Розглянуто задачу вдосконалення базового методу сценарного аналізу функціональних вимог до інформаційної системи (ІС). Проаналізовано результати досліджень методів сценарного аналізу функціональних вимог до ІС. Головним їх недоліком визнано необхідність виконання цих методів аналітиками виключно вручну. Для усунення цього недоліку запропоновано вдосконалити базовий метод сценарного аналізу за рахунок використання моделей і методів, основаних на формальному представленні знань.

Для формального опису представлення сценарію виконання функціональної вимоги на рівні знань запропоновано використовувати модель структурних паттернів проектування функціональних вимог. Показано, що формальний опис знань, які вилучаються з діаграм Use Case, є частковим випадком даної моделі. Запропоновано модель підкласу структурних паттернів проектування сценаріїв виконання функціональних вимог.

Розроблено вдосконалений метод сценарного аналізу функціональних вимог до ІС. Суть вдосконалення полягає у виділенні з публікацій сценаріїв виконання функціональних вимог знань та наступному аналізі знання-орієнтованих описів цих сценаріїв з метою виявлення дублюючих один інший сценаріїв виконання різних функціональних вимог до ІС. Для виявлення та усунення випадків подібного дублювання запропоновано використовувати вдосконалений метод синтезу варіантів описів архітектури створюваної ІС.

Наведено приклад апробації вдосконаленого методу сценарного аналізу функціональних вимог під час аналізу функціональних вимог до проекту функціонального модуля безпеки праці. Результати апробації підтверджують достовірність запропонованого методу.

Запропонований вдосконалений метод сценарного аналізу функціональних вимог до ІС дозволяє отримати описи архітектури створюваної ІС на основі значно меншого обсягу інформації про функціональні вимоги до цієї ІС

Ключові слова: функціональні вимоги, метод сценарного аналізу, діаграма Use Case, знання-орієнтована модель, опис архітектури

UDC 044.03; 658.11.05.06
DOI: 10.15587/1729-4061.2019.170351

IMPROVEMENT OF THE METHOD OF SCENARIO ANALYSIS OF FUNCTIONAL REQUIREMENTS TO AN INFORMATION SYSTEM

Mohammed Q. Mohammed
PhD, Senior Lecturer*
E-mail: mqmhf82@gmail.com
Saif Q. Muhamed
PhD, Senior Lecturer*
E-mail: saifkassimm@gmail.com
M. Ievlanov
Doctor of Technical Science,
Associate Professor**
E-mail: maksym.ievlanov@nure.ua
Z. Gazetdinova**
E-mail: zarinagazetdinova@gmail.com
*Department of
Information Technology and Business
University of Information Technology
and Communications
Al-Nidhal str., Baghdad, Iraq, 00964
**Department of Information Control Systems
Kharkiv National University of Radio Electronics
Nauky ave., 14, Kharkiv, Ukraine, 61166

1. Introduction

Modern trends in the development of IT sector pay special attention to reducing unproductive expenses of IT projects. One way to reduce these costs is to maximally accurately identify the content of an IT project during its initiation and planning [1]. Solving this task is typically considered as part of the work to build and analyze the requirements of rightsholders to the created system [2].

A modern paradigm for describing and modeling system requirements is based on the publication of the requirements in the form of scripts to perform certain activities. An example of the implementation of a given paradigm using visual models of the Unified Modeling Language (UML) is a model of the requirement provided by Microsoft Corporation [3].

However, the application of a given paradigm is complicated when solving the tasks on a system requirements analysis. Using a scenario approach to the description and publication of requirements is seriously complicated while defining the boundaries for a created or upgraded system. In addition, the use of UML, and, in particular, the Use Case diagrams for the description of scenarios for meeting the requirements by rightsholders requires the development of special models, methods, and information technologies to formalize and automate the work on a requirements analysis. Running an analysis of scenarios for meeting the requirements by rightsholders manually inevitably causes a high risk of error going undetected. These errors can be caused by an incorrect definition of the scenarios for implementing individual requirements of rightsholders, as well as synthesis of the descriptions of the system architecture without aligning the descriptions of scenarios for implementing individual requirements to the functions of a given system.

The base method for scenario analysis of the rightsholders' requirements to a system was considered in [4] as a sequence of the following stages.

Stage 1. Selection of a base scenario for a functional requirement, which describes activities of the system that occur most frequently regardless of the type of a request by an actor and the conditions for implementing these queries.

Stage 2. Description of relationships among requirements based on the Use Case diagrams.

Stage 3. Identification of mutually exclusive requirements and decomposition of individual scenarios.

Stage 4. The formation of the resulting Use Case diagram that describes the stated functional requirements to a system and the transformation of a given diagram into a numbered list of the functional requirements to the system.

This method's implementation is based on the assumption about a possibility to represent a rightsholder's requirement in the form of a single basic scenario and a set of many additional scenarios that extend the possibilities of the base scenario for different situations in a subject area [4].

The base method for scenario analysis in IT projects aimed at creating information systems (IS) is, at present, implemented by analysts mostly manually. In this case, Stages 1 and 2 are performed by analysts based on general recommendations on the collection and publication of the rightsholders' requirements. Stages 3 and 4 are performed by analysts with the use of methods for a visual analysis of the Use Case diagrams, which describe the stated functional requirements to the system. Such an organization of work stems from the fact that existing CASE-tools for visual modeling, as well as information technology to control requirements, have no special functions to analyze the Use Case diagrams. In addition, limited size of display screens leads to inevitable exclusion of a large group of elements from the resulting Use Case diagram from the analysts' consideration, particularly for cases that describe the architecture of medium and large IS.

Therefore, it is a relevant task to automate the method for analyzing the scenarios of fulfilling the rightsholders' requirements, which would make it possible to reduce the time required to perform this analysis and to eliminate errors caused by the analysts' lack of attention.

2. Literature review and problem statement

An analysis reported in [5] shows that there is a growing interest in scientific and applied research into the aspects of application of models and methods for requirements engineering. In this case, the field of requirements engineering that addresses models and methods of requirements analysis, as well as features of their practical application, is one of the most important areas of such studies.

Modern research in the field of analysis of functional requirements, and in particular scenario analysis, can be divided into three main areas. One of these areas implies considering a scenario analysis as a special case of the general methods for analysis of requirements based on formal models and methods. In this area, one can note, as an example, reported in [6], the representation of the process of establishing requirements to data as a feedback control system with a continuous optimization of the models of user behavior. In [7], the behavior of users of the created system is proposed to simulate using an apparatus of category theory, based on which, by using graphics methods, a specialized declarative language was constructed. Paper [8] proposes an algebraic approach to the analysis of probabilistic models of software performance.

However, the construction of tools based on the results obtained in this field is difficult because of the high complexity of formal models and methods for analysis of requirements to IS. Therefore, one should consider the two other areas of research into the methods of requirements analysis more promising.

The first of these areas implies the creation and modification of requirements analysis methods by using maximally easy-to-implement tools. Thus, paper [9] proposed to reduce the number of errors in requirements by establishing a special method of requirements analysis that helps bridge the gap in communication between a customer and a developer. Article [10] suggests a method for identifying and analyzing requirements to software development, based on the joint participation of representatives of all stakeholders in the IT project.

However, the results obtained in the framework of a given direction do not make it possible to solve the task on the automated execution of a scenario analysis. In the best case, the tools that are created based on such results enable the automated execution of Stages 1 and 2 from the base method of scenario analysis. In this case, the most complex Stages 3 and 4 from a given method, which are the main source of errors, are excluded from researchers' consideration.

The second area of research implies the development and improvement of models and methods of requirements analysis based on the identification and the formal description of knowledge from unstructured and weakly-structured texts. However, studying the application of such models and methods in requirements engineering makes it possible to draw the following conclusions [11, 12]:

a) there is empirical evidence of the benefits of using ontologies, knowledge-oriented models and methods in requirements engineering specifically to reduce ambiguities, inconsistencies. and incompleteness in requirements;

b) the process of requirements engineering in most studies is considered only partially;

c) at present, there is no any uniform style for modeling the processes of requirements engineering based on ontologies, knowledge-oriented models and methods;

d) most research in this field relate only to functional requirements;

e) there are no ontologies, knowledge-oriented models and methods for requirements engineering, which would be commonly used in the community of specialists in this field.

Currently, most studies in this direction focus on investigating particular improvements for base models and methods [5]. Thus, paper [13] discusses the issues on comparing and merging the elements of a system whose description are published in the form of Use Case diagrams, Activity diagrams, and data flow diagrams. Article [14] addresses the issue of converting the publications on requirements by rightsholders in an executable system model using behavior models by applying the Activity and State diagrams of UML. Solving the tasks on analyzing the requirements to IS, the description of which employed the UML class diagrams, was considered by one of the authors of the current article in [15]. However, the issues on the improvement of models and methods for the analysis of scenarios for meeting requirements in the form of the Use Case diagrams without reference to the particular characteristics of subject areas, have remained almost unexplored. Therefore, one can draw a conclusion about the need to undertake specialized studies in the field of development of models and methods for a scenario analysis of system requirements, based on the knowledge-oriented models of requirements publications in the form of the Use Case diagrams.

3. The aim and objectives of the study

The aim of this work is to improve the base method for analyzing functional requirements to a created or modified IS based on the knowledge-oriented models of scenarios for meeting the rightsholders' requirements. Results of such an improvement should reduce the amount of time required to run a scenario analysis of functional requirements to IS by automating the implementation of a given method within the framework of tools to manage requirements.

To accomplish the aim, the following tasks have been set:

– to adapt the models for representing functional requirements to IS at the level of knowledge to patterns in scenario description of functional requirements;

– to improve a method of scenario analysis of functional requirements to IS based on the customized models.

4. Results of adaptation of models representing the functional requirements to an information system

The models for representing the functional requirements to IS at the knowledge level were considered in [16] as a subclass of structural patterns in the design of functional requirements. In a general case, this subclass takes the following form:

$$K_{IS}^{Pt} = \{Pt_{fr\_str}, Pt_{if}, Pt_{fr\_rel}, Pt_{net\_fr}\} =$$
$$= \{< At_n, At_{el\_fr}, At_{el\_fr\_t}, < at_n, at_{el\_fr}, at_{el\_fr\_t} >>,$$
$$< At_g, At_{el\_if}, At_{el\_if\_t}, < at_g, at_{el\_if}, at_{el\_if\_t} >>,$$
$$< At_{fr\_rel\_n}, At_{el\_fr\_rel}, At_{el\_fr\_rel\_t},$$
$$< at_{fr\_rel\_n}, at_{el\_fr\_rel}, at_{el\_fr\_rel\_t} >>,$$
$$< at_n^1, at_n^2, at_{if}, at_{fr\_rel\_n} >\}. \quad (1)$$

Here Pt_{fr_str} is the model of a structural pattern for a frame design; At_n is the tuple of attributes that describe the name of a frame; At_{el_fr} is the tuple of attributes that describe an element of a frame (slot, interface, method); $At_{el\_fr\_t}$ is the tuple of attributes describing the type of a frame element; at_n is the attribute that identifies the name of the frame; at_{el_fr} is the attribute that identifies the frame's element; $at_{el\_fr\_t}$ is the attribute that identifies the type of an element in a frame; Pt_{if} is the model of a structural pattern in the design of a frame's interface; At_g is the tuple of attributes describing the globally unique identifier of the interface of the frame; At_{el_if} is the tuple of attributes that describe an element in the interface of a frame (slot, method); $At_{el\_if\_t}$ is the tuple of attributes that describes the type of an element in the frame's interface; at_g is the attribute that identifies the globally unique identifier of a frame's interface; at_{el_if} is the attribute that identifies an element in the interface of a frame; $at_{el\_if\_t}$ is the attribute that identifies the type of an element in a frame's interface; Pt_{fr_rel} is the model of a structural pattern in the design of relationships between nodes in a network of frames; $At_{fr\_rel\_n}$ is the tuple of attributes describing the name of the relationship; $At_{el\_fr\_rel}$ is the tuple of attributes describing the description element of relation; $At_{el\_fr\_rel\_t}$ is the tuple of attributes describing the type of an element in the description of relation; $at_{fr\_rel\_n}$ is the attribute that identifies the name of a relationship; $at_{el\_fr\_rel\_t}$ is the attribute that identifies the type of an element in the description of relation; $at_{el\_fr\_rel\_t}$ is the attribute that identifies the type of an element in the description of relation; *Ptnet_fr* is the model of a structural pattern in the design of a network of frames; at_n^1 is the attribute that identifies the name of the first frame that can participate in forming a relationship (maybe not defined); at_n^2 is the attribute that identifies the name of the second frame, which can participate in forming a relationship (maybe not defined) [16].

It should be noted that some authors propose a different set of elements to create the use case diagrams. However, in most cases, a basic set of such elements includes [4]:

– elements that relate to the class "Actor", which reflect the roles of staff in relation to the system (or a business process (BP));

– elements that relate to the class "Use Case" that reflect BP in general, individual operations within BP or individual functions of the developed IS;

– elements that relate to the class of "Interface" that reflect the existing relations between the elements of type Actor and elements of type Use Case;

– elements that relate to the class "Extends" that reflect relationships between individual elements of the type Actor or Use Case in cases when one of the elements is similar to another, but carries a somewhat larger load;

– elements that relate to the class "Uses" that reflect relations between individual elements of the type Use Case in cases when one element is repeated more than once and copying its description is undesirable for certain reasons.

It should also be noted that the Use Case diagrams are purely of declarative character. Each element in a diagram, regardless of its belonging to one of the above classes, declares its existence, but does not make it possible to concretize its implementation in the diagram. Thus, for example, the existence in a diagram of an element from the class Extends, relating two elements from the class Use Case, means that the child element of Use Case is similar to the parent, but it has some features missing in the parent element. It does not follow, however, that a given element Extends will be absolutely identical to a relationship of the type "Generalization" in the UML class diagram.

This feature makes it possible to represent a Use Case diagram elements as special cases of frames, whose description is defined by the pattern Pt_{fr_st}. In this case, in order to describe the relationships between the elements of the diagram, it is inappropriate to use the descriptions that are generated based on the pattern Pt_{fr_rel}. Then any Use Case diagram can be represented as a separate network of frames, the relations between which are exclusively of a service nature and do not have their own semantics. Such a network can be represented by the following expression

$$Pt_{net\_fr}^{UseCase} =< Pt_{fr\_str}, < at_n^1, at_n^2, at_{fr\_rel\_id} >>, \quad (2)$$

where $at_{fr\_rel\_id}$ is the attribute that identifies the relationship between two frames of the Use Case diagram.

The proposed formal description of the Use Case diagram at the level of knowledge makes it possible to consider a subclass of structural patterns for the design of scenarios to implement functional requirements as a special case of the subclass of structural patterns in the design of functional requirements (1), $K_{UseCase}^{Pt} \subset K_{IS}^{Pt}$. Then, in a general case, the model of a subclass of structural patterns for the design of scenarios for implementing functional requirements takes the following form:

$$K^{Pt}_{UseCase} = \{Pt_{fr\_str}, Pt^{UseCase}_{net\_fr}\} = \\ = \{< At_n, At_{el\_fr}, At_{el\_fr\_t}, < at_n, at_{el\_fr}, at_{el\_fr\_t} >>, \\ < at^1_n, at^2_n, at_{fr\_rel\_id} >\}. \quad (3)$$

The use of model (3) makes it possible to implement a scenario analysis of a rightsholder's requirements as a special case of analysis of the stated functional requirements. In this case, the implementation of a scenario analysis does not require a fundamental change in the elements of an appropriate information technology, described in [17].

5. Results of improving the method for a scenario analysis of functional requirements to an information system

Among the above discussed stages in a base method of scenario analysis, the most time-consuming is Stage 4. A given stage implies the formation of the resulting Use Case diagram that describes the stated functional requirements to a system. In this case, it is necessary to eliminate the following cases of elements overlap in the resulting Use Case diagram:

– overlapping individual Use Case diagrams that describe different requirements by rightsholders;

– overlapping individual frames of the resulting Use Case diagram describing various elements in a given diagram.

In the description of a base scenario analysis, Stage 4 is recommended to run entirely by hand. In this case, the search and elimination of overlapping fragments in as the resulting Use Case diagram are recommended to carry out based on the results from a visual analysis of this diagram.

The use of the proposed model of the subclass of structural patterns for designing the scenarios for implementing functional requirements (3) can improve the method for a scenario analysis by automating the implementation of its individual stages. The improved method of scenario analysis is proposed to be represented as a sequence of the following stages.

Stage 1. Selection of a base scenario for a rightsholder's requirement, which describes activities of IS, repeated most often, regardless of a query type from the actor and conditions for fulfilling these queries.

Stage 2. Construction of a set $\{P^i_{UseCase}\}, i = 1,...,n$ of publications of scenarios for meeting the requirements by rightsholders based on the Use Case diagrams, where n is the number of individual requirements by rightsholders.

Stage 3. Construction of a set $\{K^i_{UseCase}\}, i = 1,...,n$ of representations of scenarios for meeting the requirements by rightsholders at the knowledge level based on model (3).

Stage 4. Identification of contradictory requirements by fulfilling the method, described in [15], for analysis of individual frames of representations of requirements for consistency.

Stage 5. Automatic generation of variations for the resulting Use Case diagram that describes the architecture of IS as a set of the stated functional requirements to IS.

Stage 6. Selection of the rational description of IS architecture based on the results from comparative analysis of the resulting Use Case diagrams, formed at Stage 5.

Stage 7. Convert the selected rational description of the IS architecture into a numbered list of functional requirements to the system.

The methods for generating representations of scenarios for meeting the rightsholders' requirements at the level of knowledge in a general case are similar to the methods described in [18].

The greatest attention should be given to methods for constructing the variants to the resulting Use Case diagram. The application of these methods should lead to the construction of descriptions of such an Use Case diagram that would eliminate the identified cases of duplications and describe the maximum number of scenarios for meeting the rightsholders' requirements. Thus, Stage 5 in the improved method for scenario analysis is to considered as a sequence of the following steps.

Step 5. 1. Implementation of the improved method for synthesis of variants for descriptions of the architecture of a created IS for the set of representations of scenarios for meeting the rightsholders' requirements at the level of knowledge.

Step 5. 2. Implementation of the modified method for synthesis of variants of descriptions of the architecture of a created IS for the variants of resulting representation of scenarios for meeting the functional requirements to IS at the level of knowledge, formed as a result of implementing Step 5. 1.

To perform these steps, it is proposed to use the improved method for synthesis of the descriptions of architecture of a created IS, proposed in [19]. In this case, the use of a given method at Step 5. 1 will be slightly different from the use of this same method at Step 5. 2. Let us consider these differences in more detail.

Application of the improved method for the synthesis of variants of descriptions of architecture of a created IS at Step 5. 1 is proposed to represent as a sequence of the following stages.

Stage 1. Generate the initial variant for the description of architecture of a created IS $Arch_{base}$.

Step 1. 1. Determine the number of representations of scenarios for meeting the rightsholders' requirements at the level of knowledge n.

Step 1. 2. Generate a set of descriptions of IT-services $\{IT_{acm_i}\}$ by performing operation

$$IT_{acm_i} = K^i_{UseCase}, \quad i = 1,...,n.$$

Step 1. 3. For the set $\{IT_{acm_i}\}$, formed at Step 1.2, construct a matrix of architecture description $Arch_{base}$ of the following form:

$$Arch_{base} = \\ = \begin{bmatrix} t_{11}(IT_{acm_1}) = 1 & ... & t_{1j}(IT_{acm_1}) = 1 & ... & t_{1n}(IT_{acm_1}) = 1 \\ ... & ... & ... & ... & ... \\ 0 & ... & t_{ij}(IT_{acm_i}) = 1 & ... & t_{in}(IT_{acm_i}) = 1 \\ ... & ... & ... & ... & ... \\ 0 & ... & 0 & ... & t_{nn}(IT_{acm_n}) = 1 \end{bmatrix}. \quad (4)$$

Stage 2. Set the value for coefficient of repulsion r and calculate a win function

$$\Pr ofit(IT_{acm}, r) = \sum_{j=1}^{n} \frac{S(IT_{acm_j})}{W(IT_{acm_j})^r} \times \left|IT_{acm_j}\right| \Big/ \sum_{j=1}^{n} \left|IT_{acm_j}\right|.$$

Here $S(IT_{acm_j})$ is the number of elements in a scenario for meeting the *j-th* requirement by User to an IT-service;

$W(IT_{acm_j})$ is the number of unique elements in a given scenario; $|IT_{acm_j}|$ is the number of IT services in the examined variant of an IS architecture description; r is the factor of repulsion, defining the degree of permissible duplication of individual requirements by an IT service user in a created IS (for IS with a monolithic architecture, r=1, for IS with a modular architecture, r=2, for IS with a service-oriented architecture, $r \geq 3$).

Stage 3. Conduct synthesis of optimal and/or acceptable variants of architecture description for a created IS.

Step 3. 1. Accept $Profit_{max}=Profit(IT_{acm},r)$, i=1, $j=i$+1.

Step 3. 2. If $t_{ij}(IT_{acm_i})=1$, exclude $K^i_{UseCase}$ from IT_{acm_i} and include $K^i_{UseCase}$ in IT_{acm_j}. Otherwise, proceed to Step 3.7.

Step 3. 3. Calculate for the resulting variant of architecture description the value for

$$\Pr ofit(IT_{acm},r)=\sum_{j=1}^{n}\frac{S\left(IT_{acm_j}\right)}{W(IT_{acm_j})^r}\times\left|IT_{acm_j}\right|\Bigg/\sum_{j=1}^{n}\left|IT_{acm_j}\right|.$$

Step 3. 4. If $Profit(IT_{acm},r)>Profit_{max}$, accept

$$t_{jj}(IT_{acm_j})=t_{jj}(IT_{acm_j})+t_{ii}(IT_{acm_i}),$$

$$t_{ij}(IT_{acm_i})=0,\ \ t_{ji}(IT_{acm_i})=0$$

for j=1,...,n,

$$K^j_{UseCase}=K^j_{UseCase}\cup K^i_{UseCase}$$

and proceed to Step 3. 1.

Step 3. 5. If

$$\Pr ofit(IT_{acm},r)\in[\Pr ofit_{\max}-\varepsilon;\Pr ofit_{\max}],$$

accept $t_{ij}(IT_{acm_i})=1$.

Step 3. 6. If

$$\Pr ofit(IT_{acm},r)<[\Pr ofit_{\max}-\varepsilon;\Pr ofit_{\max}],$$

accept $t_{ij}(IT_{acm_i})=0$.

Step 3. 7. Accept $j=j$+1. If $j \leq n$, proceed to Step 3. 2.

Step 3. 8. Accept $i=i$+1, $j=j$+1. If $i<n$, proceed to Step 3. 2. Otherwise, finalize implementation of the method's stage.

Stage 4. Exclude from consideration all variants of architecture description for a created IS $Arch_{base}$, registered at Stage 3, for which the following condition does not hold

$$\Pr ofit(IT_{acm},r)\in[\Pr ofit_{\max}-\varepsilon;\Pr ofit_{\max}].$$

Finalize implementation of the method.

Step 4. 1. Generate a variant of architecture description for a created IS, including those IT_{acm_i}, for which

$$t_{ii}(IT_{acm_i})\geq 1.$$

Step 4. 2. For each $t_{ij}(IT_{acm_i})=1$ in matrix $Arch_{base}$, generate a variant of architecture description for a created IS, accepting during generation

$$K^i_{UseCase}=K^i_{UseCase}\cup K^j_{UseCase},\ \ IT_{acm_i}=\varnothing \text{ and } t_{jj}(IT_{acm_j})=0.$$

Finalize implementation of the method.

As shown in [19], the application of the improved method makes it possible to reduce the number of iterations in the search for overlapping representations of scenarios for meeting the rightsholders' requirements.

The main difference in the application of the improved method of synthesis of variants for architecture descriptions of a created IS at Step 5. 2 is the construction of a set $\{K^i_{UseCase}\}, i=1,...,n$ from the representations of individual frames of each architecture description for a created IS, generated as a result of implementing Step 5. 1. For all other aspects, application of a given method at Step 5. 2 is no different from the use of a given method at Step 5. 1.

6. Verification of elements from the improved method of scenario analysis of functional requirements to an information system

As noted above, the implementation of functions that enable the automated execution of a scenario analysis would not require radical changes to the elements of the appropriate information technology aimed at generating and analyzing the requirements described in [17]. Because model (3) model is a special case of model (1), the fragment of data scheme representing the description of data scheme in the technology for an automated scenario analysis takes the form shown in Fig. 1. This scheme omits some minor elements that do not seriously change the essence of the proposed solution.

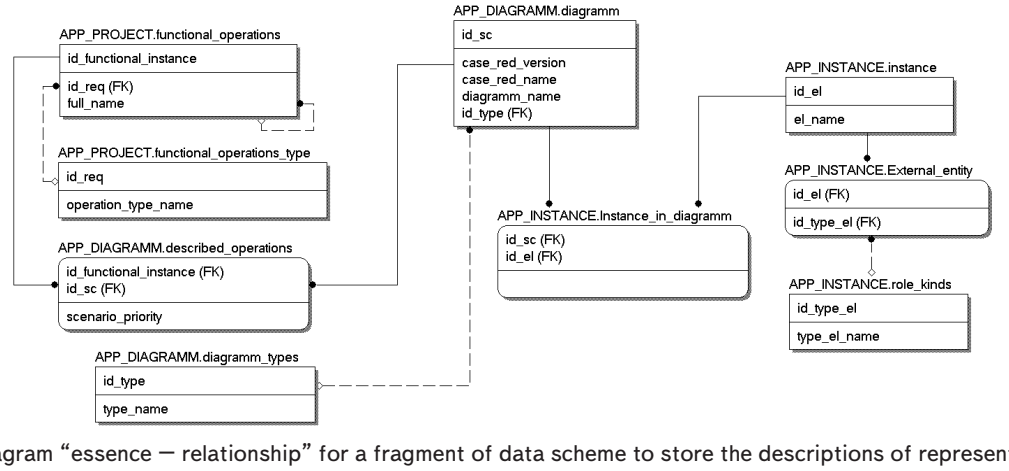

Fig. 1. Diagram "essence – relationship" for a fragment of data scheme to store the descriptions of representations of requirements to an information system

Consider the data scheme shown in Fig. 1 in more detail. The essence APP_DIAGRAMM.diagramm describes the names of visual models of IT services and CASE-tools employed for their generation. The essence APP_DIAGRAMM.diagrammTypes describes the types of visual models of IT services. The essence APP_PROJECT.functional_operations describes individual functional operations for entering, storing, processing, and generating the output data for an IT-service and the IT-service that implements this service. The essence APP_PROJECT.functional_operations_types describes the types of individual functional operations. The essence APP_DIAGRAMM.described_operations describes evidences of descriptions of individual functional operations in specific visual models of IT services. The unification of frame descriptions of elements in the visual models of IT Services is executed by the essence APP_INSTANCE.instance. This essence describes the elements of both static and behavioral visual models of IT services. The essence APP_INSTANCE.Instance_in_diagramm describes the evidence of the existence of an element in specific visual models. The essence APP_INSTANCE.role_kinds describes the types of elements in visual models. The essence APP_INSTANCE.External_essence describes the types of specific elements of the visual models of IT services. For the essences considered, Fig. 1 shows the required (denoted by solid line) and optional (shown dashed) relationships of the type "one-to-many".

To verify the improved method of scenario analysis, it was decided to use an example, described in [15, 19], of designing a functional module of safety at work (FM SW). The User of IT services put forward the following requirements to a given module:

a) "to implement the function to register information on the enterprise and the processes (operations) executed at a given enterprise, which, during their execution, might negatively affect the employees at the enterprise through a set of harmful industrial factors (HIF)" (first functional requirement);

b) "to implement the function to register personnel data (data on employees at the enterprise), minimally required for making management decisions to ensure safety at work at the enterprise" (second functional requirement);

c) "to implement the function of compiling and keeping a HIF handbook that act or can act during the execution of individual processes or operations at the enterprise" (third functional requirement);

d) "to implement the function of integrating the results of observations of the effect of each HIF during the execution of processes or individual operations at the enterprise" (fourth functional requirement);

e) "to implement a function to forecast the impact exerted by a set of HIF on the body of an employee performing a separate process or operation at the enterprise" (fifth functional requirement).

Using this example makes it possible to compare the results of application of the improved method of scenario analysis to those solutions, obtained earlier, based on more detailed descriptions of functional requirements to FM SW.

When executing Stage 1 and Stage 2 of the improved method of scenario analysis, we obtained publications of the above functional requirements in the form of the Use Case diagrams, shown in Fig. 2–6.

In Fig. 2–6, the following designations are used: DCE – department of chief engineer; SWD – safety at work department; DHR – department of human resources; HIF – harmful industrial factor.

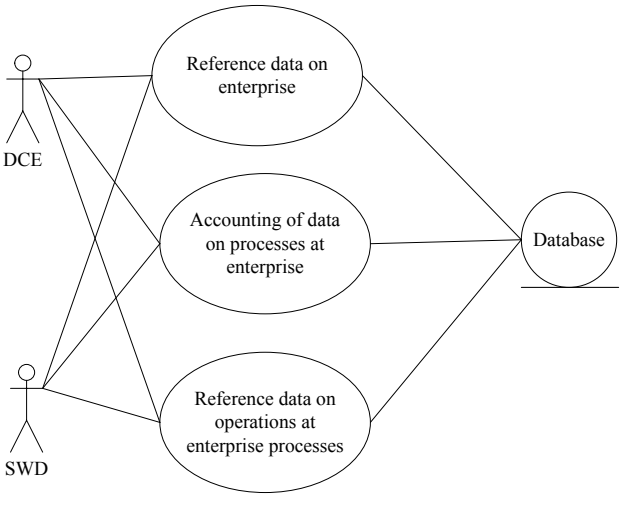

Fig. 2. Scenario for fulfilling the first functional requirement

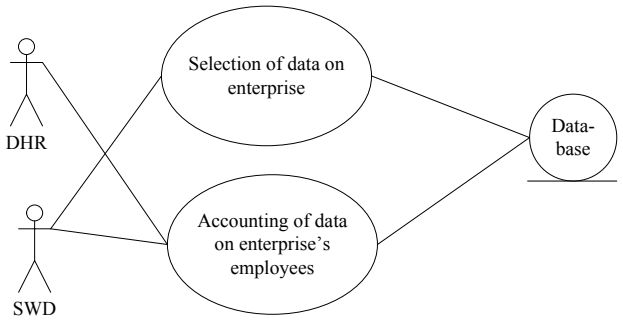

Fig. 3. Scenario for fulfilling the second functional requirement

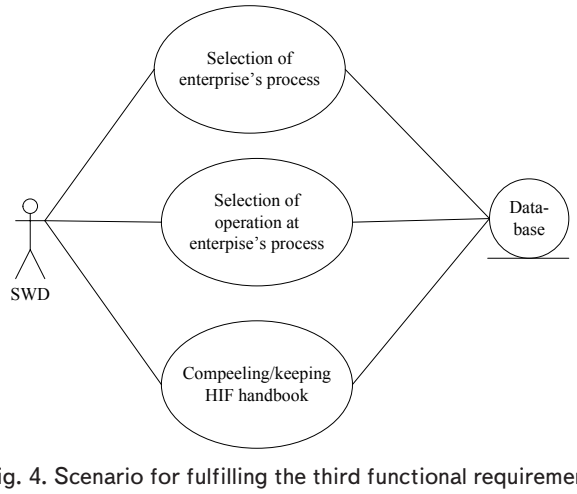

Fig. 4. Scenario for fulfilling the third functional requirement

When executing Stage 3, the knowledge is derived from the Use Case diagrams shown in Fig. 2–6, and one constructs a set $\{K^i_{UseCase}\}$, $i=1,...,5$ of representations of scenarios for meeting the rightsholders' requirements at the level of knowledge. This construction is performed by entering data to the tables of a database whose scheme is shown in Fig. 1.

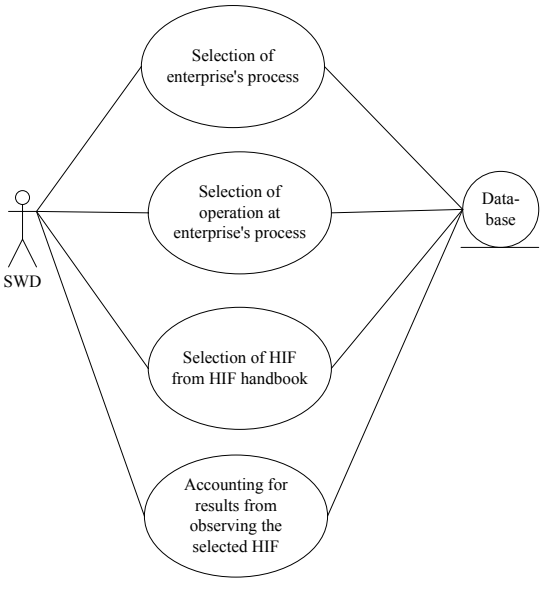

Fig. 5. Scenario for fulfilling the fourth functional requirement

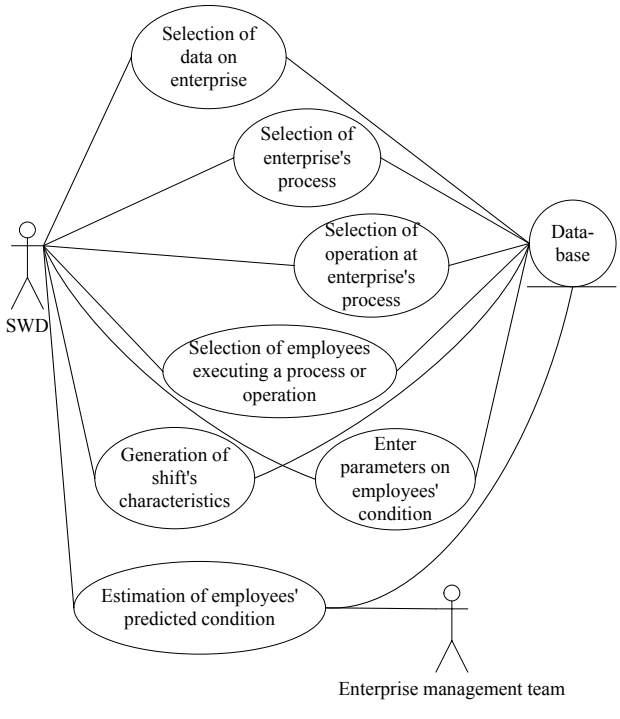

Fig. 6. Scenario for fulfilling the fifth functional requirement

When executing Stage 4, one identifies conflicting requirements. An example of a given type of analysis for individual frames is discussed in detail in [15].

While executing Stage 5, the first to run is Step 5.1. Consider the features of performing this step in detail.

When executing Stage 1 in the improved method of synthesis of variants of architecture descriptions for a created IS, one performs the following activities. As a result of performing Step 1.1, one determines from the database table APP_DIAGRAMM.diagramm the number of representations of scenarios for meeting the rightsholders' requirements at the knowledge level n=5.

As a result of performing Step 1.2, based on the completed database, one generates a set of descriptions of IT-services $\{IT_{acm_i}\}$ by performing operation

$$IT_{acm_i} = K^i_{UseCase}, \quad i = 1,...,5.$$

As a result of performing Step 1. 3, one constructs, for the set $\{IT_{acm_i}\}$, generated at Step 1.2, in accordance with expression (4), a matrix of architecture description $Arch_{base}$ of FM SW in the following form:

$$Arch_{base} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (5)$$

Matrix (5) is the initial variant of architecture description for a created IS $Arch_{base}$. A given variant implies the implementation of each considered requirement by the IT services User using a separate function of FM SW. In this case, the matrix (5) elements, residing above the main diagonal, that are equal to 1, show a possibility, when executing Stage 3, to merge individual requirements by the customer of IT services to identify the overlapping requirements to FM SW. The matrix (5) elements that reside under the main diagonal elements, which are equal to 0, will be used when executing Stage 3 to exclude the repeated execution of steps at Stage 3 to identify the overlapping requirements by the User of IT services.

The result of implementation of Stage 2 is the established value for repulsion coefficient r=2 and the calculated value for a win function

$$\mathrm{Pr}\,ofit\left(IT_{acm},2\right) = \sum_{j=1}^{5} \frac{S\left(IT_{acm_j}\right)}{W(IT_{acm_j})^2} \times \left|IT_{acm_j}\right| \Big/ \sum_{j=1}^{5} \left|IT_{acm_j}\right|.$$

The calculation results are given in Table 1.

Table 1

Result of implementing Stage 2 of the improved method for synthesizing variants of architecture descriptions for a created information system

Scenario number	Scenario representation at the knowledge level	Scenario value $S(IT_{acm_j})$	Scenario value $W(IT_{acm_j})$
1	$\{K^1_{UseCase}\}$	15	15
2	$\{K^2_{UseCase}\}$	10	10
3	$\{K^3_{UseCase}\}$	11	11
4	$\{K^4_{UseCase}\}$	14	14
5	$\{K^5_{UseCase}\}$	23	23
Result of calculation of function $\mathrm{Pr}\,ofit\left(IT_{acm},r\right)$			
$\left\|IT_{acm_j}\right\| = 5$	$r = 2$	$\mathrm{Pr}\,ofit\left(IT_{acm},r\right) = 0{,}0744$	

Table 1 shows for each scenario of meeting the requirement by an IT-services User the number of elements within a given scenario $S(IT_{acm_j})$, as well as the number of unique elements within a given scenario $W(IT_{acm_j})$. In addition, Table 1 gives the number of IT services for the examined variant of architecture description for FM SW $\left|IT_{acm_j}\right|$ and the value for repulsion coefficient *r*. These data were used to calculate the value for function $\Pr ofit(IT_{acm},r)$, the result of which is also given in Table 1. This result characterizes the level of overlapping requirements by an IT-services User in the variant of architecture description of FM SW, proposed as a result of the implementation of Stage 2. When executing Stage 3, the result of the calculation of value for function $\Pr ofit(IT_{acm},r)$, shown in the Table 1, is used to quantify the overlapping requirements by an IT-services User.

When executing Stage 3 of the improved method for synthesizing the variants of architecture descriptions for a created IS, we conducted an iterative search for overlapping scenarios. Let us consider the features in performing these iterations using an example of the first iteration of Stage 3 implementation.

When executing Step 3.1 at the first iteration, the value for function $\Pr ofit(IT_{acm},r)=0{,}0744$, computed when executing Stage 2, is accepted as the maximum. In addition, we set the values for variables i=1, j=2.

When implementing Step 3. 2 at the first iteration, as a result of checking the value for an element in the matrix description of the base architecture for i=1, j=2, it was established that $t_{12}(IT_{acm1})=1$.

When executing Step 3.3 at the first iteration one calculates the value for function $\Pr ofit(IT_{acm},2)$ in order to verify the assumption on the duplication of the first and second scenarios:

$$\Pr ofit(IT_{acm},r)=\frac{0+\left(\frac{25}{23^2}\right)\times 2+\frac{11}{11^2}+\frac{14}{14^2}+\frac{23}{23^2}}{5}=$$
$$=\frac{0.094+0.091+0.071+0.043}{5}=0.0598. \quad (6)$$

When implementing Step 3. 4 at the first iteration the condition that is checked, 0.0598>0.0744, does not hold.

When implementing Step 3. 5 at the first stage one forms the upper and lower values for range $\Pr ofit_{\max}=0.0744$ and

$$\varepsilon=0{,}1\times\Pr ofit_{\max}-0{,}1\times 0{,}0744=0{,}00744, \quad (7)$$

$$\Pr ofit_{\max}-\varepsilon=0.0744-0.00744=0.06696, \quad (8)$$

respectively. The checked condition for the inclusion of expression (6) into the predefined range does not hold.

When implementing Step 3.6 at the first iteration the checked condition 0.0598<0.06696 holds. This means that the proposed variant of architecture description for FM SW is worse than the initial one. To exclude it from further consideration we adopt $t_{12}(IT_{acm_1})=0$, and matrix (5) is reduced to the following form:

$$Arch_{base}=\begin{bmatrix}1&0&1&1&1\\0&1&1&1&1\\0&0&1&1&1\\0&0&0&1&1\\0&0&0&0&1\end{bmatrix}. \quad (9)$$

When implementing Step 3.7 at the first iteration one accepts the value for variable j=2+1=3. Since 3<5, we finalize implementation of the iteration and return to Step 3.2.

All subsequent iterations of Stage 3 in the improved method for synthesizing the variants of architecture descriptions for a created IS are performed similarly.

When implementing the eighth iteration of Stage 3 we found the new maximum value for function $\Pr ofit(IT_{acm},2)$. Results from this iteration are given in Table 2.

Table 2

Result of executing the eighth iteration of Stage 3 in the method of scenario analysis of functional requirements

Scenario number	Scenario representation at the knowledge level	Scenario value $S(IT_{acm_j})$	Scenario value $W(IT_{acm_j})$
1	$\{K^1_{UseCase}\}$	15	15
2	$\{K^2_{UseCase}\}$	10	10
3	$\{\varnothing\}$	–	–
4	$\{K^3_{UseCase},K^4_{UseCase}\}$	25	17
5	$\{K^5_{UseCase}\}$	23	23
Result of calculation of function $\Pr ofit(IT_{acm},r)$			
$\left\|IT_{acm_j}\right\|=5$	$r=2$	$\Pr ofit(IT_{acm},r)=0{,}0766$	

The matrix description of the architecture, taking into consideration the found duplication of the third and fourth scenarios, took the following form:

$$Arch=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&0&0&0\\0&0&0&2&1\\0&0&0&0&1\end{bmatrix}. \quad (10)$$

The found variant of the architecture description is the only and final result of implementing Stage 3 of the improved method for synthesizing the variants of architecture descriptions for a created IS.

When implementing Step 4.1 of the improved method for synthesizing the variants of architecture descriptions for a created IS one generates a variant of the description of the architecture for a created FM SW. In this scenario, the first, second and fourth functions of FM SW fulfill, respectively, the first, second and fifth requirement by a rightsholder. The third function of FM SW meets the third and fourth requirements by a rightsholder. The Use Case diagram for the scenario of meeting the third function of FM SW is shown in Fig. 7.

When implementing Step 4.2 it was revealed that none $t_{ij}(IT_{acm_i})=1$ was found in the resulting matrix $Arch_{base}$. Therefore, the set of publications of scenarios for fulfilling the functions of FM SW, shown in Fig. 2, 3, 6, 7, is adopted as the resulting variant of the architecture description for a created FM SW. Application of the improved method for synthesizing the variants of architecture descriptions for a created IS is over at Step 5.1.

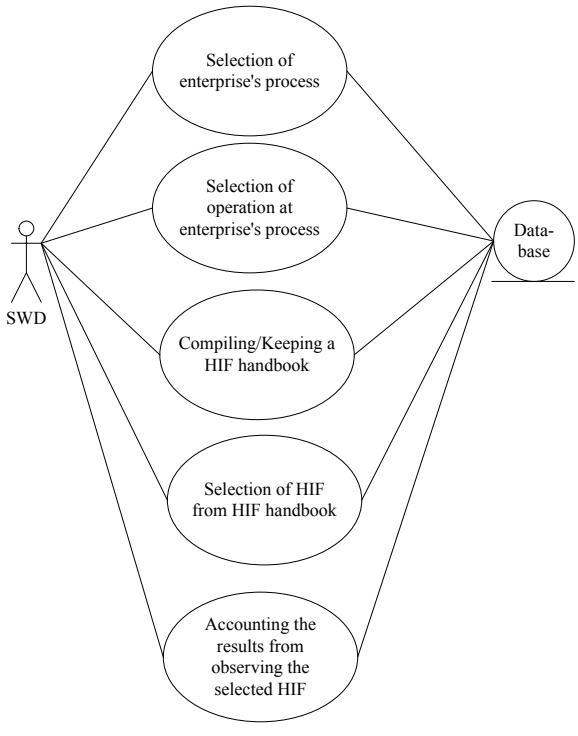

Fig. 7. The Use Case diagram for the scenario of fulfilling the third function in the functional module of safety at work

When implementing Step 5.2 one forms the resulting Use Case diagram of FM SW and applies the improved method for synthesizing the variants of architecture descriptions for a created IS for the following case

$$IT_{acm} = \{K^1_{UseCase}\} \cup \{K^2_{UseCase}\} \cup \{K^3_{UseCase}, K^4_{UseCase}\} \cup \{K^5_{UseCase}\}.$$

Based on the results of application of the method we revealed and eliminated the duplication of frames, describing actors "СБТ" and "Database", cases "Selection of data on an enterprise", "Selection of enterprise's process", "Selection of operation at enterprise's process", as well as respective interfaces.

When implementing Stage 6 we confirmed the choice of the constructed resulting Use Case diagram as the only variant for architecture description of FM SW.

When implementing Stage 7 we confirmed the selection of four main functions for a created FM SW in accordance with the results obtained during the execution of Step 5.1 (see the description of executing Step 4.1 of the improved method for synthesizing the variants of architecture descriptions for a created IS).

To validate a possibility to apply the results that have been described in the current study we compared the following:

a) the proposed improved method for scenario analysis of functional requirements;

b) the improved method for synthesizing the variants of architecture descriptions for a created IS reported in [19].

In this case, it was considered that the description of scenarios for fulfilling functional requirements is less detailed than the one used in [19] that describes functional requirements as sets of business classes and relationships between them.

When validating the improved method of scenario analysis, we revealed the duplication of scenarios for fulfilling the third and fourth functional requirements.

The description of the architecture for FM SW, based on the results of validation of the improved method for scenario analysis, corresponds to one of the two descriptions of architecture, formed as a result of the application of the improved method for synthesizing the variants of architecture descriptions for a created IS. It should be noted that application of the improved method for scenario analysis has not produced the descriptions of architecture for FM SW that would differ from those proposed in [19].

Consequently, the application of the improved method of scenario analysis makes it possible to find acceptable descriptions of IS architecture based on the significantly less detailed descriptions of the stated functional requirements. This will reduce the amount of time spent on the construction and analysis of functional requirements to IS, through the identification of overlapping requirements by rightsholders at the early stages of IS creation.

7. Discussion of the results of improving the model and the method of scenario analysis of functional requirements to an information system

In the course of the study, the following results were obtained for solving the stated tasks:

a) we have constructed a model of the subclass of structural patterns for designing the scenarios for the implementation of, functional requirements (3);

b) we have improved a method of scenario analysis of functional requirements to IS.

The model of the subclass of structural patterns for the design of scenarios for implementing functional requirements (3) was obtained by adapting the previously developed model of structural patterns for designing functional requirements (1) to features in the publication of scenarios for fulfilling functional requirements to IS in the form of the Use Case diagrams. Therefore, model (3) is a set of tuples that describe a Use Case diagram as a network of frames the relations between which in a given particular case bear no semantic load. Such a representation of patterns for designing scenarios for the implementation of functional requirements makes it possible to maximally separate the description of knowledge acquired from publications of functional requirements from the specific features of IS subject domains.

A given model is the base for the construction of a database (Fig. 1) for storing the knowledge, acquired from the Use Case diagrams, about scenarios for fulfilling individual functional requirements to IS. This should be considered the main advantage of model (3), since it makes it possible to implement this model and any methods based on it as a separate service in a scenario analysis of functional requirements to IS that would extend the possibilities of existing information technologies for the creation, analysis, and management of requirements to IS. An example of integrating a similar module by converting XML documents that describe the visual models of requirements to IS into a set of records in the database, storing the knowledge acquired from the functional requirements, can be found in [17].

It should also be noted that the proposed model of the subclass of structural patterns for designing the scenarios for fulfilling functional requirements (3), in contrast to the alternative techniques for solving the task of research, considered in chapter 2, was developed by using the maximally simple mathematical apparatus and does not require con-

ducting a scenario analysis to expand the Use Case diagrams with additional visual models.

Model (3) underlies the improved method of scenario analysis of functional requirements to IS. The essence of a given method is the automatic identification of overlapping scenarios for fulfilling functional requirements to IS and their individual frames (actors, cases, interfaces), as well as the automated synthesis of the resulting Use Case diagram describing the architecture for a created IS. These improvements have become possible due to the adaptation of the improved method for synthesizing the variants of architecture descriptions for a created IS to the peculiarities of the formal representation of about scenarios for fulfilling functional requirements. The result of these improvements is the significant decrease in the proportion of manual labor by analysts in the analysis of a set of the Use Case diagrams describing individual functional requirements, as well as in the synthesis of the resulting Use Case diagram. Owing to the use of model (3), the specificity of subject areas for created IS have almost no effect on the implementation of the improved method of scenario analysis of functional requirements.

A given method can be implemented as a set of SQL queries to the database constructed based on model (3). The result of such an implementation of a given method would be the materialized representation that stores data about the variants of description of the architecture for a created IS considering the remote overlapping functional requirements to this system.

However, the model and method proposed in the current study have certain flaws. The main drawback, limiting the application of the results obtained is the focus of the improved method of scenario analysis of functional requirements mainly on the analysis of symbol-based descriptions of individual elements in the Use Case diagrams. A given feature of model (3) and the improved method of scenario analysis of functional requirements requires that analysts should pay special attention to correct spelling of the names of elements that are repeated in different Use Case diagrams.

The results obtained define the need for the further research in this field. The purpose of the current study is to develop the models, methods, and technologies for the automated refinement of descriptions for the constructed functional requirements. One of the most promising ways of achieving a given goal, in our opinion, is the automated generation and subsequent refinement of descriptions of business classes for a functional requirement, based on the names for individual cases in the scenario for fulfilling a given requirement.

The further research in this field will inevitably require solving such tasks as the issues of homonymy and synonymy of terms in the descriptions of a subject area, requirements to IS, and IS functions. One of the simplest ways to address these issues is that an IT company should compile and update specialized dictionaries of terms using which could specify the publication of requirements to IS. However, the effectiveness of a given technique is low, which necessitates further research in this field.

8. Conclusions

1. We have adapted the formal representation of patterns for designing requirements to IS at the level of knowledge in terms of the features in the description of scenarios for fulfilling functional requirements by rightsholders in the form of the Use Case diagrams. It is shown that the Use Case diagram elements can be formally described as special cases of frames, the relationships between which have no their own semantics. Based on this feature, we have constructed a model of the subclass of structural patterns for designing scenarios for the implementation of functional requirements. A given model establishes the rules and semantics for the Use Case diagrams and their elements and enables the automated execution of a scenario analysis of the rightsholders' requirements using the previously developed elements of information technology.

2. We have suggested an improvement to the method of scenario analysis of functional requirements, which makes it possible to automate its most labor-intensive stages. In contrast to the base method, the improved method of scenario analysis makes it possible to automatically detect the overlapping scenarios and their individual frames (actors, cases, interfaces). In this case, the identification of overlaps is carried out both for individual scenarios and for the resulting Use Case diagram. A given advantage was achieved by selecting and subsequent processing of knowledge about scenarios for the fulfillment of requirements. We have verified the improved method of scenario analysis of functional requirements to IS based on the analysis of functional requirements to FM SW. In the course of verification, the duplication of scenarios for fulfilling the third and fourth functional requirements was revealed. In addition, we subsequently found the duplication of individual elements in the resulting Use Case diagram, which is a description of the architecture of FM. The results from comparing the proposed method to the previously constructed improved method for synthesizing the variants of architecture description for a created IS show that the proposed method makes it possible to find the acceptable descriptions of IS architecture based on the considerably less detailed descriptions of the statedd functional requirements.

References

1. Rukovodstvo k svodu znaniy po upravleniyu proektami (Rukovodstvo PMBOK). 5-oe izd. Newton Square: Project Management Institute, Inc., 2013. 586 p.
2. GOST R 57193-2016. Systems and software engineering. System life cycle processes (ISO/IEC/IEEE 15288:2015, NEQ). Moscow: Standartinform, 2016. 98 p.
3. Modelirovanie trebovaniy pol'zovateley. URL: https://docs.microsoft.com/ru-ru/visualstudio/modeling/model-user-requirements?view=vs-2015
4. Kobern A. Sovremennye metody opisaniya funktsional'nyh trebovaniy k sistemam. Moscow: Lori, 2002. 288 p.
5. Empirical research in requirements engineering: trends and opportunities / Ambreen T., Ikram N., Usman M., Niazi M. // Requirements Engineering. 2018. Vol. 23, Issue 1. P. 63–95. doi: https://doi.org/10.1007/s00766-016-0258-2

6. Requirements cybernetics: Elicitation based on user behavioral data / Liu L., Zhou Q., Liu J., Cao Z. // Journal of Systems and Software. 2017. Vol. 124. P. 187–194. doi: https://doi.org/10.1016/j.jss.2015.12.030
7. Asteasuain F., Braberman V. Declaratively building behavior by means of scenario clauses // Requirements Engineering. 2017. Vol. 22, Issue 2. P. 239–274. doi: https://doi.org/10.1007/s00766-015-0242-2
8. Yu Y.-J., Liu C. Little Model in Big Data: An Algebraic Approach to Analysing Abstract Software Behaviours // Ruan Jian Xue Bao/ Journal of Software. 2017. Vol. 28, Issue 6. P. 1488–1497. doi: http://doi.org/10.13328/j.cnki.jos.005229
9. Stowell F., Cooray S. The Appreciative System, Learning And Its Impact Upon Is Design // Communications of the Association for Information Systems. 2017. Vol. 40. P. 93–119. doi: https://doi.org/10.17705/1cais.04006
10. Ali N., Lai R. A method of requirements elicitation and analysis for Global Software Development // Journal of Software: Evolution and Process. 2017. Vol. 29, Issue 4. P. e1830. doi: https://doi.org/10.1002/smr.1830
11. Applications of ontologies in requirements engineering: a systematic review of the literature / Dermeval D., Vilela J., Bittencourt I. I., Castro J., Isotani S., Brito P., Silva A. // Requirements Engineering. 2016. Vol. 21, Issue 4. P. 405–437. doi: https://doi.org/10.1007/s00766-015-0222-6
12. Serna M. E., Bachiller S. O., Serna A. A. Knowledge meaning and management in requirements engineering // International Journal of Information Management. 2017. Vol. 37, Issue 3. P. 115–161. doi: https://doi.org/10.1016/j.ijinfomgt.2017.01.005
13. Kaiya H., Adachi K., Chubachi Y. Requirements Exploration by Comparing and Combining Models of Different Information Systems // Knowledge-Based Software Engineering: 2018. 2019. P. 64–74. doi: https://doi.org/10.1007/978-3-319-97679-2_7
14. Execution of natural language requirements using State Machines synthesised from Behavior Trees / Kim S.-K., Myers T., Wendland M.-F., Lindsay P. A. // Journal of Systems and Software. 2012. Vol. 85, Issue 11. P. 2652–2664. doi: https://doi.org/10.1016/j.jss.2012.06.013
15. Ievlanov M., Vasiltcova N., Panforova I. Development of methods for the analysis of functional requirements to an information system for consistency and illogicality // Eastern-European Journal of Enterprise Technologies. 2018. Vol. 1, Issue 2 (91). P. 4–11. doi: https://doi.org/10.15587/1729-4061.2018.121849
16. Levykin V., Ievlanov M., Neumyvakina O. Developing the models of patterns in the design of requirements to an information system at the knowledge level // Eastern-European Journal of Enterprise Technologies. 2017. Vol. 5, Issue 2 (89). P. 19–26. doi: https://doi.org/10.15587/1729-4061.2017.110586
17. Levykin V. M., Evlanov M. V., Kernosov M. A. Patterny proektirovaniya trebovaniy k informatsionnym sistemam: modelirovanie i primenenie: monografiya. Kharkiv: OOO «Kompaniya «Smit», 2014. 320 p.
18. Ievlanov M. Methods of presenting formulated requirements to the information system at the level of knowledge // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 4, Issue 3 (76). P. 4–11. doi: https://doi.org/10.15587/1729-4061.2015.47535
19. Yevlanov M. V. Improved method for synthesizing variants of description of the architecture of the created information system // Management Information System and Devices. 2018. Issue 175. P. 32–41.