УДК 681.518.658.512

M. V. TKACHUK, R. A. GAMZAEV, I. O. MARTINKUS, S. D. IANUSHKEVYCH

METHODS AND TOOLS FOR DYNAMIC REQUIREMENTS CATALOG MANAGEMENT IN AGILE SOFTWARE DEVELOPMENT

A method for managing dynamic requirements catalog in agile software development, especially on example of Scrum-methodology is proposed. Popular approaches to solving this problem are reviewed. The proposed approach is based on the combined usage of the latent semantic analysis and analytical hierarchy process, it allows to evaluate the given textual software specification with respect to their possible redundancy and possible logical conflicts. Besides that this approach supports the decision making procedure to prioritize the requirements taking into account their functionality importance for target software product. The effectiveness of the proposed method was tested on the test case.

Keywords: Agile, latent semantic analysis, dynamic catalog of requirements analytic hierarchy process.

Introduction: Problem Actuality and Research Goal. Requirements management (RM) is one of the most important and weak-formalized disciplines in the modern software engineering (SE), because of permanent changes in an application business logic and /or in a projects configuration. This is the real challenge for any new software system to be designed "from scratch" or for already existing software systems. To resolve these problems the several adaptive SE-methodologies called as agile software development (ASD) are proposed [1]. The main concepts of any ASD-methodology supposed to meet requirement changes in a following way:

(1) organization of a permanent and closed collaboration between several actors involved in ASD: stakeholders, analysts, developers, end-users, etc.;
(2) usage of an iterative approach for the project development;
(3) providing a requirements traceability in order to reflect all requirement changes into appropriate projects artifacts.

In all ASD-methodologies, taking into account their base principles (1)–(3) is supposed, that the management of permanent changed in software requirement specifications (SRS) should be provided in an adaptive mode. It means necessity in appropriate way quickly to change the design solutions, development environment, the organization forms of a developer team, etc. Such dynamic and multidimensional vision for Requirements management is well shown on qualitative model of this process proposed by S. Ambler [2], which is shown below in Fig. 1.

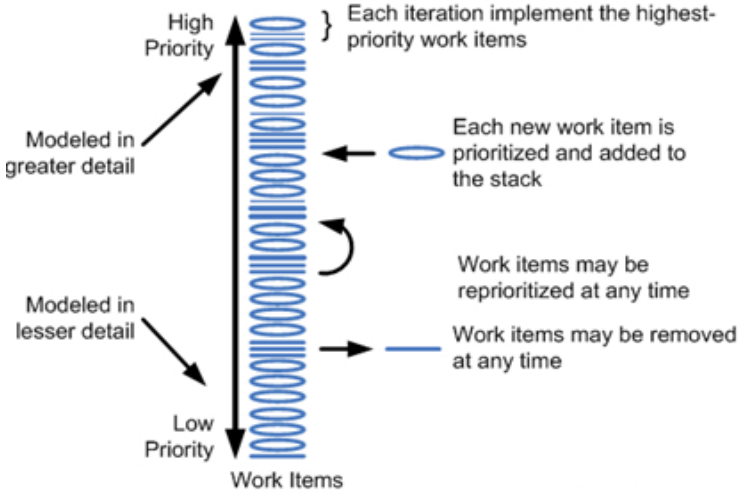

Figure 1 – Conceptual scheme for adaptive requirements management by S. Ambler [2]

It is necessary to note that besides some empirical recommendations concerning the problem how to build such a dynamic requirements catalog (DRC), in recent literature there are no more or less proved suggestions about possible quantitative effectiveness estimations for this purpose.

This problem is actual also in a broader sense, because DRC is a high-level requirements representation for any domain, independently from their specification method with any notation (UML, IDEF0 etc.). So DRC is essentially an example of such important concept in Agile methods as initial requirements[3], which are required for building on their base Domain Model, which in turn is basis for such modern and effective software development approach as **Domain Driven Development (DDD)** [4]. But in [3] don't proposed any quantitative methods and evaluations as for priority determination of particular requirements, as well as solving of such important tasks, as the elimination of logical contradictions and data duplications in text specifications.

That is why the main research aim of this paper is to consider some specific features of requirements management in ASD, to analyze available formal methods for requirements evaluation and prioritization, and to

propose the complex of algorithms and software tools, which can be used for DRC processing.

Specific features of requirements management in Agile-software development on Scrum-methodology example. As already mentioned above, ASD methodologies are widely used due to new nature of software projects and due to permanent changes in the SRS. ASD implies refusal from a huge of project documentation, but really needs support for automated of stakeholders knowledge processing, in order to reduce the time for adaptation to requirements changes and to minimize risks related to them.

Exactly because of these reasons in the modern SE-concepts process control methods called also as "software cybernetics" (e.g. in [5]) are used. On Fig. 2 the cybernetics-centered scheme is shown [6], which represent one of the most used ASD-methodology, namely Scrum method [1, 2]. There are 2 feedback loops included in this control scheme: (1) daily process control loop, which is organized basing on sprint backlog (SB) and using some source code quality metrics, (2) iteration process control loop, which is provided basing on product backlog (PB) and using some requirements quality metrics. The SB and PB both collect the selected requirements to be met in final software product. In the control loop (1) usually some IDE, e.g. the Eclipse is used by developers team, while in the control loop (2) an appropriate requirements management system (RMS) can be exploited by stakeholders (or Product owner – in Scrum project) and by domain analysts.

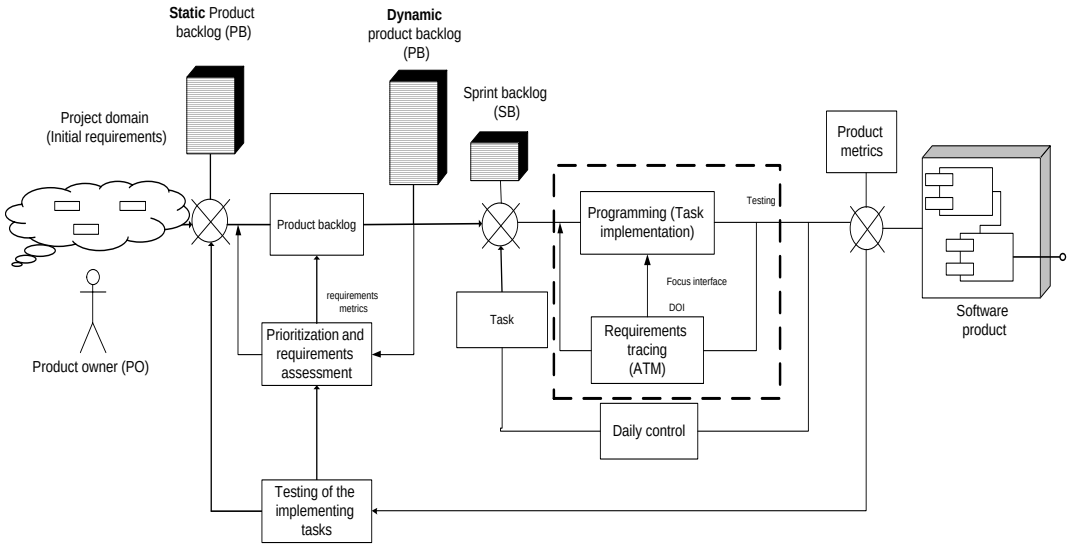

Figure 2 – The cybernetic-centered scheme of a Scrum-methodology

In [6] the approach to creation so-called advanced traceability matrix is elaborated, which allows to take into account all developers activities in time-oriented data model, and the appropriate CASE-tool designed to combine the features of typical RMS and the functionality of integrated development environments (e.g. Eclipse). But the problem how to form and to maintenance an appropriate DRC is still open, and below we propose methods and software tools for this problem solution.

Overview of the text specifications analysis and prioritization methods for the composition of the dynamic catalog of requirements. Current approaches to the treatment of software requirements formulated in natural language can be divided into 3 main groups:

1. The construction of formal models for the original text descriptions of requirements, using the algebraic notation or the apparatus of predicate logic ([7]), for subsequent qualitative analysis and automatic verification;
2. Development and application of different methods for building automated visual, structural and graphical forms of presentation of software requirements: e.g. in the form of ER-diagrams and UML-diagrams, based on their initial descriptions of natural language etc., this group includes many works such as [8];
3. Various methods of logical and linguistic analysis of the semantic features of natural language texts of specifications that are used to extract hidden knowledge, establish semantic dependencies, eliminate logical contradictions, etc. [9].

Obviously, considering the properties of agile software development in general, and the scheme of the Scrum-project in particular, out of the above methods for processing requirements text descriptions, in the context of this work the methods of the third group should be considered, as no formal requirements specification (covered by the Group 1 methods), or their diagrammatic documentation (covered by Group 2 methods) constitute essential prerequisites (conditions) for the successful implementation of agile-projects [10–11].

In the selected group of methods for processing text requirements, taking into account the specific tasks in the control loop of the process of forming the dynamic catalog of requirements DRC, for the solution of the first of them – namely, for eliminating possible logical inconsistencies and redundancy (duplication) of data – different data mining methods [12], the methods of the analysis of hypertext links [13], and the various modifications of the method of latent semantic analysis (LSA) [14] can be effectively applied.

The LSA method is a statistical method for processing natural language texts, the purpose of which is to establish the relationships between documents that are the part of a collection (case), and a set of some terms contained in these documents. It is possible to identify the thematic proximity of certain terms, which can then be used to calculate statistical estimates of the thematic proximity of the entire documents. As the input data the LSA method uses so-called "term-to-documents" matrix ("term" refers to a specific lexical unit), which reflects the frequency of occurrence of the particular term in the text of the particular document. Each column of the matrix refers to the certain document, and each line – to the certain term. Various approaches (metrics) are used for defining the value of a certain matrix element on the intersection of the *i*-th row and *j*-th column. One of such metrics is so-called TF-IDF (Term Frequency – Inverse Document Frequency) – the metrics which is used for the estimation of the importance of the term for the certain document from a collection [15]. To solve the second problem of the initial processing of the text description of requirements in the control loop of forming the dynamic catalog – namely, to prioritize the requirements, we first should take into account some methodological features of this process in the context of the agile methodologies of software development. It should be noted that in many works concerning the characteristics of the requirements engineering in agile methodologies it is directly stated that the creation and use of the appropriate mechanisms for prioritization and reprioritization of requirements is a key factor in the success of the projects making use of the agile methodologies. At the same time, in the agile projects (XP, Scrum, ASD), there are characteristic differences from the traditional methodologies (cascade, spiral, etc.) in the requirement prioritization process (RPP), these differences are analyzed in detail in [15], and are summarized in table.1.

Table 1 – The characteristic properties of RPP in the agile methodologies [15]

RPP characteristics	Traditional methodologies	Agile methodologies
When RPP is carried out	As a rule, once, after the analysis stage and before the implementation stage	Before each design iteration, or even during the iteration
Who initializes / leads RPP	Software developers, with the assistance of the manager of the project and other stakeholders	The software customer with the help of the manager of the agile development group (Scrum master)
Goals / prerequisites of carrying out RPP	The control for the process of carrying out the project	Support of the max. business significance of the results of software development / more precise specification of the amount of works of the current iteration of the project

The presence of all these factors leads to the fact that for RPP various expert methods are generally used, a fairly complete overview of which is given in [15]. Among them the following methods can be considered:

1. The method of the direct pairwise analysis – is carried out by pairwise comparing of separate requirements until the requirement with the highest priority appears at peak of their initiating list; owing to it this approach is applicable only in projects with a small amount of well-structured and plain text descriptions of requirements;
2. Wiegers matrix approach – is based on representing the set of software functional requirements in the form of the special table (matrix), the elements of which represent estimations, and are directly proportional to their technical significance for the final software products and inversely proportional to the cost and possible risks of their implementation; as a result of the analysis of such matrix it is possible to select the requirements with a maximum priority taking into account a compromise choice based on the values of the corresponding elements of this matrix;
3. The method of creation of a binary search tree – represents the approach based on a well-known method of binary search from the area of information search which can be adapted to the tasks of the prioritization of requirements, as a result of it the initial list of requirements can be represented in a way when the requirements with maximal and minimal priorities will be positioned in the ordered list.

Among the other methods it is important to mention the analytic hierarchy process [16]. This method will be used in this paper to prioritize DRC requirements after LSA preprocessing.

The complex procedure for constructing the dynamic requirements catalogue. The corresponding algorithms implementing its separate stages are given in more details below.

LSA requirements processing algorithm.

Step 1: Forming the matrix of the occurrence frequency for the *i*-th term in *j*-th requirements text:

$$\boldsymbol{M}[x_{ij}] = \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots \\ x_{m1} & \dots & x_{mj} & \dots & x_{mn} \end{bmatrix}, \quad (1)$$

where x_{ij} – is the occurrence frequency of the *i*-th term in the *j*-th requirement text.

Step 2: Removal of the columns in the matrix $\boldsymbol{M}\left[x_{ij}\right]$ where the term is included in only one requirement, resulting in reduced matrix $\boldsymbol{M}^{i}\left[x_{ij}\right]$ with dimensions $m \times n$.

Step 3: Creating the reduced matrix $\boldsymbol{M}^i[x_{ij}]$ by means of a method of singular expansion [11], i.e. to present a matrix in a type:

$$\boldsymbol{M}' = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T, \quad (2)$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the orthogonal matrixes.

Step 4. By selecting *k* largest singular values w_j, the approximation of an initial matrix by the matrix of the smaller rank *k* can be made. Performing the procedure of the rank reduction allows to eliminate surplus information, the rank value *k* is defined heuristically [13]. Matrixes are displayed as follows:

$$\boldsymbol{W}' = \begin{bmatrix} w_1 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & w_j & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & w_k \end{bmatrix}, \quad (3)$$

$$\boldsymbol{U}' = [[u_1]\dots[u_k]], \quad (4)$$

$$\boldsymbol{V}' = \begin{bmatrix} [v_1] \\ \dots \\ [v_k] \end{bmatrix}. \quad (5)$$

Step 5: Obtaining the final matrix $\boldsymbol{A}$ with the frequencies of the occurrence of *i*-th term in *j*-th requirements description that contains only *k* first linearly independent elements:

$$\boldsymbol{A} = \boldsymbol{U}'\boldsymbol{W}'\boldsymbol{V}'^T. \quad (6)$$

In contrast to the original matrix $\boldsymbol{M}$, the resulting matrix $\boldsymbol{A}$ is not sparse [13], which allows calculating dependencies between requirements documents even if the documents have no common terms.

Step 6: For finding dependencies between texts of different requirements on the basis of the obtained matrix **A** it is necessary to calculate the Piersons correlation [17] between requirements documents:

$$R = [r_{ij}], \text{ for } \forall i > j,\ i, j = \overline{1,n} \text{ and } r_{ij} = 1 \text{ if } i = j.$$

As a result of the given step the triangular matrix R containing correlations coefficients is obtained:

$$\boldsymbol{R} = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & r_{xy} & \dots & 1 & 0 \\ r_{1n} & \dots & r_{xn} & \dots & r_{nn-1} & 1 \end{bmatrix}. \quad (7)$$

The values of the correlation coefficients $\boldsymbol{r_{xy}}$ between two documents $\boldsymbol{x}$ and $\boldsymbol{y}$ are calculated on the basis of the data received in 8. Denoting pairwise the compared columns of matrix $\boldsymbol{A}$ as vectors $\boldsymbol{x} = \{x_i\}$ and $\boldsymbol{y} = \{y_i\}$, we receive:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}},\ r_{xy} \in [-1;1] \quad (8)$$

where: $i = \overline{1,N}$, $j = \overline{1,N}$, $\bar{x}, \bar{y}$ – average value of samples $\bar{x} = \frac{\sum_{1}^{n} x_i}{n}$ and $\bar{y} = \frac{\sum_{1}^{n} y_i}{n}$.

Step 7: Determining the critical value of the Pearson correlation coefficient $\boldsymbol{r_{cr}}$ for a given level of signify-cance. Here the positive value of correlation coefficients between the requirement *x* and *y*, $r_{xy} > r_{cr}$ which exceeds the positive critical value $r_{cr} > 0$, means the existence of the contradiction in the description of these requirements. The negative value of the correlation coefficients between the requirement *x* and *y*, $r_{xy} < -r_{cr}$ means that these requi-rements have redundancy in their description, and one of the requirements can be removed.

As a result of the execution of this algorithm (possibly, in multiple iterations) it is possible to determine all requirements whose descriptions contain logical contradictions and overlapping texts.

AHP-based requirements processing algorithm.

1 stage: generate criteria which will be used for an estimation of the requirements importance.

2 stage: generate a matrix of pairwise comparison of the importance criteria, we denote it as $\boldsymbol{M}[x_{ij}]$ with dimensions $m \times m$, where m is the number of criteria. As an estimation of the importance measure we will use score estimations. The resulting matrix by definition should possess the following properties:

All diagonal elements are equal to 1

All remaining elements possess the property of reverse symmetry, i.e. the values of matrix elements symmetric to the main diagonal are reverse values [16].

3 stage: to generate *m* reverse-semantic matrixes with dimensions $n \times n$ where *m* – the number of criteria, *n* – the number of requirements. The matrixes' values are estimations of pairwise importance of each requirement for each criterion:

$$\boldsymbol{A_k} = [a_{ij}],\ k = \overline{1,m};\ i, j = \overline{1,n}. \quad (9)$$

4 stage: For $\boldsymbol{M}$ and $\boldsymbol{A_k}$ matrixes obtained at the previous steps calculate local priorities of criteria by the formulas (10) and (11) respectively:

$$L_i = \sqrt[n]{\prod_{j=1}^{n} x_{ij}},\ i = \overline{1,n} \quad (10)$$

$$AL_i^k = \sqrt[n]{\prod_{j=1}^{n} a_{ij}^k},\ i = \overline{1,n} \quad (11)$$

where x_{ij} – an element of matrix $\boldsymbol{M}$;

a_{ij}^k – elements of a matrix $\boldsymbol{A_k}$;

$\prod_{j=1}^{n} x_{ij}$ and $\prod_{j=1}^{n} a_{ij}^k$ – product of all elements in *i-th* line.

5 stage: For L_i and AL_i^k values obtained at the 4th step carry out the normalization of the local priority criteria for all pairwise comparisons, for this purpose it is necessary to calculate local priorities of criteria by the formula (12):

$$L_i = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}^k},\ i = \overline{1,m}, \quad (12)$$

$$AL_i^k = \frac{a_{ij}^k}{\sum_{i=1}^{m} a_{ij}^k},\ i = \overline{1,m}. \quad (13)$$

On the base of the normalized calculated values it is possible to form matrixes ***L*** and ***AL*:**

$$\boldsymbol{L} = \left\| 1_i \right\|,\ i = \overline{1,m} \quad (14)$$

$$\boldsymbol{AL} = \left\| AL_j^k \right\|,\ j = \overline{1,n},\ k = \overline{1,m}. \quad (15)$$

6 stage: calculate global priorities, as matrix products – the formula (16):

$$\boldsymbol{G_j} = \boldsymbol{AL} \times \boldsymbol{L},\ j = \overline{1,n}. \quad (16)$$

The matrix $\boldsymbol{G_j} = \{w_1,\ldots,w_n\}$ contains coefficients defining requirements priorities.

As a result, by means of AHP algorithm the weight coefficients defining the importance of the requirements based on the selected criteria have been obtained.

The instrumental complex for implementing the procedure of constructing the dynamic requirements catalogue and the test calculations results. As the frequency of solving the task of constructing and modifying the dynamic requirements catalogue is not real-time, but assumes the possibility of performing the analysis and the formation of such requirements set synchronously with the planning of the next iteration of the agile-project, i.e. in case of Scrum-methodology its execution frequency corresponds to the period of 2-4 weeks, for the implementation of the corresponding procedure it is possible to use already existing software tools, such as MS Ecxel from the MS Open Office [18] tool suite and Matlab math package [19].

The input data in this schema are unprocessed initial requirements to the developed software product. Requirements are received from the product owner who forms the list of requirements and highlights keywords for each requirement. The developed application allows to form an occurrence matrix of keywords which forms the basis for calculation of correlation coefficients between requirements. The MatLab environment is used for this purpose, the data is imported from Excel file into this environment.

For carrying out the numerical experiment the table of requirements and the highlighted keywords has been made, see table 2.

Table 2 – The list of requirements and keywords from the terms dictionary

№	The requirement (the initial text)	Keywords	Id.
1	«The automated calculation system (ACC) should be expected to be used for processing the data about corporate customers, physical persons, legal bodies».	«The automated calculation system», "customers", "physical persons", "legal bodies".	*r1*
2	“The capability of working with groups of clients, supporting functions" friends "and" family "and the cost-based user groups that are grouped based on arbitrary attributes, adding / removing subscribers in a group as the operator and authorized person	"Customer", "add/remove", "subscribers".	*r2*
3	«Creating complex tariff plans with the participation of services that belong to different personal accounts».	"Tariff plan", "Personal account".	*r3*
4	«Convenience of tariff plans creation, a visual programming environment».	"Tariff plan", "Programming environment".	*r4*
5	«Formation of the new, modification and removal of existing tariff plans by ACC operator».	"Tariff plan", "modification", "removal".	*r5*
6	«Explicit representation of tariff plans in automated ACC workplace for different types of ACC users».	"Tariff plan", “Types of users”.	*r6*
7	«Creation of tariff plans of arbitrary complexity (including in the TP an unlimited number of tariff zones)».	"Tariff plan", “Tariff zone”.	*r7*

The calculation final results of Pierson mutual correlation are shown in table 3. For a significance level *rcr* = 0,874 values that have the correlation above critical, are of interest for the further analysis. As correlation between *r1* and *r6, r3* and *r4, r3* and *r7* is above critical value these requirements have been analyzed and the conclusion was made about the data redundancy in their description. So the requirements *r4, r6* and *r7* have been excluded from the log of requirements. After this, for each criterion, the pairwise comparison of the requirements importance has been performed, their local priorities, and the normalized priorities have been calculated, and then the generalized global priority has been calculated. The results are shown in table 4.

Table 3 – Pierson correlation

r1	1					
r2	-0,423	1				
r3	0,362	0,430	1			
r4	0,362	0,430	0,897	1		
r5	0,507	0,557	0,640	0,640	1	
r6	0,936	-0,092	0,485	0,485	0,774	1
r7	0,362	0,430	0,919	0,904	0,640	0,485

Table 4 – Result of LSA application

	Criteria priorities	*r1*	*r2*	*r3*	*r4*
Importance for the customer	0,615	0,263	0,203	0,267	0,267
Development time	0,292	0,311	0,311	0,198	0,180
Development complexity	0,093	0,266	0,263	0,276	0,195
The generalized priority		0,277	0,240	0,248	0,235

As a results of calculation we formed table 5 which contains requirements in decreasing order of their priority. Thus, as a result of applying LSA and AHP methods with a complex procedure proposed earlier (fig. 2 in the previous section of this paper) the dynamic catalogue of requirements is formed, the requirements are then transferred then into adaptive tracing control loop.

Conclusions and Future Work. In this paper some existing approaches to requirements assessment and prioritization were e analyzed, and the appropriate methods and software tools were developed to management dynamic requirements catalog (DRC) management in agile software development, especially on example of Scrum-methodology. The proposed approach is based on the combined usage of the latent semantic analysis and analytical hierarchy process, was allows to evaluate the given textual software specification with respect to their possible redundancy and possible logical conflicts. Besides that this approach supports the decision making procedure to prioritize the requirements taking into account their functionality importance for target software product. Our future work concerns final implementation of the proposed CASE-tool, as well as improving the algorithms for DRC management using other experts estimation methods in order to make its more precise and efficient.

Table 5 – Method AHP results

№	The requirement (the initial text)	Keywords	Id.
1	«The automated calculation system (ACC) should be expected to be used for processing the data about corporate customers, physical persons, legal bodies».	«The automated calculation system», "customers", "physical persons", "legal bodies".	*r1*
2	«Creating complex tariff plans with the participation of services that belong to different personal accounts».	"Tariff plan", "Personal account".	*r3*
3	"The capability of working with groups of clients, supporting functions" friends "and" family "and the cost-based user groups that are grouped based on arbitrary attributes, adding / removing subscribers in a group as the operator and authorized person	"Customer", "add/remove", "subscribers".	*r2*
4	«Formation of the new, modification and removal of existing tariff plans by ACC operator».	"Tariff plan", "modification", "removal".	*r5*

Bibliography: **1.** *Anderson D. J.* Agile Management for Software Engineering / *D.J. Anderson* // Prentice Hall, 2003. – 336 p. **2.** *Ambler S.* Agile Architecture: Strategies for Scaling Agile Software Development / *S. Ambler* // Web http://agilemodeling.com/essays/agileArchitecture.htm **3.** *Ambler S.* Agile/Evolutionary Data Modeling: From Domain Modeling to Physical Modeling / *S. Ambler* // Web http://agiledata.org/essays/agileDataModeling.html#DisasterStrikes **4.** *Evans E.* Domain-Driven Design: Tackling Complexity in the Heart of Software / *E. Evans.* – Addison-Wesley. – 2003. – 359 p. // **5.** *Hong Xu.* Requirement process establishment and improvement from the viewpoint of cybernetics / *Hong Xu, Pete Sawyer, Ian Sommerville.* // The Journal of Systems and Software, issue. – 2006. – № 79. – P. 1504–1513. **6.** *Tkachuk M. V.* Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development / *M. V. Tkachuk, R. O. Gamzayev, H. C. Mayr, V. O. Bolshutkin* // Проблемы программирования. – К.: НАН України. – 2012. – № 2–3 (спец. выпуск). – С. 252–260. **7.** *Селяков Е. Б.* Методика выбора требований при проектировании АСУТП на основе логико-формальной модели системы требований / *Е. Б. Селяков* // Вісник Донецького національного університету, Сер. А: Природничі науки. – 2009. – вип. 1. – С. 472–477. **8.** *Kop C.* An Interlingua based Approach to Derive State Charts form Natural Language Requirements In: Hamza M.H. (Hrsg.) / *C. Kop, H. C. Mayr* // Proceedings of the 7th IASTED Anaheim, Calgary, Zurich: Acta Press, – 2003. – P. 538 – 543. **9.** *Ilieva M.,* Representing Textual Requirements as Graphical Natural Language for UML-diagram Generation / *M. Ilieva, H. Boley* // Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering, San Francisco, CA. – 2008. – P. 478 – 483 **10.** *Highsmith J.* Agile Project Management. / *Highsmith J.* – Addison-Wesley – 2004. – 277 p. **11.** *Фаулер М.* Архитектура корпоративных программных приложений / *Фаулер М.*; пер. с англ. – М.:Вильямс, 2006. –404 с. **12.** *Olson L.* Advanced Data Mining Techniques/ *Olson L., Delen D.*– Berlin: Springer-Verlag. . – 2008. – 180 p. **13.** Ланде Д. В. Основы интеграции информационных потоков. / Ланде Д. В. – К. Инжиниринг, 2006. – 240 с. **14.** *Антонова А. А.* Применение латентно-семантического анализа при кластеризации слов на основании их контекстов / *А. А. Антонова, Е. А. Ильюшина, А. В. Прохоров* // Тезисы докладов научной конференции «Ломоносовские чтения». – М.: Изд-во МГУ, 2007. – С. 23–24. **15.** *Racheva Z.* Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products / *Z. Racheva, M. Daneva, L. Buglione,* // 2nd International Workshop on Software Product Management, – 2008. – P. 49–58. **16.** *Саати Т.* Принятие решений – метод анализа иерархий / *Саати Т.* – М.: Радио и связь, 1993. – 278 с. **17.** *Вентцель Е. С.* Теория вероятностей и ее инженерные приложения. / *Вентцель Е. С., Овчарова Л. А.* – М: Наука, 1988. – 480 с. **18.** Matlab – Web http://www.mathworks.com **19.** Open Office . – Web http://wiki.openoffice.org/wiki/Documentation/OOoAuthors_User_Manual/Migration_Guide/Calc_and_Excel\

Bibliography (transliterated): 1. Anderson D.J. Agile Management for Software Engineering. D.J. Anderson. Prentice Hall, 2003. 336 p. Print 2. Ambler S. Agile Architecture: Strategies for Scaling Agile Software Development. S. Ambler http://agilemodeling.com/essays/agileArchitecture.htm 3. Ambler S.

Agile/Evolutionary Data Modeling: From Domain Modeling to Physical ModelingS.Ambler. http://agiledata.org/essays/agileDataModeling.html#DisasterStrikes 4. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Evans E. Addison-Wesley, 2003. 359. Print. 5. Hong Xu. “Requirement process establishment and improvement from the viewpoint of cybernetics”. Hong Xu, Pete Sawyer, Ian Sommerville. The Journal of Systems and Software, issue 79 (2006): 1504–1513. Print. 6. Tkachuk M.V. “Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development”. M.V.Tkachuk, R.O.Gamzayev, H.C.Mayr, V.O. Bolshutkin. Problemy programmirovanija. K.: NAN Ukraine. 2012. № 2–3 (spec. vypusk). 252–260. Print. 7. Seljakov E.B. “Metodika vybora trebovanij pri proektirovanii ASUTP na osnove logiko-formal'noj modeli sistemy trebovanij.” E.B. Seljakov. Visnik Donec'kogo nacional'nogo universitetu, Ser. A: Nature Scince. 2009. vol. 1. .472–477. Print. 8. Kop C. “An Interlingua based Approach to Derive State Charts form Natural Language Requirements” In: Hamza M.H. (Hrsg.). C. Kop, H.C. Mayr. Proceedings of the 7th IASTED Anaheim, Calgary, Zurich: Acta Press, 2003. 538–543. Print. 9. Ilieva M., “Representing Textual Requirements as Graphical Natural Language for UML-diagram Generation.” M. Ilieva, H. Boley. Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering, San Francisco, CA. 2008. 478–483. Print. 10. Highsmith J. Agile Project Management. Highsmith J. Addison-Wesley, 2004. 277. Print. 11. Fauler M. Arhitektura korporativnyh programmnyh prilozhenij. Fauler M.; per. s angl. M.:Vil'jams, 2006. 404. Print. 12. Olson L. “Advanced Data Mining Techniques.” Olson L., Delen D. Berlin: Springer-Verlag. 2008. 180. Print. 13. Lande D.V. 1. Anderson D.J. Agile Management for Software Engineering D.J. Anderson. Prentice Hall, 2006. 240. Print. 14. Antonova A.A. “Primenenie latentno-semanticheskogo analiza pri klasterizacii slov na osnovanii ih kontekstov.” A.A. Antonova, E.A. Il'jushina, A.V. Prohorov. Tezisy dokladov nauchnoj konferencii “Lomonosovskie chtenija”. M.: Izd-vo MGU, 2007. 23–24. Print. 15. Racheva Z. “Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products” Z.Racheva, M.Daneva, L.Buglione, 2nd International Workshop on Software Product Management, 2008. 49–58. Print. 16. Saati T. Prinjatie reshenij – metod analiza ierarhij. Saati T. M.: Radio and connecion, 1993. 278 p. Print 17. Ventcel' E.S. Teorija verojatnostej i ee inzhenernye prilozhenija. Ventcel' E.S., Ovcharova L.A. M: Scince, 1988. 480. Print. 18. Matlab. MathWorks. Matlab, 2015 Web 13 June 2015 <http://www.mathworks.com> 19. Open Office. Differences in Use between Calc and Excel, OpenOffice, 2013 Web Web 13 June 2015 <http://wiki.openoffice.org/wiki/Documentation/OOoAuthors_User_Manual/Migration_Guide/Calc_and_Excel\>

Received 12.10.2015