the older application". Reverse and forward engineering are practiced in the legacy systems to extend the system usable lifespan [12].

DOI reference number: 10.18293/SEKE2019-144

Requirements are client's invariant statements related to sub system or system [13]. Functional requirement describes the complete functionality of software components that should be required in the software. Non-functional requirement describes the requirements of software with respect to security, usability, portability, availability, capacity, efficiency and reliability [14]. In past lot of work has been done by the researchers on functional and non-functional requirements. But illustrating the graphical user interface for user requirements in the software specification, in order to validate the user requirements has always been ignored in the requirement gathering phase. This research focus on the importance of validation of user requirements so that time and budget wasted in the modification of software in the maintenance phase can be saved. Thalheim [15] suggested the design quality parameters which include completeness, naturalness, minimality and flexibility. The software after the development process is said to be in the completeness form, if it satisfies all the functional and non-functional requirements. Component is a reusable visible interface, which is the factored form of any software or sub system. Software architecture is a static structure that represents arrangement of components [13]. This research was conducted with the collaboration of software company. In this new template is introduced by the authors that demonstrates the requirements of software components, which is illustrated in Table 1. In order to identify the adjusted and unadopted requirements Table 1 was discussed in the requirements validation framework.

In this research authors defines the two types of requirements unadopted and adjusted requirements. The functional or non-functional requirements rejected by Chief Executive Officer (CEO) of software users/customers are called unadopted requirements. Business analyst gathers the software requirements from software users/customers in natural language, after this these requirements were illustrated in Table 1, i.e. if software users/customers identifies that user login should not be by user name and password, but it must be by any of the followings: thumb scan, scan of Quick Response (QR) code or scanning the bar code of employee card etc. in the requirement validation framework then the rejected requirements are called unadopted requirements. Adjusted requirements are new requirements (functional or non-functional requirements) which are added in the software according to users demand or when any existing software components are replaced with new software components, then new functional and non-functional requirements are incorporated into the software. Examples includes: replacement of software component of login (email address and password) with QR code. In addition, as existing functionality of software was to calculate percentile of student result and now the customer of software has demanded that the software must also calculate Cumulative Grade Point Average (CGPA) of the student. Completeness is defined as "the state or condition of having all the necessary or appropriate parts" [16]. Whereas requirements completeness is defined as "a quality demanded to the set of software requirements and to each requirement itself, in order to ensure that there is no information left aside" [17].

In order to adapt the complete customer requirements in the software, software industries are developing the software globally [18]. The purpose of global software development is to gather the adjusted requirements of the software. As different countries use different social network software, such as Instagram, Twitter, Reddit, and Facebook etc., according to their requirements, therefore different regions in the world has different adjusted requirements. These differences are due to cultural difference, language difference, platform difference, business process difference and how the organization interpret with manual work. Different countries have different cultural and business-related problems so there is need to develop the software that captures the complete organization processing tasks. For this adjusted requirement must be incorporated in the requirement gathering phase, so that the developed software must be in the complete form.

II. METHODOLOGY

In this research authors develops the software by using forward engineering completeness approach. The methodology structure is illustrated in the Fig. 1. The developed system is said to be in forward engineering completeness, if it is developed with complete conditions or states of new business procedures and rules according to software engineering philosophies. In System Specification (SyS) information related to functional requirements, data requirements, quality requirements and constraints for software was determined. Problem definition, objectives, goals, context and major capabilities of the software was determined.

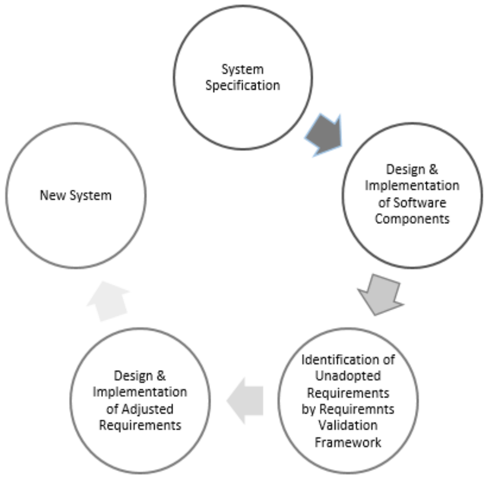

System Specification
Design & Implementation of Software Components
Identification of Unadopted Requirements by Requiremnts Validation Framework
Design & Implementation of Adjusted Requirements
New System

Figure 1. Forward Engineering Completeness

The purpose of requirement validation framework was to identify unadopted requirements as illustrated in the Fig. 2. In this business requirements were gathered from the users in which user identifies the goal and objectives of the system.

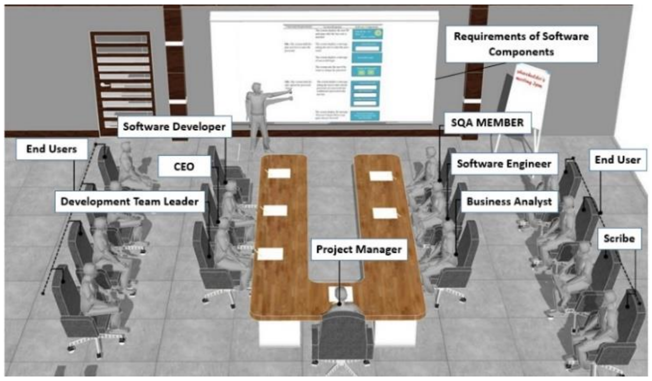


Figure 1. Requirements Validation Framework

The business analyst specifies the functional and non-functional requirements of the system. The business analyst and software engineer completed the task of software specification as shown in Table 1. The Software Quality Assurance (SQA) team members performed testing on the software to identify the errors in the software components. The basic purpose of this Table 1 was to present it in the framework meeting so the different categories of end-users from different regions can present their views. Scribe writes the report of the meeting. Chief executive officer attended the meeting along with end-users. Project manager and development team leader monitored the complete working process from requirement gathering to validation process. Software developer delivers the presentation of software requirements. The advantage of using this requirement validation framework showed the successful completion of software because after this process software modifications were not required in the software maintenance phase.

Rules & Definitions

Where S stands for Software, R_i, FR_j, NFR_k stands for n number of Requirements, Functional Requirements and Non-Functional Requirements respectively. Where $1 \leq i \leq n$, $1 \leq j \leq n$ and $1 \leq k \leq n$.

$$S\ (R_1, R_2, R_3, \ldots R_n)$$

whereas

$$FR_j\ \&\ NFR_k \in R_i$$

As defined by Sommerville [19]

S ("what a software should do" & "how the system will do so")

Therefore

$$S\ (FR_1\ \&\ NFR_1, FR_2\ \&\ NFR_2, FR_3\ \&\ NFR_3, \ldots FR_n\ \&\ NFR_n)$$

Unadopted and adjusted requirements can be functional or non-functional requirements. Unadopted functional, unadopted non-functional, adjusted functional and adjusted non-functional requirements are represented by UFR_l, $UNFR_m$, AFR_p and $ANFR_q$ respectively. *SFEC* stands for software developed by forward engineering completeness approach. Where $1 \leq l < n$, $1 \leq m < n$, $1 \leq p < n$ and $1 \leq q < n$.

$$FR_j\ \&\ NFR_k \in S \text{ and}$$

also

$$UFR_l\ \&\ UNFR_m \in S$$

but

$$AFR_p\ \&\ ANFR_q \notin S$$

whereas

$$AFR_p\ \&\ ANFR_q \in SFEC$$

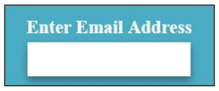
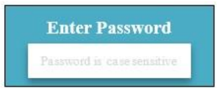
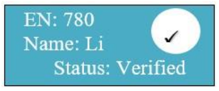
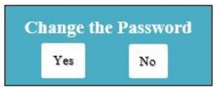
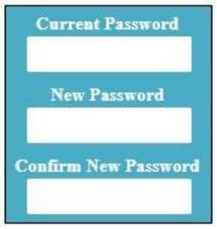
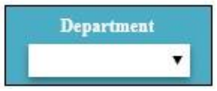
In order to measure the Forward Engineering Completeness for the software this research incorporates case study and two steps were considered in the measurement process. In the first step, unadopted requirements were identified in the requirement validation framework and for adjusted requirements, structure of software component was also discussed. In the second step, software completeness was calculated by using following parameters. *i-First parameter* was functional and non-functional requirements attributes. These attributes determine the completeness of the forward engineering. In this integration among the attributes was also determined, which increases the completeness value. More completeness scales the forward engineering process closer to the actual budget and development schedule of the software and vice versa. *ii-Second parameter* was unadopted attributes. In which authors identified those attributes that were not required in the software by requirement validation framework. The identification of unadopted attributes helps in budget and time saving. The time spent on developing the software components that were not required in the developed software, was saved by identifying the unadopted attribute in the initial phase.

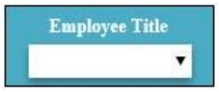
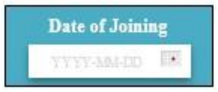
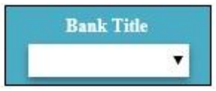
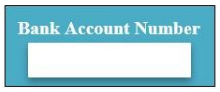
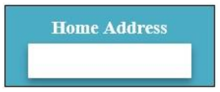
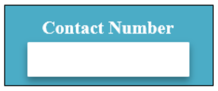
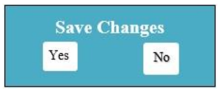
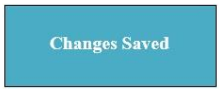
As clarified by Sommerville [19] "the non-functional requirements should define the usability, security, availability and performance requirements of the service". Therefore, usability and security were important for each component in non-functional requirements. The introduced technique was helpful for software engineers to measure the forward engineering completeness for software. In forward engineering, the software changes can be in terms of technology or adding new functionality e.g. if software was developed in old technology it can be changed to new technology for this, functional, non-functional and unadopted requirements were identified according to plate form difference.

III. CASE STUDY

Table 1 consists of three columns, first column describes the functional requirements of software, second column describes the system response and the third column illustrates the software components structure. Table 1 was discussed in the requirement validation framework. The non-functional requirements where were applicable, were explained by using software component structures in the requirement validation framework. In the pre-condition, user login the system by using organization email address. The organization email addresses and default password were issued by the organization for their employees.

TABLE I. REQUIREMENTS OF SOFTWARE COMPONENTS

Functional Requirements	**System Response**	**Software Components**
FR_1: The system shall display text box to enter the email address	The system displays a message asking the user to enter organization email address	
FR_2: The system shall display text box to enter the password	The system displays a message asking the user to enter the password	
	The system displays a message of "Successful Login"	
	The system displays the user Employee Number (EN) and name after the login	
FR_3: The system shall display options (yes/no) to change the password	The system asks the user if he/she wants to change the password	
	If the user clicks on the yes option, the system displays a message asking the user to enter current password, new password and confirm new password	
	The system displays the message "Password Changed Please login again with new Password"	
FR_4: The system shall display drop-down list for the selection of department	The system displays a message asking the user to select the department from the drop-down list	

FR$_5$**:** The system shall display drop-down list for the selection of employee title	The system displays a message asking the user to select employee title from the drop-down list	
FR$_6$**:** The system shall display calendar control for the selection of date of joining	The system displays a message asking the user to select date of joining the organization from the calendar control	
FR$_7$**:** The system shall display drop-down list for the selection of bank title	The system displays a message asking the user to select bank title from the drop-down list	
FR$_8$**:** The system shall display text box to enter bank account number	The system displays a message asking the user to enter bank account number	
FR$_9$**:** The system shall display text box to enter home address	The system displays a message asking the user to enter home address	
FR$_{10}$**:** The system shall display text box to enter contact number	The system displays a message asking the user to enter contact number	
FR$_{11}$**:** The system shall display options (yes/no) to save the changes	The system displays a message asking the user whether he/she wants to save the required data in the software or not	
	The system displays the message "Changes Saved" if user clicks on the yes option	

A. *Condition 1*

Customer$_1$ from organization$_1$, requested the modification in the software, functional and non-functional requirements for user login were by scanning the bar code of employee card with the bar code reader instead of login by email address and password. The bar code reader will be connected with the system through serial port or interface device called wedge or keyboard port. The bar code of the card will be matched with the repository of the user saved in the software in order to find the user matching text (identification).

According to Table 1, Functional Requirement Attributes *(FRA)* are those that defines the system behavior under precise circumstances. *FRA (email_address, password, change_password, current_password, new_password, confirm_new_password, department, employee_title, date_of_joining, bank_title, bank_account_number, home_address, contact_number, save_changes).* Non-functional Requirement Attributes *(NFRA)* are those that defines in what way a system must act and create restraints on its functionality. *NFRA (security, usability, portability, availability, capacity, efficiency, reliability, performance, integrity, recovery, compatibility, maintainability).* Unadopted Attributes, Total functional and non-functional requirements Attributes are represented by UA and TA respectively. In this following were the unadopted attributes *email_address, password, change_password, current_password, new_password, confirm_new_password. FRA = 14, NFRA = 12, UA = 6, TA = 26.*

Functional Requirement Attributes Completeness

$$(FRAC) = FRA/TA = 14/26 = 0.54$$

Non-Functional Requirements Attributes Completeness

$$(NFRAC) = NFRA/TA = 12/26 = 0.46$$

Unadopted Attributes Completeness

$$(UAC) = UA/TA = 6/26 = 0.23$$

$$\text{Software Completeness} = FRAC + NFRAC - UAC$$
$$= 0.54 + 0.46 - 0.23 = 0.77$$

As the value are represented in unit interval (0, 1).

B. *Condition 2*

Customer$_2$ from organization$_2$, whose functional and non-functional requirement were: when the user login the system by email address and password. The system shall send PIN at the user cellphone for further authentication of user. So, there was requirement of new functionality by the user to be added in the software. The new functionality was required to be integrated with email software component. According to Table 1, *FRA = 14, NFRA = 12, UA = 1, TA = 26.*

Functional Requirement Attributes Completeness

$$(FRAC) = FRA/TA = 14/26 = 0.54$$

Non-Functional Requirements Attributes Completeness

$$(NFRAC) = NFRA/TA = 12/26 = 0.46$$

Unadopted Attributes Completeness

$$(UAC) = UA/TA = 1/26 = 0.04$$

$$\text{Software Completeness} = FRAC + NFRAC - UAC$$
$$= 0.54 + 0.46 – 0.04 = 0.96$$

C. *Condition 3*

Customer$_3$ from organization$_3$, functional and non-functional requirements for user login were by thumb scan or by scanning the Quick Response (QR) code instead of login by email address and password. QR code functionalities are represented in the Fig. 3 [10].

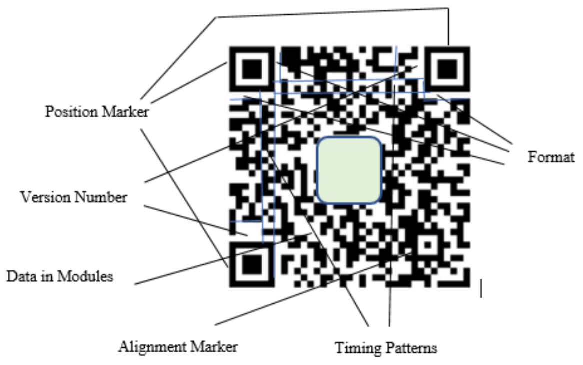


Figure 3. QR Code

According to Table 1, the value of *FRA*, *NFRA*, *UA* and *TA* were same as for condition 1, because unadopted attributes in both conditions were same.

$$\text{Software Completeness} = FRAC + NFRAC - UAC$$
$$= 0.54 + 0.46 - 0.23 = 0.77$$

D. Condition 4

Customer$_4$ from organization$_4$, functional and non-functional requirements were by selecting images for the password. The images must be available in the software. According to Table 1, *FRA = 14, NFRA = 12, UA = 4, TA = 26.* In this unadopted attribute were *password, current_password, new_password, confirm_new_password.*

Functional Requirement Attributes Completeness
(FRAC) = FRA/TA = 14/26 = 0.54
Non-Functional Requirements Attributes Completeness
(NFRAC) = NFRA/TA = 12/26 = 0.46
Unadopted Attributes Completeness
(UAC) = UA/TA = 4/26 = 0.15
Software Completeness = FRAC + NFRAC - UAC
= 0.54 + 0.46 - 0.15 = 0.85

Figure 4. Images Displayed for Password

In this user selects six images for password according to his/her order.

Figure 5. Images Selected for Password

In this research, complete software was developed in which team A used Forward Engineering (FE) approach. Team B developed the software by using Forward Engineering Completeness (FEC) approach. Both approaches were evaluated by monitoring following types of errors/defects [11]: Incomplete or Erroneous Specification (IES), Misinterpretation of Customer Communication (MCC), Intentional Deviation from Specification (IDS), Inconsistent Component Interface (ICI), Miscellaneous (MIS). Total number of Correct Functionalities (CF) in the software were also counted. In this research only IES, MCC, IDS, ICI, MIS, were monitored, because these errors/defects were related to the introduced technique. Total number of functional and non-functional requirements in the software were 1398. Table 1 represents the basic software components. During the evaluation process following errors in the software were identified: *IES = 173, MCC = 122, IDS = 27, ICI = 42, MIS = 301, CF = 733* as shown in Fig. 6. All other errors/defects that does not belong to IES, MCC, IDS, ICI, were considered in the MIS. One value was assigned for one error/defect whether that error/defect belongs to functional or non-functional requirements of *1398* total requirements.

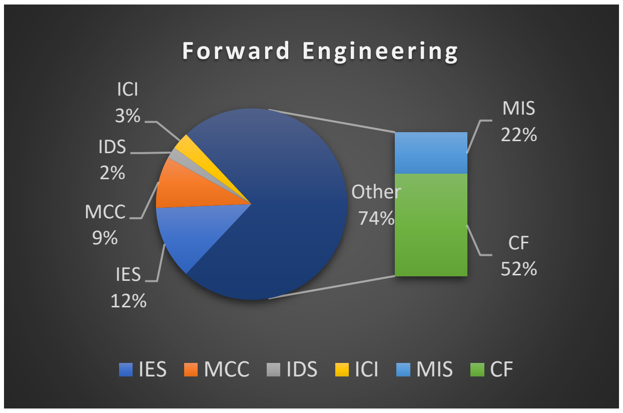


Figure 6. Errors & Correct Functionalities in FE Approach

In forward engineering completeness same type of errors/defects were monitored in the development process in order to determine the importance of requirement validation framework. Total number of functional and non-functional requirements were in the range of 1398 to 1430. The range in requirements were due to modifications in the unadopted requirements in order to fulfill different customer needs. Maximum value of requirements was assigned to the total requirements. During the evaluation process total number of errors in the software were as: *IES = 11, MCC = 14, IDS = 9, ICI = 21, MIS = 39, CF = 1336* as shown in Fig. 7.

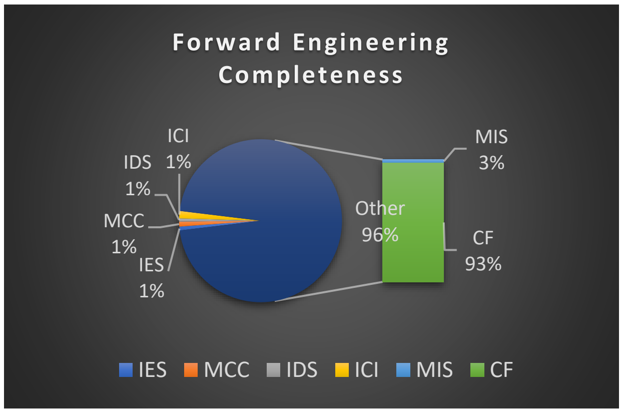


Figure 7. Errors & Correct Functionalities in FEC Approach

Team A developed the software without following the introduced techniques whereas team B followed the template of table for software requirements. After this these requirements were validated in the requirement validation framework before the actual development of software. Team A software development duration was more than the prescribed duration whereas team B developed the software in less than the prescribed duration. Total percentage of errors in software requirements was about 48% in forward engineering approach. In forward engineering completeness approach total percentage of errors in software requirements was about 7%. The decrease in errors was due to the validation of requirements before the software development. It has been observed that if development time of software increase, budget allocated for that software becomes less. As team A completed the software two months more than prescribed duration, so for these two months extra budget was used in order to fulfill the salaries requirements of employees and other expenses of software company. Software development time increases due to identification of errors and defects in software, if these are found in last phases of system development life cycle than more time is required to remove them. As team B developed the software by using requirement validation framework therefore the defects found in this were nearly negligible. From the Fig. 7 it has been observed that whenever unadopted requirements are identified at the start of software development, there was reduction in budget and time duration for development also reduces. Forty-five days were required by Team A to perform corrective, adaptive, perfective and preventive maintenance whereas Team B completed all maintenance types in one day.

IV. Conclusions

This research supported the convincing evidence that, whenever requirement validation framework was used in the forward engineering, then surplus budget allocated for the maintenance phase was saved. The suitability of attributes allows illustrating conclusion about how suitable software component was for a specific problem. The time required to develop the software was also reduced. The time spends on corrective, adaptive, perfective and preventive maintenance reduces approximately 1 to 2 months for one-year projects, whereas in normal routine it takes 2 to 3 times more than scheduled time. It has been observed that software size is increasing day by day due to change in technology and new requirements of end-users. As software size increases ultimately the software complexity also increases. In the final phase the software, size becomes like a pyramid so if user stories are ignored in the requirement gathering phase then large number of errors and defects are identified in the software. The requirement validation framework identified the unadopted requirement in the software and new requirement were also identified in the face to face meeting which resulted the software in completeness form. The identification of unadopted requirement saved the software engineers from complexity of errors and defects.

Acknowledgment

This work was supported by Chongqing Research Program of Basic Science & Frontier Technology with Grant No. cstc2017jcyjB0305.

References

[1] S. Lauesen, "Problem-Oriented Requirements in Practice -A Case Study", In REFSQ 2018, Utrecht, Netherlands, 2018, pp. 3-19.

[2] R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, "Quality requirements in industrial practice - an extended interview study at eleven companies," IEEE Trans. Softw. Eng. Vol. 38, 2012, pp. 923–935.

[3] R. Berntsson-Svensson, T. Olsson, B. Regnell, "An Investigation of How Quality Requirements are Specified in Industrial Practice", Inf. Softw. Technol. vol. 55, 2013, pp. 1224–1236.

[4] K. Wnuk, R. K. Kollu, "A Systematic Mapping Study on Requirements Scoping", In: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, Limerick, Ireland, 2016.

[5] P. Zave, M. Jackson, "Four Dark Corners of Requirements Engineering", ACM Trans. Softw. Eng. Methodol. 6(1), 1997, pp. 1–30.

[6] S. Hotomski, E. B. Charrada, M. Glinz, "An Exploratory Study on Handling Requirements and Acceptance Test Documentation in Industry", In: 24th IEEE International Requirements Engineering Conference, Beijing, China, 2016, pp. 116–129.

[7] G. Ana-Maria, O. Cristina-Claudia, A. B. Robert, "Security Requirements Elicitation from Engineering Governance, Risk Management and Compliance", In REFSQ 2018, Utrecht, Netherlands, 2018, pp. 283-289.

[8] S. Vishal, F. L. Xiaoqing, "Analysis of Conflicts Among Non-Functional Requirements Using Integrated Analysis of Functional and Non-Functional Requirements", In 31st Annual International Computer Software and Applications Conference, IEEE Computer Society, Beijing, China, 2007, pp. 215-218.

[9] M. K. Chawla, I. Chhabra, "A Quantitative Framework for Integrated Software Quality Measurement in Multiversions Systems," International Conference on Internet of Things and Applications, IEEE, Pune, India, 2016, pp. 310-315.

[10] M. Yan, X. Xia, X. Zhang, L. Xu, D. Yang, "A Systematic Mapping Study of Quality Assessment Models for Software Products," International Conference on Software Analysis, Testing and Evolution, IEEE, Harbin, China, 2017, pp. 67-71.

[11] R. S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill Education, New York, USA, 8th ed, 2014.

[12] M. Rouse, "Legacy Platform", Internet: https://whatis.techtarget.com/definition/legacy-platform-legacy-operating-system, March 2011, [February 22, 2019].

[13] D. Gustafson, Schaum's Outlines Software Engineering, McGraw-Hill Education, New York, USA, 1st ed, 2002.

[14] X. Lian, L. Zhang, "Optimized Feature Selection Towards Functional and Non-Functional Requirements in Software Product Lines", IEEE, 22nd International Conference on Software Analysis, Evolution, and Reengineering, Montreal, QC, Canada, 2015, pp. 191-200.

[15] B. Thalheim, Entity-Relationship Modeling: Foundations of Database Technology, Springer-Verlag, Berlin, Germany, 1st ed, 2000.

[16] G. D. S. Hadad, C. S. Litvak, J. H. Doorn, M. Ridao, Dealing with Completeness in Requirements Engineering, McGraw-Hill Education, New York, USA, 5th ed, 2015.

[17] M. K. Pour, "Encyclopedia of Information Science and Technology", IGI Global, USA, 4th ed, 2017.

[18] J. Noll, S. Beecham, I. Richardson, Global Software Development and Collaboration: Barriers and Solutions, ACM Inroads - Special Section on Global Intercultural Collaboration, 1(3), 2010, pp. 66-78.

[19] I. Sommerville, Software Engineering, Pearson Education Limited, London, England, 10th ed, 2015.