MATURITY REQUIREMENTS MODEL FOR SOFTWARE REQUIREMENTS WITH THE IMPLEMENTATION OF ISO/IEC 25010 RECOMMENDATIONS

Vasyl Yatsyshyn, Oleksandr Kharchenko, Andriy Lutskiv

Abstract*: The article is devoted to the construction and formalization of the model of requirement maturity based on Requirements Maturity Model (RMM). The RMM model has been modified by implementing the recommendations of the ISO/IEC 25010 standard, which allows to reduce the threshold of transition of companies from a lower level of maturity to a higher level at the stage of requirements development and analysis, as well as to ensure efficiency of requirements management process. Formalized representation of the modified model of maturity in the form of terms of set theory allows to automate the processes of communication of quality requirements at the stages of software development, and the means of automating the collection of requirements - to increase the criterion of completeness of requirements.*

Key words: *model, maturity, quality, software, requirements*

Introduction

Modern software engineering technologies make it possible to implement systems with a high degree of functional integration, ease of use by end users, performance and reliability. Ensuring such properties of the software requires the introduction of both formal methods of displaying the objects of the subject environment, and technological and instrumental means of supporting life cycle processes. The final quality of the developed software product directly depends on the maturity of the software engineering processes used by a particular developer. Therefore, determining the level of maturity of processes and product at the stages of the life cycle is an important and urgent task in the field of software engineering. Solving this problem involves substantiation or construction of models of maturity, in particular at the stage of definition and analysis of requirements, formalization of criteria for quality management and monitoring at the stages of the life cycle, creating procedures for their implementation in real projects.

Analysis of the current state of research

The maturity of software engineering processes is directly or indirectly studied by the scientific community of Ukraine [1-4] and foreign scientists [5-8]. As a result of such studies, models CMM [9], RMM [10], SMMM [11] and a number of others have been built, which determine the appropriate levels of process excellence and completeness of software development stages. These models allow to describe and evaluate a set of measures and resources of each stage of software implementation, to determine the level of maturity of the processes of a particular developer, which can be used by customers as a basis for decision-making when choosing a company.

One of the approaches that allows to significantly improve both the quality of software development processes and the quality of the final product is the implementation of the recommendations of the standards at the stages of the life cycle. In this context, it is advisable to apply the standards of the ISO / IEC 25000 series (ISO / IEC 25010, ISO / IEC 25030, etc.), but their effective implementation is hampered by factors such as imperfect formalized presentation of processes and quality assurance criteria, lack of strictly defined and standardized management procedures , monitoring and tracing requirements at the stages of the life cycle. As a result, it is almost impossible to achieve high maturity of process implementation and maturity of the software product itself. This study proposes the formalization of the requirements maturity model [10] with the integration of quality models of the standard [12], as well as a software module to collect software needs, which together will reduce the entry threshold for the development team and increase process maturity and performance at the analysis stage requirements.

Construction and formalization of the model of maturity of software requirements

The requirements maturity model (RMM) is given in [10] and consists of six levels, the structure and description of which are shown in Fig. 1.

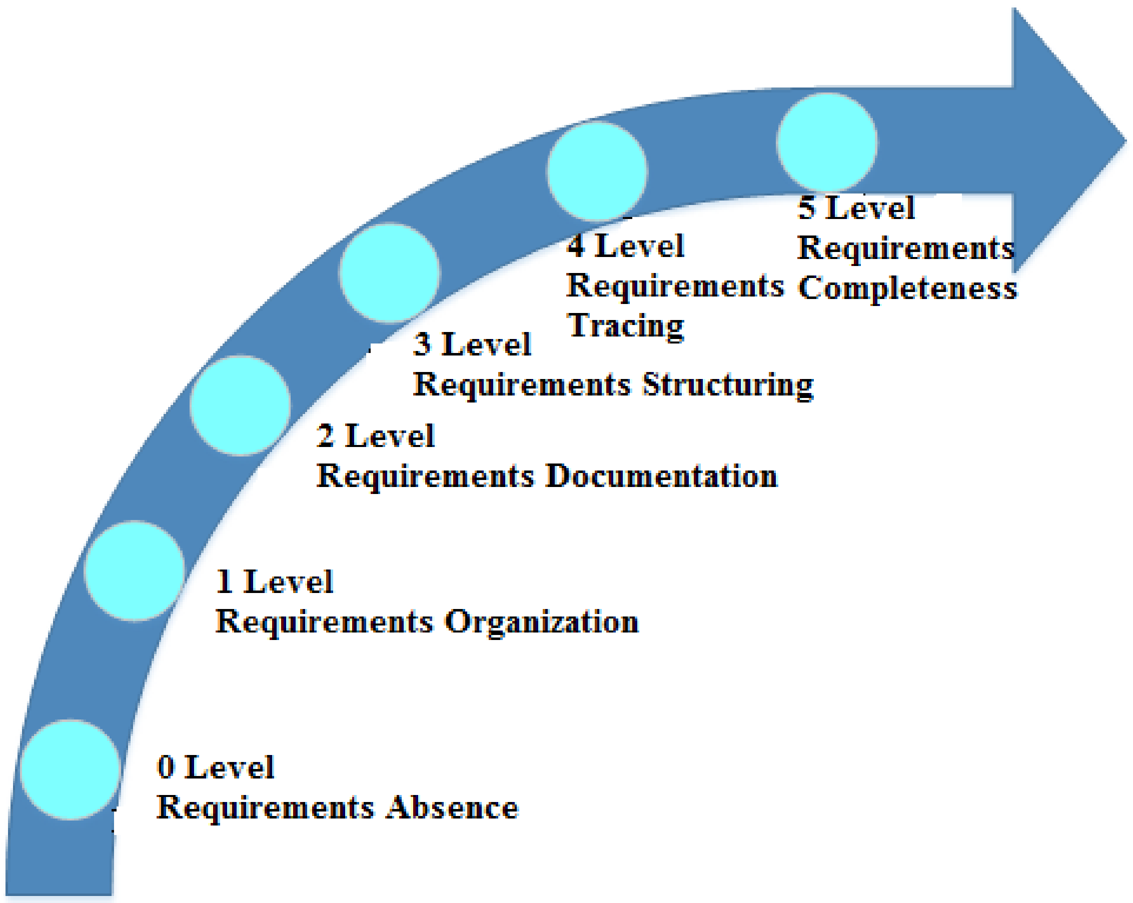


Fig.1. Requirements Maturity Model (RMM)

The software quality management process based on the RMM maturity model will allow developers to improve their process to meet the requirements step by step. At the same time, the improvement of the quality management process of requirements occurs at each subsequent step. In accordance with the concept of continuous quality improvement, presented in Fig. 1, after the transition to the next stage, developers must put into practice all relevant to the process of action.

Having conducted a detailed description of each level of the maturity model (Requirements Maturity Model) and the corresponding processes, we formalize the presentation of the maturity model of software requirements in the implementation of software in terms of set theory. In the general case, the model of maturity can be represented as:

$$RMM = \{L_N, \{L_{N-1}\{L_{N-2}\{\ldots\{L_0\}\ldots\}\}\}\} \quad (1)$$

where *L* is a level of the requirements model maturity,

N is a number of levels of the maturity model.

According to the reasonable model, the number of maturity levels is 6.

L_0 is a zero level of requirements maturity, which implies their absence in terms of chaos and the presence of only part of the customer's needs. There are no software requirements at this level.

To meet the needs of the customer or users of the computer system at this level, it is proposed to automate the collection process by connecting data sources such as Skype, E-mail, and other documents that describe the need to implement software. This will allow storing in a single database information about the needs of users in the software and thus allow you to quickly move to the second level of the model of maturity requirements.

Formally, the first level of the model of maturity of requirements can be represented as:

$$L_1 = \left\{Doc_i, ExpMark_i, \{L_0\}\right\}, i = 1..T \tag{2}$$

where *Doc* is a set of documents describing the software requirements,

ExpMark is the expert scores of documents,

T is a number of documents describing the requirements at the first level of the maturity model.

Software needs, in general and according to [13], can be represented as:

$$R_c = \{P_i, C_{ik}\}\ i = 1..I, K = 1..M_i \tag{2}$$

where P_i are user needs;

C_{ik} are constraints of needs;

I is a number of customer needs;

K is a number of constraints of needs.

The second level of the software requirements maturity model of computer systems is focused on refining software requirements by applying different methods and creating an appropriate database of requirements. It can be formally represented as follows:

$$L_2 = \left\{R_i^c, MClar_{ij}, RClass_{ik}, RVer_{im}, \{L_1\{L_0\}\}\right\} \tag{3}$$

where R_i^c are customer requirements for software, *i=1..B, B* is a number of requirements at the second level of the maturity model;

$MClar_{ij}$ are methods of clarification of requirements (brainstorming, questionnaires, clarifications, collective inspection, etc.), *j=1..S, S* is a number of methods for clarifying the requirements;

$RClass_{ik}$ are classes to which the requirements belong R_i, *k=1..K*, *K* is a number of classes of software requirements;

$RVer_{im}$ are versions of the software requirements to control changes to it, *m=1..M*, *M* is a number of versions of the requirement.

To classify the requirements, it is proposed to use two classes: functional and non-functional requirements:

$$RClass = \{Func, NonFunc\} \tag{3}$$

where *Func* is a class of functional requirements that can be represented in the form of templates;

NonFunc is a class of non-functional software requirements at the second level of the maturity model.

The third level of the model of maturity of software requirements can be represented as a set, the elements of which are:

- attributes of requirements;
- requirements templates;
- requirements models;
- checklists;
- recommendations.

Formally, the third level of the maturity model can be represented as follows:

$$L_3 = \left\{R_i^{use}, Pattern_{ij}, RModel_{ik}, RList_{im},, Recom_{im}\{L_2\{L_1\{L_0\}\}\}\right\} \quad (4)$$

where R_i^{use} are customer requirements in the form of a quality model in use, *i=1..G*, *G* is a number of quality requirements in use;

$Pattern_{ij}$ are patterns for describing the software requirements for computer systems, *j=1..Q*, *Q* is a number of patterns;

$RModel_{ik}$ are models for requirements describing, *k=1..P, P* is a number of requirements models;

$RList_{im}$ are checklists to check the requirements, *m=1..M, M* is a number of checklists;

$Recom_{im}$ are recommendations for improving the requirements, *h* is a number of recommendations.

Requirements model R^{use} of the business system user is proposed in [13]. It looks like:

$$R^{use} = \{H_i^u, A_{ik}^u, C_{ik}^u, M_{ik}^u\}, i \in N_u^k, K = 1..S_i \quad (5)$$

Representation of quality requirements in use in the form of model (5) ensures their formalization in standardized, unified terms. This, in turn, allows to adequately and fully reflect the needs of business system users, to avoid vague interpretations and "substitution of concepts", as well as greatly simplify the development of automation tools focused on supporting the technological processes of the early stages of LF. Based on the model (5), it is convenient to present the user requirements for the aircraft and then effectively manage the quality requirements, to ensure their variability, because they are structured.

At the fourth level of the maturity model, hierarchies of requirements are built, dependencies are established between them, standard solutions are determined, and so on.

For the formal presentation of the fourth level of the maturity model, it is proposed to use the external quality model of the ISO / IEC 9126 standard.

Formally, the fourth level of the maturity model of requirements is presented as follows

$$L_4 = \{R_i^{ext}, \{L_3\{L_2\{L_1\{L_0\}\}\}\}\} \qquad (6)$$

R_i^{ext} are requirements in the form of a model of external quality in [13] are presented as follows:

$$R^{ext} = \{H_i^x, S_i^x, A_{ik}^x, C_{ik}^x, M_{ik}^x\}, i \in N_x \;, K = 1..F_i^x \qquad (7)$$

Formalized presentation of external quality requirements allows to unambiguously and fully reflect the quality requirements in use, which meet the needs of the user and the customer of the aircraft. In addition, the formalization of requirements at the stage of their development ensures the accuracy and simplicity of further development of the project, in particular, it concerns the selection and construction of the future architecture of the aircraft. Therefore, the implementation of the proposed formalization of the model of external quality of the aircraft is quite relevant because it simplifies and automates this process, as well as scientifically justify the feasibility and practicality of the application of standards [13].

The fifth level of the requirements maturity model is characterized by the processes of tracing requirements for design and testing components, prioritizing requirements, quantifying labor cost requirements, and implementing requirements management tools and integrating with computer systems software development tools. Formally, the fifth level of the maturity model can be represented:

$$L_5 = \left\{R_i^{in}, \{L_4\{L_3\{L_2\{L_1\{L_0\}\}\}\}\}\right\} \tag{8}$$

Internal quality requirements R_i^{in} are used to determine the properties of intermediate states of the product. You can use static and dynamic models, technical documentation and code. Internal quality requirements can be used to determine the strategy for further development and can serve as criteria for evaluating and verifying the development process.

Formalized presentation of internal quality requirements R^{in} according to [13] has the form:

$$R^{in} = \left\{H_i^x, S_i^x, A_{ik}^y, C_{ik}^y, M_{ik}^y\right\}, i \in N_x \ , K = 1..F_i^y \tag{9}$$

Graphically, the model of maturity of requirements with the appropriate levels and processes is shown in Fig. 2.

Thus, a model of software requirements maturity is formalized, which will allow more effective implementation of requirements management and ensure their quality at a low entry threshold for the development team and the organization as a whole

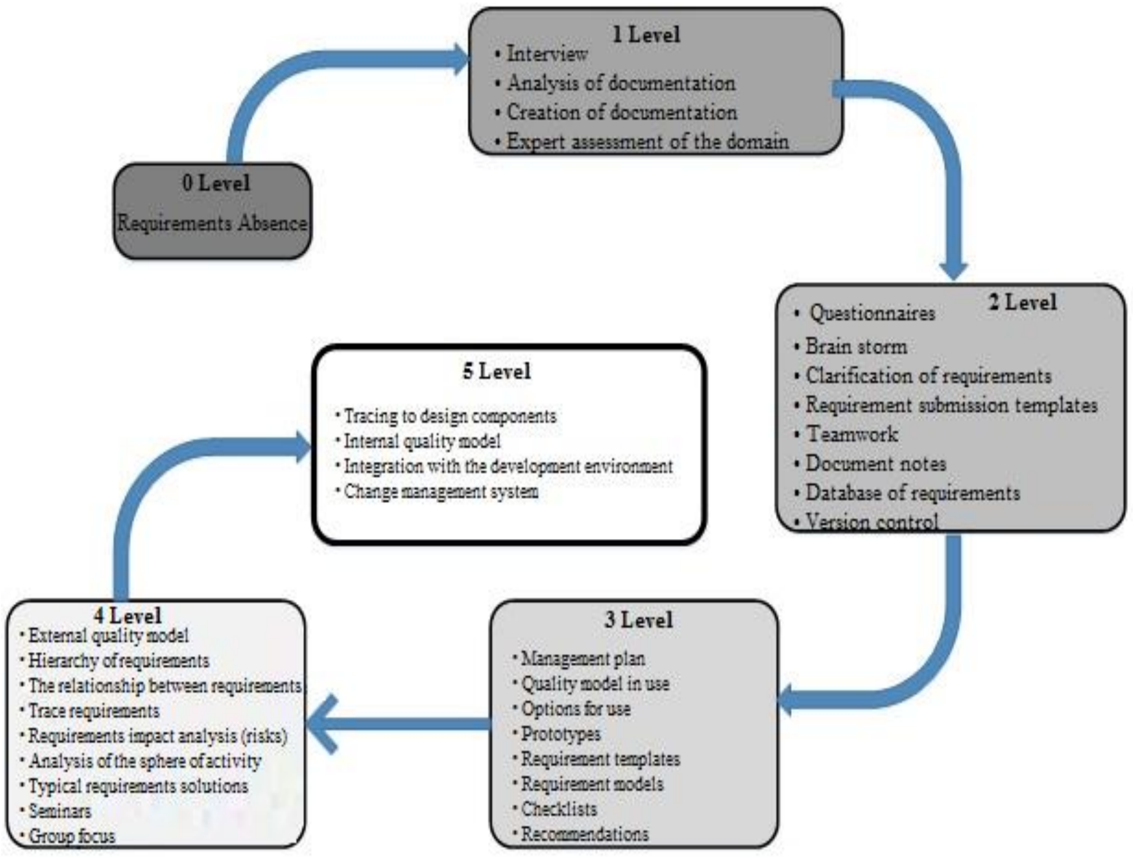

1 Level
• Interview
• Analysis of documentation
• Creation of documentation
• Expert assessment of the domain
0 Level
Requirements Absence
2 Level
• Questionnaires
• Brain storm
• Clarification of requirements
• Requirement submission templates
• Teamwork
• Document notes
• Database of requirements
• Version control
5 Level
• Tracing to design components
• Internal quality model
• Integration with the development environment
• Change management system
4 Level
• External quality model
• Hierarchy of requirements
• The relationship between requirements
• Trace requirements
• Requirements impact analysis (risks)
• Analysis of the sphere of activity
• Typical requirements solutions
• Seminars
• Group focus
3 Level
• Management plan
• Quality model in use
• Options for use
• Prototypes
• Requirement templates
• Requirement models
• Checklists
• Recommendations

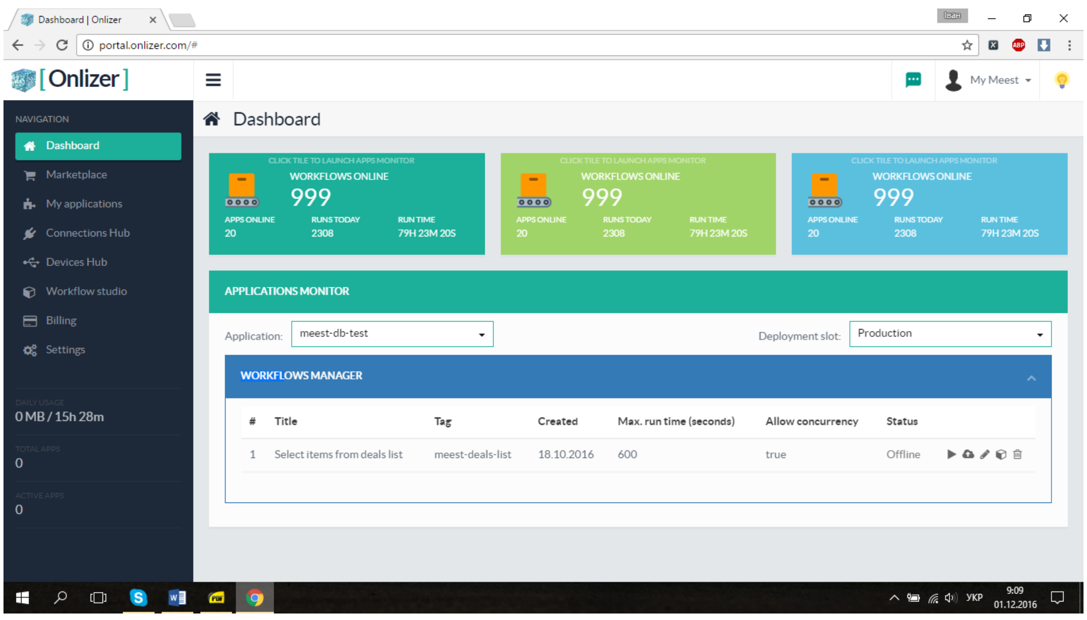
Fig. 2. RMM maturity model with implemented quality models of ISO / IEC 25010 standard

Automation of the process of requirements gathering at the second level of the maturity model

The most time-consuming process at the stage of determining and analyzing the requirements is their collection and structuring, which corresponds to the second level of the maturity model. Automation of this process is proposed to be implemented by creating and implementing an additional module to the quality management system [14].

The module for collecting software requirements and further integration with a sound CASE tool has been proposed to be developed on the basis of the Onlizer platform [16].

In order to start working with the Onlizer platform, it is needed to register on the website http://portal.onlizer.com/, and then log into personal account using the login form. If the authorization is successful, the main page of the system will be opened (Fig. 3), where it is possible to see the existing Applications, as well as the Workflows of this Application and their status.



Application: meest-db-test

Deployment slot: Production

#	Title	Tag	Created	Max. run time (seconds)	Allow concurrency	Status
1	Select items from deals list	meest-deals-list	18.10.2016	600	true	Offline

Fig. 3. Home page of the Onlizer system

Consider working with the Onlizer system on the example of a workflow that will collect requests from different messengers (Fig. 4).

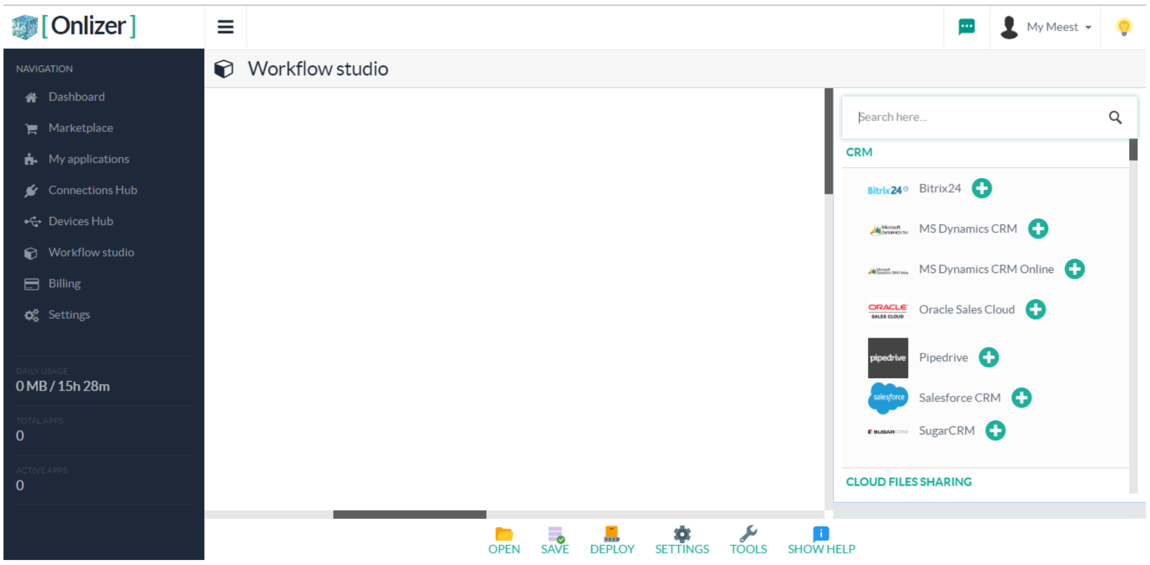


Fig. 4. Appearance of Workflow studio

To start, it is needed to create an application and workflow in the Workflow studio. Then on the right panel you need to select the connectors: HTTP Endpoint, Skype, Viber, Telegram and Slack, as well as the connector for storing information (user needs) in the MS SQL Server database.

The next step is to move the connectors to the work area, where they are configured (Fig. 5).

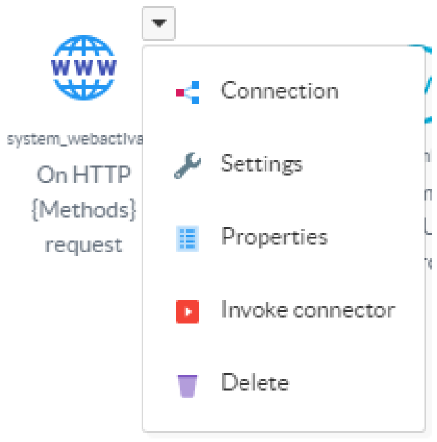


Fig. 5. Context menu of the connector

By adjusting the parameters of the connectors and the connections between them, it is obtained a workflow (Fig. 6), which provides a collection of requirements from certain potential sources of information.

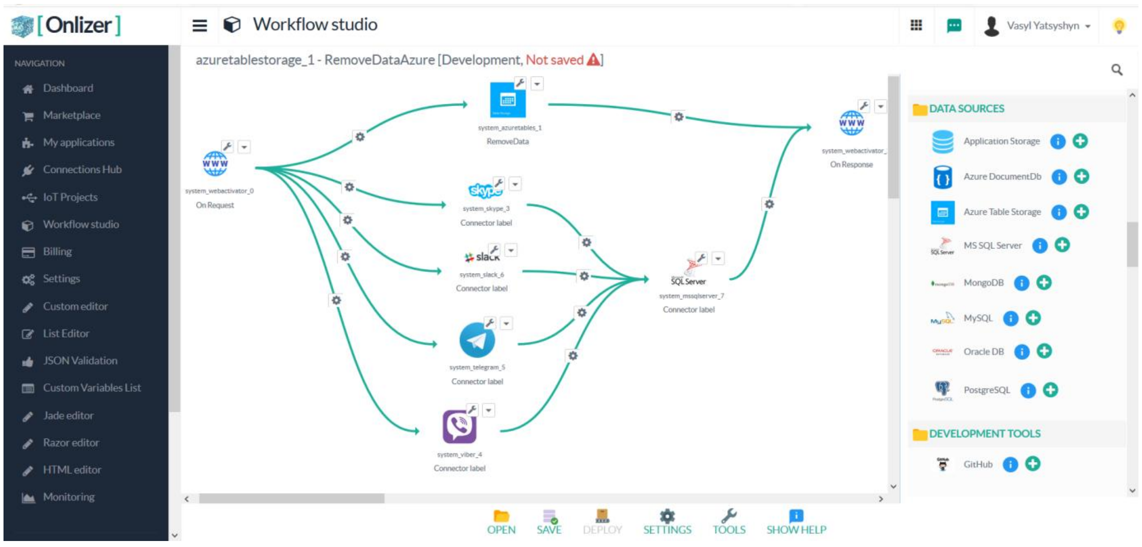


Fig. 6. Workflow to collect customer needs

When all the components are configured it is needed to click Save to save the workflow. To start the saved workflow is possible by clicking the Start button in the application settings window.

As a result, to obtain records in the database about the needs of a particular stakeholder in software, in the form of a field structure of the corresponding table. The fields of the table are:

- record identifier;
- type of means of communication;
- communication time;
- contact person from the customer;
- contact person from the developer;
- message content (or file link);
- recording time.

In this case, the integration of the developed requirements collection module and CASE requirements management tool based on the quality model is carried out through shared access to the database needs.

The architecture of the software requirements maturity support tool will look as shown in Figure. 7.

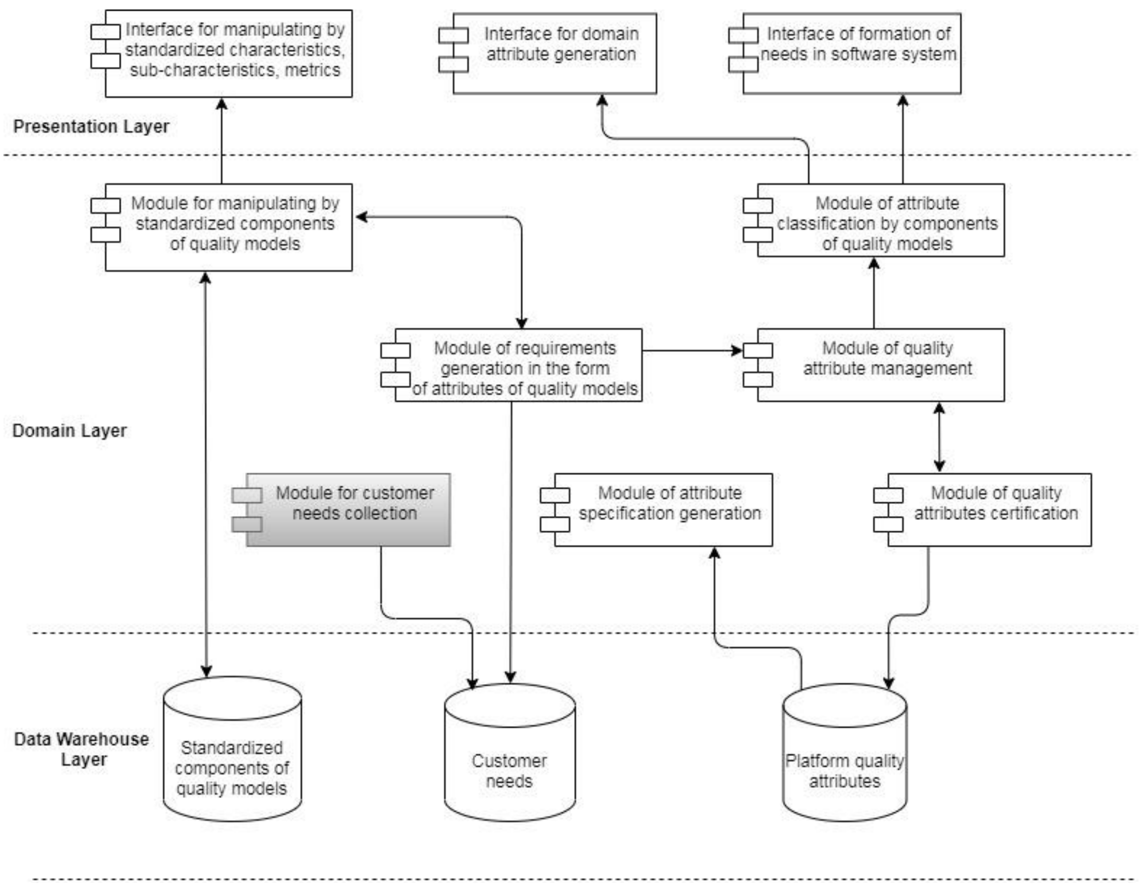

Interface for manipulating by standardized characteristics, sub-characteristics, metrics
Interface for domain attribute generation
Interface of formation of needs in software system
Presentation Layer
Module for manipulating by standardized components of quality models
Module of attribute classification by components of quality models
Module of requirements generation in the form of attributes of quality models
Module of quality attribute management
Domain Layer
Module for customer needs collection
Module of attribute specification generation
Module of quality attributes certification
Data Warehouse Layer
Standardized components of quality models
Customer needs
Platform quality attributes

Fig. 7. The architecture of the means of supporting the second level of the maturity model

The developed module for collecting customer needs from additional sources such as Skype, Viber, Slack, Telegram makes it possible to supplement the customer's needs in software and in the computer system as a whole. This module provides greater completeness of the requirements for the software to be implemented.

The integration of the developed module into the CASE tool for management and requirements development based on quality models is carried out on the basis of communication with the customer needs database located in the data warehouse layer. This module is completely isolated from the CASE tool and works as a service that can be run on schedule when establishing communication between the development team and the system customer. Payment for the use of the module is made only for the real time of its use.

Conclusions

The article substantiates and formalizes the requirements maturity model RMM (Requirements maturity model) with implemented quality models of ISO / IEC 25010, which allows to reduce the transition threshold of companies in the presence of chaos in the requirements for higher levels of maturity model and thus increase the efficiency of requirements management. .

Formalized representation of the modified maturity model allows to automate the processes of communication of quality requirements at the stages of software development, which ensures the maturity of this process.

The Onlizer platform has created a module for collecting requirements from messengers, which are used in the interaction of members of the development team and customer representatives, which is also an element of increasing the maturity of processes at the second level of the RMM model. The developed module is at the same time a means of automation, which allows to increase the quality criterion of requirements for their completeness, and thus improve the quality of the final product.

References

1. Вершина А., Семерюк Т., Солдатов Б. Модель процесса разработки программного обеспечения. Пробл. програмув. 2006. № 2-3 [спец. вип.]. С. 269-274.
2. Paulk M., Curtis B., Chrissis M., Weber Ch. Capability Maturity Model. IEEE Software.1993. Vol. 10. 4. pp. 18–27.
3. Jessyka Vilela, Jaelson Castro, Luiz Eduardo G. Martins, Tony Gorschek, Safety Practices in Requirements Engineering: The Uni-REPM Safety Module, IEEE Transactions on Software Engineering, 10.1109/TSE.2018.2846576, 46, 3, (222-250), (2020).
4. Keefer G., Lubecka H. The CMMI in 45 minutes. Stuttgard: AVOCA GmbH. 2005. p. 10.
5. Paulk M. A Comparison of ISO 9001 and the Capability Maturity Model for Software. Software Engineering Institute. Pittsburgh (USA). 1993. p. 78 .
6. Yaseen M., Ali Z. Requirements Management Model (RMM): A Proposed Model for Successful Delivery of Software Projects. International Journal of Computer Applications. 2019. 178(17):32-36
7. CMMI Levels and Requirements Management Maturity Introduction. URL: http://tynerblain.com/blog/2007/01/25/cmmi-and-rmm-intro (дата звернення 05.07.2019)
8. Харченко О., Яцишин В. Розроблення та керування вимогами до програмного забезпечення на основі моделі якості. Вісник ТДТУ. Том 14, No 1. 2009. С. 201-207.
9. Kharchenko A., Galay I., Yatcyshyn V. The method of quality management software. VII International Conference on Perspective Technologies and Methods in MEMS Design: Proce-edings. Polyana (Ukraine). 2011. pp. 82–84.
10. Yatsyshyn V, Medvid I, Pundyk V. Using Onlizer as efficient and productive tool at the software life cycle stages. VI Międzynarodowa konferencja studentów oraz doktorantów „Inżynier XXI wieku". Wydawnictwo

naukowe akademii techniczno-humanistycznej w Bielsku-Białej. 2016. pp. 201-209.

11. Kharchenko A., Bodnarchuk I, Yatcyshyn V. The method for comparative evaluation of software architecture with accounting of trade-offs. American Journal of Information Systems. 2(1). 2014 pp. 20 – 25.
12. The Requirements Maturity Model Explained. URL: https://www.iag.biz/about-iag/requirements-maturity-model-explained (дата звернення 05.07.2019)
13. Яцишин В.В., Харченко О.Г. CASE-технологія розроблення вимог до програмного забезпечення та оцінювання його якості. Науковий вісник НЛТУ України. Вип. 20.2. 2010. С. 277-285.
14. ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.
15. ISO/IEC 25030:2019 Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Quality requirements framework

Authors' Information

Vasyl Yatsyshyn *– Candidate of Technical Sciences, Associate Professor, Associate Professor of Computer Systems and Networks Department of the Ivan Pulyuy Ternopil National Technical University. Box: 46000, Ternopil, Ukraine; e-mail: kaf_ki@tu.edu.te.ua*

Major Fields of Scientific Research: Software Quality

Oleksandr Kharchenko *– Candidate of Technical Sciences, Associate Professor, Associate Professor of Information Technologies Department of the National Aviation University. Box: 03058, Kyiv, Ukraine; e-mail: kit.kharchenko@nau.edu.ua*

Major Fields of Scientific Research: Software Quality

Andriy Lutskiv *– Candidate of Technical Sciences, Associate Professor, Associate Professor of Computer Systems and Networks Department of the Ivan Pulyuy Ternopil National Technical University. Box: 46000, Ternopil, Ukraine; e-mail: kaf_ki@tu.edu.te.ua*

Major Fields of Scientific Research: Software Engineering