DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY
Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2018, vol. 23, no. 1, pp. 5–11
doi: 10.2478/acss-2018-0001
https://www.degruyter.com/view/j/acss

Mapping of Activities for Object-Oriented System Analysis

Laima Leimane[1*], Oksana Nikiforova[2]
[1, 2] *Riga Technical University, Riga, Latvia*

***Abstract* – Even though object-oriented approach exists for more than 20 years, there are many issues regarding object-oriented system analysis: ambiguity, incompleteness and redundancy in requirements, difficulties with definition and traceability of non-functional requirements, requirements ignorant of business processes that are important for business operations and others. Although software can be engineered using many methodologies, different approaches to the analysis should be considered as they can provide a solution to the known issues. The paper describes the result of the research on object-oriented system analysis in a form of mapping activities offered in advanced software development methods.**

***Keywords* – Mapping, object-oriented system analysis, system analysis activities.**

I. Introduction

Object-oriented approach to system analysis, which emerged in the late 1980s, is considered to be an advanced paradigm for software engineering among others: Structured Design, Component Based Software Engineering, Agile Technologies etc. [1]. This approach sees the world as a set of classes, which contains objects with attributes and considers different class relationships. Listed object-oriented concepts are therefore modelled or one could simply put it – visualised in different diagrams, which characterise both the static and dynamic structure of a system. Unified Modelling Language (UML) is a notation that is usually used for this task [2].

Objects, classes, attributes, relationships are discovered through the process of analysis. It is a crucial and challenging step as it provides with requirements, which are used to develop software. Therefore, quality and accuracy of requirements are strongly linked to project's overall success [3]. As one of the main goals of object-oriented system analysis is to solve the real problem that created the need of building software, it is important to understand the big picture of organisation's needs [4]. One can achieve it by understanding the domain, business needs, identifying and specifying requirements, developing ideas for design and modelling structures for a system using UML notation [4]. Some researchers suggest using UML in conjunction with other notations such as Business Process Notation and Model (BPMN) and Decision Model and Notation (DMN) as these notations can be used to specify requirements for the system [5]. Organisation's business processes and decision logic should be considered when specifying requirements as they are fundamental to organisation's existence [6]. BPMN and DMN can help to understand the existing business processes and the way the decisions are made [5]. Others even suggest using Data Flow Diagrams (DFD) as these diagrams could be easily understood by the customer and Entity Relationship (ER) diagrams, which can be used as an addition to UML Use Case diagram to demonstrate the static structure [7].

Requirements can be gathered by conducting interviews with customers, creating surveys, analysing the existing business processes, other relevant data and structuring results in such forms as use cases, user stories, software requirements specification, activity diagrams, sequence diagrams etc. [6]–[8]. Thus, it can be concluded that methods to elicit, analyse, specify requirements vary. The same applies to methodologies used in projects. Since the introduction of Agile model in the 2000s, heavy-weight methodologies are often substituted with light-weight Agile: Scrum, XP (extreme programming), Lean software development, crystal family and feature driven development [1]. Agile methodologies deliver frequent but partial functionality, which can be evaluated and improved in consequent iterations and, therefore, develop higher-quality software faster compared to traditional methods [9]. Nevertheless, the specific Agile or traditional method – the analysis and requirements specification are performed. In case of Agile methods, requirement elicitation and specification have no structure and yet are important for success of a project [6]. Some researchers have attempted to provide a methodology or described activities, which could and should be performed in this step. Few authors have pointed out issues regarding non-functional requirement definition, traceability as functional requirement change impacts non-functional requirements [6], [10]. Other researchers have developed tools used to clarify, address inconsistency, incompleteness in natural language-based requirements using ontologies [11]. Apart from the syntactic and semantic analysis of requirements, ambiguity can be addressed making decisions regarding quality, preference, classification and property [6]. It is evident that there are existing issues in requirements specification and as stated above, the analysis is performed in both traditional and agile methodologies. Incorrect requirements can result in software with no value to the

* Corresponding author's e-mail: laima.leimane@edu.rtu.lv

customer. Therefore, research on activities in the object-oriented system analysis can provide with a set of currently identified issues and possible solutions. The goal of the paper is to summarise a list of activities offered by advanced software development methods by mapping them into a table, where differently named activities are presented at the same level by their implementation essence.

The paper is structured as follows. In the second section, an overview of existing advanced approaches to the object-oriented system analysis is provided. The third section presents the created mapping tables. The fourth section provides information about the related work. Finally, the fifth section contains authors' conclusions as well as covers several areas of further research.

II. Advanced Approaches to Object-Oriented System Analysis (2008–2017)

In order to collect data about activities in the object-oriented system analysis, research was conducted focusing on the period of 2008–2017. Keywords such as object-oriented system analysis, agile system analysis, scrum, requirements specification were used across different scientific databases containing books, papers and scientific journals. Fourteen sources are considered in the present paper.

Almost every source mentions either Use Cases or User Stories, or conjunction of both. Modelling such structures and leaving other models as an optional choice are characteristic of Agile methodologies [12], [13], [8]. Some suggest identifying Epics – large User Stories – and dividing them into smaller ones [12], [8].

Regarding the previously mentioned methodologies, Q. Wei et al. offer methodology for Agile requirement analysis, which consists of several activities [9]:

- Establishing a use case model;
- Determining system boundary and range;
- Recognising actors;
- Determining use cases for each actor;
- Defining relations between use cases and actors;
- Converting a use case to a user story;
- Making a product backlog.

Last step might not be considered a part of object-oriented system analysis, but it prioritises and can affect requirements specification. Additionally, the authors state that system static structure can be modelled with UML Class diagram, and dynamic behaviour can be modelled with UML sequence, activity, state and collaboration diagrams. Similarly, A. Zeaaraoui et al. offer requirement elicitation, creating Use Cases, User Stories and modelling Sequence diagrams, and additionally suggest identifying entities from a text, which indicates that a business or domain description is needed and propose using a specific template for creating user stories [3]:

"As a <role> I can <activity> so that <business value>", where

- Role – who (person, system) is performing the action or receiving the value from the activity.
- Activity – action to be performed by the system.
- Business Value – value to the business justification.

This form is simple and can assist in recognising actors, their role and value of the activities. T. Tenso et al. offer a similar template using different terminology [8].

S. Dragicevic et al. [6] provide a detailed description of Method for Elicitation, Documentation and Validation of Software User Requirements (MEDoV), which can be used for agile and lean methodologies, and claim to eliminate needless features and ease improvement and maintenance [6]. MEDoV method consists of several phases where the first one identifies business goals and assigns at least one Key Performance Indicator (KPI) to every functional and non-functional requirement, thus ensuring that objectives are measurable. The second phase deals with requirement elicitation, analysis and documentation, which include defining conventions, business process modelling, business rule extraction, model creation and cataloguing user comments. Finally, the third phase includes analysis and potential conflict resolution, requirements specification, "As-Is" and "To-Be" models, and requirement validation [6].

M. Kundi et al. [11] proposed a novel approach to requirement ambiguity by using FrameNet frames and, therefore, analysing natural-language requirements from semantic point of view. It is done by selecting lexical units from Use Case descriptions and matched against FrameNet lexical units. As a result, roles are involved in frames, concept synonyms and homonyms are presented and conceptual completeness of use case can be checked, missing entities, relationships can be identified [11]. This shows that semantics and terminology can greatly impact requirements. If understanding of vocabulary used in describing a requirement is not the same among all stakeholders, then expectations may greatly differ leading to possible conflicts when software is presented. A.P. Murray [14] suggests that as not all stakeholders have identical understanding of concepts, demos in a form of black & white screens, mockup sites should be created in parallel to development so that expectations are managed and changes are made early in development rather than later when it is costlier [14]. Similarly, J. Chen et al. suggest involving a customer in the process by developing scenarios with an "on-site customer" [13].

As mentioned above, non-functional requirement definition and traceability are issues, which can lead to project failure due to the fact that a customer does not understand their importance [6]. Non-functional requirements include those needs regarding performance, security, portability and usability, and one must have specific knowledge regarding these subjects to be able to elicit, analyse and specify such requirements. A. Silva et al. offer a guide – ADEG-NFR that includes a set of questions, templates and examples of requirements. Formulated in terminology understandable to a customer makes this task easier [15]. Traceability issue is described and addressed by A. F. B. Arbain et al. by presenting a Traceability Process Model (TPM) that could be improved and developed [10].

It is evident that there are many approaches and activities involved in the object-oriented system analysis. The subsequent section contains a mapping table where activities found in various sources are listed and generalised by their essence.

III. Mapping Table

Discovered activities in the object-oriented system analysis are listed in two tables and written in a similar form: elicit, document, specify, create, define etc. Columns represent authors and activities mentioned. The authors divided activities in two groups based on activity count:

- Activity count 1–6;
- Activity count 9–31.

Activity count varies, which impacts the appearance of tables.

TABLE I

Mapping of Activities (1)

	ABBASABADI ET AL. 2008	CHEN, WU 2015	KLUZA ET AL. 2017	KUNDI, CHITCHYAN 2017	SHIM, LEE 2017	SILVA ET AL. 2017	ZEAARAOUI ET AL. 2013
1.	Identify and model entities and attributes, and determine relationships between them in a Class diagram	Explore / gather requirements as user stories	Model systems structure (Class, Deployment, Object etc. diagrams)	Elicit requirements	Write requirements as a hypothesis and continuously validate them	Elicit non-functional requirement using a guide containing questions, templates and examples	Elicit requirements
2.	Visualise the exchange of messages between a set of objects in a Sequence diagram	Develop scenarios with an "on-site customer"	Model systems behaviour (Activity, Use Case, State Machine etc. diagrams)	Specify requirements	Recognise internal (company) and external (customer) goals	Elicit functional requirements	Create User stories
3.	Specify and visualise object's states and transitions between states an object can go through during its lifetime in a State Machine diagram	Add expected system input / output data to each scenario, thus creating testable acceptance test cases	Design business process models (with Business Process Model and Notation (BPMN))	Perform requirement semantic and syntactic analysis on written requirements to provide complete, consistent and clear requirements	Create User stories as requirements		Create Use cases
4.	Identify and visualise a control flow from activity to activity in Activity diagram	Assemble the obtained test cases in a user manual	Model human decision logic	Model Use case diagram	Identify customers or stakeholders		Model Sequence diagram
5.	Determine and visualise activities among objects in Activity diagram	Identify class, method names	Model requirements for automatisation of decision-making		Analyse relative customer importance according to their level of interest and influence		Identify entities from a text
6.	Model system's behaviour with actors, use cases and their relationships in a Use case diagram		Model business rules (with Decision Model and Notation (DMIN))				Use form for user stories: As a <role> I can <activity> so that <business value>

TABLE II

MAPPING OF ACTIVITIES (2)

	BIK ET AL. 2017	KUNG, LEI 2016	MURRAY 2016	DRAGICEVIC ET AL. 2014	TENSO ET AL. 2017	WEI ET AL. 2014	ABDULAEV 2016
1.	Create user stories	Understand the application	Identify the Key Stakeholder	Elicit requirements	Identify requirements	Establish a use case model	Analyse requirements
2.	Ensure user story quality	Elicit requirements	Identify a Project sponsor	Determine non-functional requirements	Analyse requirements	Analyse software requirements	Specify requirements
3.	Prioritise user stories	Specify requirements	Identify users	Use business process models for requirements elicitation	Document requirements	Determine system boundary and range	Validate requirements
4.	Determine User story acceptance criteria	Identify, formulate and understand requirements	Conduct interviews	Determine whether every requirement is non-redundant, concrete and understandable	Validate requirements	Recognise actors	Identify domain
5.	Conduct interviews	Develop design ideas	Document requirements	Determine which requirements should be considered for next release	Create User stories	Determine Use cases for each actor	Analyse a domain
6.	Gather / elicit requirements	Construct models	Create Demos in a form of black and white screens, mockup sites	Classify requirements depending on which topic, component, team requirement belongs to	Identify Epics	Define relations between use cases and actors	Create a system description
7.	Validate user stories	Perform Domain / conceptual modelling	Validate requirements	Document requirements	Create the hierarchy of functional goals	Convert use cases to user stories	Describe graphical user interface
8.	Identify Epics (large user stories)	Collect information about an application domain	Interview stakeholders	Identify business processes and functions, and related project objectives	Attach quality (non-functional) goals and roles to functional goals	Describe a static structure with Class diagrams (optional)	Specify functional constraints for the system
9.	Estimate and select user stories	Identify domain concepts	Collect requirements	Identify the known risk factors	Write user stories using format: As a <user performing a certain role>, I need <to perform action> to support <achieving a certain good>	Model dynamic characteristics with sequence, activity, state, collaboration diagrams (optional)	Specify system design constraints

	BIK ET AL. 2017	KUNG, LEI 2016	MURRAY 2016	DRAGICEVIC ET AL. 2014	TENSO ET AL. 2017	WEI ET AL. 2014	ABDULAEV 2016
10.	Improve Epics	Classify domain concepts into modelling concepts (classes, attributes, relationships)	Refine requirements	Define at least one KPI (Key Performance Indicator) for every requirement			Validate completeness, correctness, unambiguity and manageability of every requirement
11.	Split Epics into smaller ones	Visualise the classified concepts into Class diagram		Analyse requirements			Determine if it is possible to rank requirements according to importance and/or stability
12.		Conduct User surveys and interviews		Create High-level process description			Describe system reaction in unexpected situations / when unexpected input encountered
13.		Model Use cases		Define conventions (modelling, graphic, naming etc.)			Visualise a data flow (system converting input into output) in a Data Flow Diagram. Addition to other diagrams.
14.				Determine organisational relationships			Create ER (Entity Relationship) diagram to visualise structure. Addition to Use Case diagram
15.				Determine how and which documents and data are used in process and what are their relationships			
16.				Define domains			
17.				Model process logic			

As mentioned previously, S. Dragicevic et al. [6] offer a detailed description of MEDoV method. Overall, 31 activities are identified, of which seventeen are presented in the mapping and fourteen are listed below:

- Determine how and which IT systems and services are used;
- Model collected documents and data into a UML Class diagram;
- Define business rules;
- Describe actual decision logic;
- Model Use Case diagram;
- Model a business process flow in Activity diagram;
- Catalogue suggestions and user remarks;
- Clarify requirement ambiguity;
- Detect if requirements have flaws and missing parts;
- Test requirement feasibility;
- Identify legal regulation risks;
- Model "As-Is" and "To-Be" models;
- Validate correctness, completeness, consistency, realism and verifiability of every requirement;
- Resolve conflicting requirements.

Activities are divided in order to present a more readable and consistent representation of data as a description of this method is particularly detailed in distinction of every step compared to other sources.

IV. Related Work

Researchers devote attention to the comparison of methods for the object-oriented system analysis since their appearance. For example, Yuan, Patel [17] provided a comparative study of the object-oriented approach by evaluating object-oriented methodologies and exploring future object-oriented research directions. Six representative object-oriented methodologies were evaluated based on their capabilities in aiding object-oriented analysis and design [17]. M. Yusufu et al. offered a comparative study on formal methods used to specify requirements including notations: UML, Z language, Petri nets, B method and action systems [18]. A paper written by N. Pavlova and O. Nikiforova [19] provided an overview and comparison of activities necessary for analysis according to Rational Unified Process (RUP), Object Modelling Technique (OMT), Visual Modelling Technique (VMT) and Larman strategy [19]. The paper highlights issues, which still exist in the object-oriented system analysis. One of them is eliciting requirements while considering business processes and making quality decisions and prioritisation. Therefore, creation of user stories and scenarios is a complicated process. As previously discussed in the present paper, almost every author mentions or describes modelling use cases and user stories, which indicates that this topic is not entirely clear taking into account that the paper by Pavlova and Nikiforova was written more than ten years ago.

N. M. A. Munassar provided a comparative study on the object-oriented approach and traditional approaches, and in conclusion emphasised the importance of understanding requirements in the object-oriented approach that might be not the case for other models [20]. Therefore, the need for understanding the domain and goals was established.

M. Kassab researched this topic both in a form of empirical study [21] and as a comparison of changing industrial practices in 2003, 2008 and 2013 [22]. Both studies were conducted using surveys and data analysis. In the first study, requirement gathering techniques, requirement analysis techniques, requirement formalism, requirement inspection techniques as well as attitude (unsatisfaction, neutral, satisfaction) towards requirement engineering and its phases among 247 participants were reported [21]. Participants were divided into those who used Waterfall and those who used Agile methodologies. Their respective responses were compared. In the second study, data from surveys were analysed, and it was compared how requirement gathering techniques, requirement analysis techniques, requirement presentation, requirement inspection techniques and prototyping usage changed over the years regardless of the methodology used [22]. Most participants (2003 – 80 %, 2008 – 70 %, 2013 – 61 %) described their role as a programmer/developer, software/system engineer, tester, and a lesser percentage (2003 – 20 %, 2008 – 30 %, 2013 – 39 %) identified themselves as project managers, architects or analysts [22]. Most participants were not considered to be analysts and might fulfil this role in conjunction with their position as a developer, tester or other position. It is possible that some participants might have very little to no involvement in some activities regarding the requirement analysis.

To the authors' best knowledge, in the recent past a similar mapping table has not been provided, although this topic has been researched in the previously described form. The presented mapping table can serve as a template to create a survey for a similar comparative study and analyse the gathered data against findings of others.

V. Conclusion And Further Research

In the present paper, the authors have summarised research of activities performed in the object-oriented system analysis that can be used as a reference to work conducted by others. As a result, a mapping in a form of two tables containing activities found in 14 sources has been created. This mapping can be used to further investigate the current situation and suggest new or hybrid methods. Consequently, conclusions can be made and solutions presented.

The presented mapping provides evidence that:

- There are various approaches and, therefore, there is no unified approach;
- There are several activities that are mentioned in most sources;
- Some approaches focus on solely semantic and syntactic aspects of natural-language requirements;
- Some activities are uniquely mentioned in one source;
- Both commonly and rarely referenced methods can be used in requirement step prioritisation;

- The determined steps can serve as a set of minimal activities required in order to perform the object-oriented system analysis;
- The acquired set can be further used to survey an expert group and analyse the collected data.

The authors intend to further investigate and compare the obtained results. Furthermore, based on the presented mapping, research on currently recognised and applied methods of the object-oriented system analysis in Software Engineering projects is to be carried out. These are some of the goals of further research.

References

[1] A. Verma, I. Kaur, and N. Arora, "Comparative analysis of software engineering paradigms" in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 994–999.

[2] M. P. Abbasabadi, M. P. Abbasabadi, A.A.P. Kazem, "Recognizing Gap between UML Design Model and Its Implementation," in *International Symposium on Information Technology, 2008. ITSim 2008*, 2008, pp. 76–85. https://doi.org/10.1109/itsim.2008.4631570

[3] A. Zeaaraoui, Z. Bougroun, M. G. Belkasmi, and T. Bouchentouf, "User Stories Template for Object-Oriented Applications," in *Third International Conference on Innovative Computing Technology (INTECH)*, 2013, pp. 407–409. https://doi.org/10.1109/intech.2013.6653681

[4] D. Kung and J. Lei, "An Object-Oriented Analysis and Design Environment," in *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, 2016, pp. 91–100. https://doi.org/10.1109/CSEET.2016.20

[5] K. Kluza, P. Wiśniewski, K. Jobczyk, A. Ligęza, and A. Suchenia (Mroczek), "Comparison of selected modeling notations for process, decision and system modeling," in *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2017, pp. 1095–1098. https://doi.org/10.15439/2017f454

[6] S. Dragicevic, S. Celar, and L. Novak, "Use of Method for Elicitation, Documentation and Validation of Software User Requirements (MEDoV) in Agile Software Development Projects," in *2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, 2014, pp. 65–70. https://doi.org/10.1109/cicsyn.2014.27

[7] V. I. Abdulaev, Programmnaja inzhenerija: uchebnoe posobie, Ch. 1. *Proektirovanie sistem*. Russia: PGTU, 2016.

[8] T. Tenso, A. H. Norta, H. Rootsi, K. Taveter, and I. Vorontsova. "Enchancing Requirements Engineering in Agile Methodologies by Agent-Oriented Goal Models," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 268–275. https://doi.org/10.1109/rew.2017.24

[9] Q. Wei, G. Danwei, X. Yaohong, F. Jingtao et al., "Research on Software Development Process Conjunction of Scrum and UML modeling," in *2014 Fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 2014, pp. 978–982. https://doi.org/10.1109/imccc.2014.206

[10] A. F. B. Arbain, I. Ghani, and W. M. N. W. Kadir, "Agile non functional requirements (NFR) traceability metamodel," in *2014 8th Malaysian Software Engineering Conference (MySEC)*, 2014, pp. 228–233. https://doi.org/10.1109/mysec.2014.6986019

[11] M. Kundi and R. Chitchyan, "Use Case Elicitation with FrameNet Frames," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 224–231. https://doi.org/10.1109/rew.2017.53

[12] N. Bik, G. Lucassen, and S. Brinkkemper, "A Reference Method for User Story Requirements in Agile Systems Development," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 268–275. https://doi.org/10.1109/rew.2017.83

[13] J. J.-Y. Chen and M. M.-Z. Wu, "Integrating extreme programming with software engineering education," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 577–582. https://doi.org/10.1109/MIPRO.2015.7160338

[14] A. P. Murray, *The Complete Software Project Manager: Mastering Technology from Planning to Launch and Beyond*. USA: John Wiley & Sons, Incorporated, 2016.

[15] A. Silva, P. R. Pinheiro, A. Albuquerque, and J. Barroso. "Evaluation of an approach to define elicitation guides of non-functional requirements," IET Software. Vol. 11., pp. 221–228, Sept. 2017. https://doi.org/10.1049/iet-sen.2016.0302

[16] W. Shim and S.-W. Lee, "An Agile Approach for Managing Requirements to Improve Learning and Adaptability," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 435–438. https://doi.org/10.1109/rew.2017.46

[17] G. Yuan and N. Patel, "An exploration of object-oriented methodologies for system analysis and design," Workshop on Studies of Software Design, WSSD 1993: Studies of Software Design, Lecture Notes in Computer Science book series (LNCS, volume 1078), 1996, pp 164–188. https://doi.org/10.1007/bfb0030528

[18] M. Yusufu and G. Yusufu, "Comparison of Software Specification Methods using a Case Study," in *2008 International Conference on Computer Science and Software Engineering*, 2008, pp. 784–787. https://doi.org/10.1109/csse.2008.1493

[19] N. Pavlova and O. Nikiforova, "An Overview of Advanced Approaches for Construction of Platform-Independent System Model," Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, Riga, Latvia, 2005, pp. 156–168.

[20] N. M. A. Munassar and Dr. A. Govardhan, "Comparison study between Traditional and Object-Oriented Approaches to Develop all projects in Software Engineering," in *2011 5th Malaysian Conference in Software Engineering (MySEC)*, 2011, pp. 48–54. https://doi.org/10.1109/mysec.2011.6140642

[21] M. Kassab, "An Empirical Study on the Requirements Engineering Practices for Agile Software Development," in *2014 40th Euromicro Conference on Software Engineering and Advanced Applications*, 2014, pp. 254–261. https://doi.org/10.1109/seaa.2014.77

[22] M. Kassab, "The Changing Landscape of Requirements Engineering Practices over the Past Decade," in *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)*, 2015, pp. 1–8. https://doi.org/10.1109/empire.2015.7431299

Laima Leimane received the Bachelor degree in computer systems from Riga Technical University, Latvia, in 2015. At present, she is the second-year Master student at the Department of Applied Computer Science, Riga Technical University.
E-mail: laima.leimane@edu.rtu.lv

Oksana Nikiforova received the Doctoral degree in information technologies (system analysis, modelling and design) from Riga Technical University, Latvia, in 2001.
She is a Professor at the Department of Applied Computer Science, Riga Technical University, where she has been working since 1997. Her current research interests include object-oriented system analysis, design and modelling, especially the issues in Model Driven Software Development.
E-mail: oksana.nikiforova@rtu.lv
ORCID iD: https://orcid.org/0000-0001-7983-3088