UDC 004.415.2.045 (076.5)

# FORMAL FOUNDATIONS FOR SOFTWARE MODEL TO MODEL TRANSFORMATION OPERATION

*O.V. Chebanyuk*

Software model transformation operations are central operations in Model-Driven approaches. In order to represent software models, graphical modeling notations, for example UML, are used. Quality of software model, obtained after transformation, influences on further operations with this model. Thus, it is important to design formal approaches for model to model transformation that are grounded on analytical and mathematical tools. These approaches should provide a background for flexible adopting software model transformational techniques for peculiarities of specific software development lifecycle model.
Challenges to mathematical tools and transformation rules that are involved to designing of model to model transformation approaches are formulated in this paper. The ground of mathematical tools choice that is based on these challenges is performed.
An approach for performing model to model transformation, which is based on graph transformation, is presented in this paper. Transformational operations are considered on meta-level and concrete level. On meta-level choosing of mathematical tools for representing of transformation stages and transformational artifacts are grounded. Software models are represented as graphs. Initial information for transformation is represented as a set of sub-graphs. Transformation rules are composed using second and first order logics. On the level of the first-order logic all software model elements that participate in transformation are considered. In the level of second-order logic transformation rule considers types of software model element that are participate in the transformation.
Proposed approach is extensible and may be used for extend functionality of model to model tools that process software models. For example in MEDINI QVT there is no direct ways to compose a model to model transformation rule that considers those software models elements that have no direct links.
Key words: Software Model, Software Model Transformation, Graph Transformation, Model-Driven Development, First-order logic, Second-order logic, UML.

Операції перетворення моделей програмного забезпечення є центральними операціями у модельно-орієнтованих підходах розробки програмного забезпечення. Для представлення моделей програмного забезпечення використовуються графічні нотації мов моделювання, наприклад UML. Якість отриманої моделі програмного забезпечення після трансформації визначає ефективність операцій подальшої її обробки. Це визначає актуальність завдання розробки нових формальних підходів для трансформації моделей програмного забезпечення. Такі підходи забезпечують підгрунття для гнучкої адаптації технік та підходів трансформації з урахуванням особливостей процесів життєвого циклу програмного розробки програмного забезпечення.
У статті сформульовані вимоги до аналітичних інструментів та правил трансформації які застосовуються для розробки підходів трансформації моделей. Також наведено обгрунтування вибору аналітичних інструментів, що відповідають цим вимогам.
У роботі представлено підхід проведення операції трансформації моделі в модель, який базується на графовому перетворенні. Опреації трансформації розглядаються на мета рівні та на рівні детального опису моделей. Вихідною інформацією для трансформації слугує множина під-графів. Правила трансформації задаються за допомогою логік першого і другого порядку. На рівні преставлення елементів моделей програмного забезпечення для задання правил трансформації використовується логіка першого порядку, на рівні опису типів елементів використовується логіка другого порядку.
Представлений підхід є розширюваним та може використовуватися при модифікації існуючих середовищ перетворення моделей. Наприклад у MEDINI QVT відсутня можливість сформувати правила трансформації моделей, що включають елементи, які не зв'язані безпосередньо на UML діаграмі.
Ключові слова: модель програмного забезпечення, трансформація моделей програмного забезпеченняб графова трансформація, розробка, що керується моделями, логіка першого порядку, логіка другого порядку, UML.

Операции преобразования моделей программного обеспечения являются ключевыми в модельно-ориентированных подходах разработки программного обеспечения. Для представления моделей программного обеспечения используются графические нотации языков моделирования, например UML. Качество полученной модели программного обеспечения после трансформации определяет эффективность ее дальнейшей обработки. Это определяет актуальность задачи разработки новых формальных подходов для трансформации моделей программного обеспечения. Такие подходы обеспечивают основу для гибкой адаптации техник и методов трансформации с учетом особенностей процессов жизненного цикла разработки программного обеспечения.
В статье сформулированы требования к аналитическим инструментам и правилам трансформации, которые применяются для разработки подходов трансформации моделей. Также приведено обоснование выбора аналитических инструментов, отвечающих этим требованиям.
Представленый в статье подход преобразования из модели в модель, основанный на графовой трансформации. Опреации трансформации рассматриваются на общем уровне и на уровне детального описания моделей. Исходной информацией для трансформации служит множество под-графов. Правила трансформации задаются с помощью логик первого и второго порядка. На уровне элементов моделей программного обеспечения для задания правил трансформации используется логика первого порядка, на уровне описания типов элементов используется логика второго порядка.
Представленный подход является расширяемым и может использоваться при модификации существующих сред преобразования моделей. Например, в MEDINI QVT отсутствует возможность сформировать правила трансформации моделей, включающих элементы, которые не связаны непосредственно на UML диаграмме.
Ключевые слова: модель программного обеспечення, трансформация моделей программного забезпечення, графовая трансформация, разработка, управляемая моделями, логика первого порядка, логика второго порядка, UML.

## 1. Introduction

Software model transformation operations are central operations in Model-Driven approaches. This is explained by the following facts:

- the stakeholders should use information from previous software development stages;

- different types of UML diagrams (UML 2.5) for successfully performing of specific tasks of some software development process.

Each type of UML diagram covers different scale of software project and expresses specific part of software project (structure or behavior). Automatizing of transformation operations lets quickly obtain software model, containing information from previously generated models. Also it can facilitate other model processing information, namely model refactoring, merging, comparing and others, i.e. transformations with the same type of software models.

In order to achieve this goal the following model transformation activities are implemented: Model to Model (M2M), Model to Text (M2T), and Text to Model (T2M) transformations. Text in M2T and T2M transformations means analytical representation of software model or skeleton of program. In the first case, the role of text is to be subsidiary artefact that saves information about model. In the second case the role of text is to be target of transformation (Truyen, 2006).

Other transformation' aspects are horizontal and vertical software model transformations. A horizontal transformation is a transformation where the source and target models reside at the same abstraction level. Typical examples are refactoring (an endogenous transformation) and language migration (an exogenous transformation). A vertical transformation is a transformation where the source and target models reside at different abstraction levels. A typical example is refinement, where a specification is gradually refined into a full-fledged implementation, by means of successive refinement steps that add more concrete information. Also code generation operations are considered as vertical software model transformation (Czarnecki and Helsen, 2006).

Many papers, proposing strong contribution in model to model transformational approach, consider transformational tasks, relating to concrete transformational languages (for example QVT, ATL or ATLAS). Also, transformational rules are implemented, by means of concrete transformational environments (for example MEDINI QVT). Respectively, transformational results are visualized in concrete modeling environments (for example eclipse or visual studio) and software models are represented in concrete formats (XML or XMI).

Such approaches depend on possibilities of concrete tools, formats or transformational languages. Variety of transformational operations is limited by supported features of chosen practical tools.

From other side, development of analytical approaches let avoiding transformation environment limitations and composing transformational rules with different levels of complexity. When operation with software models are defined, proper math apparatus for performing them is chosen. If one math approach can not satisfy all requirements, additional operations with this approach are introduced, for example, creating of new algebras of spreading existing ones. Other way is to define rules for transforming of one analytical representation to another.

Analytical transformation approaches let to provide a background for improving existing and designing new transformational languages, environments and tools. Also additional operations of software model processing such as model verification, validation may be described. These operations may be integrated into transformational approaches to estimate software model before or after transformation.

Consequence: designing foundation for software model transformation lets support flexible adoption of transformation techniques to business needs.

The paper is organized as follows: section two represents related works, section three contains task and challenges to transformational approach, section four describes proposed approach and choosing of analytical background for performing transformation operation. The last, fivth section represents conclusion and further researches.

## 2. Related papers

Papers, related to software model transformation, can be divided to two classes, namely those which make strong contribution in transformation techniques and those that develop analytical tools for designing new and improving existing transformational approaches and techniques.

Detail review of papers, devoting to designing transformation methods and techniques grounded on practical tools and environments is represented in paper (Chebanyuk and Markov, 2016). The result of this review is summarizing achievements of researches according Model-Driven Engineering (MDE) promising. List of MDE promising is also represented in paper (Chebanyuk and Markov, 2016). Analyzing this review, the requirements to analytical automated method for model to model transformations, that cover all MDE promising, were formulated.

Also represent review of papers, making strong contribution in development of transformational techniques.

Paper (Greiner et al, 2016) represents a case study dealing with incremental round-trip engineering of UML class models and Java source code.

Described approach tries to prevent information loss during round-trip engineering by using a so called trace model which is used to synchronize the Platform Independent and the Platform Specific Models. Furthermore, the source code is updated using a fine grained bidirectional incremental merge. Also, information loss is prevented by using Javadoc tags as annotations. Case model and code are changed simultaneously and the changes are contradicting, one transformation direction has to be chosen, which causes that some changes might get lost (Greiner et al, 2016).

The contribution of the survey (Seifermann and Groenda, 2016) is the identification and classification of textual UML modeling notations. During the survey, authors found a total of thirty one textual UML notations.

The classification is aimed to include the user's point of view and support the notation selection in teams. In total, authors found 14 new notations compared to previous surveys: Alf, Alloy, AUML, Clafer, Dcharts, HUTN, IOM/T, Nomnoml, pgf-umlcd, pgf-umlsd, tUML, txtUML, UML/P, and uml-sequence-diagram-dsl-txl. Authors presented each of the twenty categories in detail including objectively checkable conditions that cover the level of UML support, the editing experience, and the applicability in an engineering team.

Paper (Wu, 2016) addresses the issue of generating metamodel instances satisfying coverage criteria.

More specifically, this paper makes the following contributions:

- A technique that enables metamodel instances to be generated so that they satisfy partition-based coverage criteria.
- A technique for generating metamodel instances which satisfy graph properties.

A metamodel is a structural diagram and can be depicted using the UML class diagram notation. Thus, the coverage criteria defined for UML class diagram can also be borrowed for metamodels. To facilitate the transformation from class diagrams with OCL constraints to Satisfiability Modulo Theories (SMT) formulas authors use a bounded typed graph as an intermediate representation (Wu, 2016).

Paper (Natschlager et al, 2016) presents concept for Adaptive Variant Modeling (AdaVM). AdaVM is a part of the AdaBPM framework for advanced adaptability and exception handling in formal business process models. In addition, AdaVM considers linking of elements, propagation of changes, and visual highlighting of differences in process variants. Authors showed that graph transformation techniques are well-suited for process variant management and that variants can be automatically created by a few graph transformation rules specifying concrete variations. Authors show that the adaptable approach is less complex regarding the type graph, source graphs, and the number of rules and application conditions. New ideas, expressed in proposed approaches, are (i) the support of variability by restriction and by extension with graph transformation techniques, (ii) linking and propagation of changes, (iii) individual blocking of elements/attributes, and (iv) visual highlighting of differences in process variants.

A review of mathematical foundations for providing realization of model transformation techniques is outlined in (Rabbi et al, 2016).

A review of metamodeling tools is represented in paper (Favre and Duarte, 2016) and several metamodeling frameworks are described.

## 3. Task

To define steps to perform model to model transformational operation and provide the following analysis for every step:

1. Propose analytical tools for describing operations that are performed in every step.
2. To involve formal tools for representation of transformational rules and software models used in this step.

**Challenges for mathematical tools**

For software model representation:

- support both compact and detailed software model representation;
- allow flexible choosing set of diagram notation elements that participate in transformation;
- be convenient for model proceeding (analysis of structure, comparing, merging and so on);
- provide easy machine processing;
- be convenient for cognitive human perception (Chebanyuk and Markov, 2015).

For transformational rules:

- support both compact and detailed transformational operations;
- allow matching elements of compact and detailed view;
- be compatible with representation of rules in natural language. Namely reflect all transformational conditions and details of transformational process.

## 4. Proposed approach

This work continues investigations, started in papers (Chebanyuk, 2014; Chebanyuk and Markov, 2016). In paper (Chebanyuk, 2014) the method for behavioural software model synchronization was proposed. This method is grounded on software model transformation.

To perform successfully software model to model transformation operation it is proposed to use the principle of graph transformation (IBM, 2016). The idea of this principle is that elements of entire software model should be linked between each other. Other words if software model is a graph then parts for transformation should be expressed as sub-graphs. After applying transformation rules set of sub-graph is obtained.

To develop this approach it is necessary to propose:

- software model representation approach that is based on graph representation;
- formal representation of transformation rules, that is based on first and second order logics.
- set of rules, related to performing of transformational operations, in every transformation stage, namely:
  - defining sub-graphs for transformation from initial software model;
  - performing transformation operation, forming a set of resulting sub-graphs.

## 4.1 Software Model Representation Approach

Denote software model(SM) as: SM(O,L), where

O – a set of software model objects. Objects are elements of software model (SM) notations that can be expressed as graph vertexes.

L – a set of links between O, that can be expressed as graph edges. Links are elements of software model notation that can be expressed as edges.

As in transformation operation there are two software models define them as initial ( $SM_{initial}$ ) and resulting ( $SM_{resulting}$ ). $SM_{initial}$ is a software model from which transformation is started. This model contains initial information for transformation. $SM_{resulting}$ is the model which is obtained after transformation. Thus:

$$\begin{aligned} &SM_{initial} = (O_1, L_1); O_1 = \{o_{1,i} \mid i = 1,...,n_1\}; \\ &L_1 = \{l_{1,j} \mid j = 1,...,m_1\};\ n_1 = \mid O_1 \mid;\ m_1 = \mid L_1 \mid \\ &SM_{resulting} = (O_2, L_2); O_1 = \{o_{2,k} \mid k = 1,...,n_2\} \\ &L_2 = \{l_{2,p} \mid p = 1,...,m_2\};\ n_2 = \mid O_2 \mid;\ m_2 = \mid L_2 \mid \end{aligned}, \quad (1)$$

where $O_1$ – set of $SM_{initial}$ objects, $O_2$ – set of $SM_{resulting}$ objects.

$L_1$ – set of $SM_{initial}$ links, $L_2$ – set of $SM_{resulting}$ links.

*Initial* and *resulting* are types of software models. For example if transformation performed from use case to collaboration diagram we write transform $SM_{use\_case}$ to $SM_{collaboration}$.

Graph representation of software model is not new approach. Contribution of proposed one consists in allowing forming parts of graph, namely sub-graphs that are important for particular transformation operation.

Thus, propose general representation of sub-graphs for some software model:

$$\begin{aligned} &SM_{sub} = (O_{sub}, L_{sub}); O_{sub} \subset O; L_{sub} \subset L \\ &O_{sub} = \{o_{sub,i} \mid i = 1,...,n_{sub}\};\ n_{sub} = \mid O_{sub} \mid, \\ &L_{sub} = \{l_{sub,j} \mid j = 1,...,m_{sub}\}; m_{sub} = \mid L_{sub} \mid \end{aligned} \quad (2)$$

where $O_1$ – set of $SM_{initial}$ objects, $O_2$ – set of $SM_{resulting}$ objects.

$L_1$ – set of $SM_{initial}$ links, $L_2$ - set of $SM_{resulting}$ links.

Where: $O_{sub}$ – a set of objects that are chosen from $SM$. Respectively $L_{sub}$ – a set of links between elements $o_{sub} \in O_{sub}$. Denote $SMI_{sub}$ –– sub-graphs of $SM_{initial}$, respectively $SMR_{sub}$ – sub-graphs of $SM_{resulting}$.

Thus:

$$\begin{aligned} &SMI_{sub} = (OI, LI); \\ &OI = \{oI_i \mid i = 1,...,nI\}; \\ &LI = \{lI_j \mid j = 1,...,mI\} \end{aligned} \quad (3)$$

$$SMR_{sub} = (OR, LR);$$
$$OR = \{oR_i \mid i = 1,...,nR\};$$
$$LR = \{lR_j \mid j = 1,...,mR\} \quad (4)$$

The purpose for designing $SMI_{sub}$ is the forming sub-graphs for further transformation. They are formed by rules of choosing proper sub-graphs from SM(O,L) for concrete transformation operation. Denote these rules as *initial selecting rules*.

$SMR_{sub}$ are sub-graphs obtained after transformation operation.

## 4.2 Formal representation of transformation rules

Denote transformation operation as $\rightarrow$. Thus:

$$SMI_{sub} \rightarrow SMR_{sub}, \quad (5)$$

$$(OI, LI) \rightarrow (OR, LR);$$
$$OI \subset O_1,\ OR \subset O_2,\ LI \subset L_1,\ LR \subset L_2 \quad (6)$$

Representation of transformation rule in details:

$$((oI_1, lI_1), \ldots, (oI_n, lI_n)) \rightarrow ((oR_1, lR_1), \ldots, (oR_m, lR_m));$$
$$oI_i \in OI, lI_i \in LI, oR_j \in OR, lR_j \in LR;$$
$$i = 1, \ldots, n;\quad j = 1, \ldots, m;\quad n = |OI|,\ m = |OR| \quad (7)$$

To represent transformation rules transformational grammar (Chomsky, 1957) is used.

Transformation rules are syntax of this grammar (Chebanyuk and Markov, 2016). They explain how to generate new sub-graphs from initial software model. Initial and resulting transformation information is represented as sub-graphs of software models. Second order logic is used for representation of transformation rules in high level (Chebanyuk and Markov, 2016) as it is written in (6). Also, such representation can be described in details (7) using first-order logic (Chebanyuk and Markov, 2016).

## 4.3 Formal representation of initial selecting rules

Initial selecting rules define how to choose $SMI_{sub}$ from $SM_{initial}$ for performing transformational operation. Denote initial selecting rule as $R(SM_{initial})$. Thus, operation of selecting $SMI_{sub}$ applying *R* on $SM_{initial}$ is written as follows:

$$R(SM_{initial}) = SMI_{sub}. \quad (8)$$

Usually initial selecting rules are composed as conditional statements, defining which parts of $SM_{initial}$ form $SMI_{sub}$.

Denote sub-graph for selecting pairs from $SM_{initial}$ as *S*. Thus:

$$S = (OS, LS)$$
$$OS = \{oS_i \mid i = 1,...,n_s\}, n_s = |O_s| \ LS = \{lS_j \mid j = 1,...,m_s\}, m_s = |L_s|' \quad (9)$$

*S* – is a mask which is applied to every pair $(o_1, l_1);\ o_1 \in O_1, l_1 \in L_1$. A mask may contain more than one graph pair.

Graph $SMI_{sub}$ is formed by the next: every pair $(o_1, l_1);\ o_1 \in O_1, l_1 \in L_1$ of $SM_{initial}$ is compared with $(oS, lS);\ oS \in O_1, lS \in L_1$. If considered pairs are the same, then $(o_1, l_1);\ o_1 \in O_1, l_1 \in L_1$ is added to $SMI_{sub}$.

Thus statement (5) can be written as follows:

$$select\ S\ from(SM_{initial}) = SMI_{sub}. \quad (10)$$

Analogously with transformation rules also first and second logics for representation of initial selecting rules are used.

5. General description of model to model transformation according to proposed formalization Describe steps of model to model transformation approaches, that is based on formalisms, proposed above in Chapter 4.

1. Transformation rules for transforming $SM_{initial}$ to $SM_{resulting}$ are composed. Denote a set of transformation rules as *TRANS*.

$$TRANS = \{\rightarrow_i) \mid i = 1,...,t\};\ t = \mid TRANS \mid. \quad (11)$$

Formal representation of transformation rules is proposed in (6) and (7).

2. A set of initial selecting rules is formed. Denote it as RULES. Thus:

$$RULES = \{R_i(SM_{initial}) \mid i = 1,...,q\};\quad q = \mid RULES \mid. \quad (12)$$

3. *RULES* are applied to $SM_{initial}$ to form set of $SMI_{sub}$ for further transformation.

Every $R_i(SM_{initial})$ forms graph $SMI_{sub,i}$. Denote a set of received $SMI_{sub}$ as $SMI_{selected}$. Thus:

$$\begin{aligned} &SMI_{selected} = \{SMI_{sub,i} \mid i = 1,...,q\}; \\ &\mid SMI_{selected} \mid = \mid RULES \mid \end{aligned} \quad (13)$$

$$\begin{aligned} &SMI_{selected,i} = \\ &= \{(oSMI_j, lSMI_j) \mid j = 1,...,n_{selectted,i}\} \end{aligned}. \quad (14)$$

where $q$ – is the number of sub-graphs in $SMI_{sub,i}$, $n_{selected,i}$ – is a number of pairs of sub-graph $SMI_{selected,i}$.

4. $SMI_{selected}$ is verified by deleting duplicated chains of pairs, that were obtained applying different initial selecting rules. One chain can be considered as path of sub-graph.

5. $SMI_{selected}$ is transformed to set of sub-graphs in $SM_{resulting}$ notation applying TRANS. Denote all obtained sub-graphs in $SM_{resulting}$ notation as SMR. Thus:

$$TRANS(SMI_{selected}) = SMR. \quad (15)$$

$$\begin{aligned} &SMI_{sub,i} \rightarrow SMR_{sub,i}\ ; \\ &i = 1,...,q;\ , q = \mid SMI \mid \end{aligned} \quad (16)$$

$$\begin{aligned} &SMR_{sub,i} = \{(oSMR_j, lSMR_j) \\ &\mid j = 1,...,n_{SMR_i}\} \end{aligned}. \quad (17)$$

where $n_{SMR_i}$ – is a number of pairs of sub-graph $SMR_{sub,i}$.

6. From SMR $SM_{resulting}$ is composed. Analytical approach for performing this operation will be proposed in further works.

## 5. Conclusions

Formal foundations and corresponded approach for model to model transformation is proposed in this paper. They are grounded on analytical representation of software models, transformation and initial selecting rules. It is proposed to represent software models as graphs. Expressions (1) and (2) let describing software model both in general

and detailed view. In comparison with other approaches, for example (Chebanyuk, 2014), such representation allows easily separating any part of software model for further analysis. Using the first and second order logics for expressions of initial selecting and transformation rules given in (5)-(14), also let to consider transformation operations involving necessary amount of details.

Representation of transformation rules, proposed in other papers for instance (Favre and Duarte, 2016) and (Varro and Pataricza, 2003), often use predicate calculus for expression of transformation operations. It causes to using complex expression for performing difficult transformations, including several conditions. Also operations of adopting such analytical representation to transformation environments are expensive.

From the other side, methods of performing transformation tasks involving concrete transformation languages (for instance ATLAS, QVT, QVT-R and others) and environments (MEDINIQVT, WEBDBF) are limited by possibilities of considered modeling language and transformation environment.

The approach, proposed in this paper, allows considering transformation process both on metalevel and model level (IBM, 2016). General transformation ideas and software models notations can be analyzed on metalevel.

Considering of sub-graphs and software models at level of elements lets analyzing transformations in details. Doing this existing rules can be refined and new transformation rules also can be designed.

## 5. Further work

Develop a full analytical approach of model to model transformation that is grounded on collaboration of mathematical tools, used for transformation operations. In order to accomplish this goal to do the following:

- propose an algorithm for designing of $SM_{resulting}$ that is grounded on ontology analysis. This algorithm should consider possibilities of human cognitive abilities for perception of software models (Chebanyuk and Markov, 2015);
- define operations that are used for analysis of software model before and after transformation, namely software model verification and semantic checking. Propose and analytical tool for performing these operation;
- verify collaboration of proposed approaches with model of problem domain “Model-Driven Architecture formal methods and approaches” proposed in paper (Chebanyuk and Markov, 2016).

## Referencies


1. Chebanyuk Elena and Krassimir Markov. 2015. Software model cognitive value. International Journal “Information Theories and Applications”, Vol. 22, Number 4, ITHEA 2015 http://www.foibg.com/ijita/vol22/ijita22-04-p04.pdf
2. Chebanyuk Elena and Krassimir Markov. 2016. Model of problem domain “Model-driven architecture formal methods and approaches” International Journal “Information Content and Processing”, Vol. 22, Number 4, ITHEA 2016. P. 202–222.
3. Chebanyuk ELena.2014. Method of behavioural software models synchronization. International journal Informational models and analysis. – 2014, № 2. P 147–163. http://www.foibg.com/ijima/vol03/ijima03-02-p05.pdf
4. Chomsky, Noam. 1957. Syntactic Structures. Mouton publishers, Eilenberg: Mac Lane The, Hague, 1945 - 1957. ISBN 90 279 3385 5. p.107. http://ewan.website/egg-course-1/readings/syntactic_structures.pdf
5. Czarnecki, Krzysztof and Simon Helsen. 2006. Feature-based survey of model transformation approaches. IBM Systems Journal Vol. 45 N 3: 2006. P. 621–645. ISSN :0018-8670, DOI: 10.1147/sj.453.0621. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=5386627
6. Favre Liliana and Daniel Duarte. 2016. Formal MOF Metamodeling and Tool Support. In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 99–110. DOI:10.5220/0005689200990110, http://www.scitepress.org/DigitalLibrary/ProceedingsDetails.aspx?ID=j1i7qrX33Ns=&t=1
7. Greiner Sandra, Thomas Buchmann, Bernhard Westfechtel. 2016. Bidirectional Transformations with QVT-R: A Case Study in Round-trip Engineering UML Class Models and Java Source Code. In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 15–27. DOI:10.5220/0005644700150027 http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=efZXth7Zbbg=&t=1
8. IBM, 2016. http://researcher.ibm.com/researcher/files/zurich-jku/mdse-07.pdf
9. Natschlager Christine, Verena Geist1, Christa Illibauer1 and Robert Hutter. 2016. Modelling Business Process Variants using Graph Transformation Rules In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 65–74. DOI:10.5220/0005686900870098. http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1
10. Rabbi Fazle, Yngve Lamo, Ingrid Chieh Yu, Lars Michael Kristensen. 2016. WebDPF: A Web-based Metamodelling and Model Transformation Environment. In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 87–98. DOI:10.5220/0005686900870098, http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1
11. Seifermann, Stephan and Henning Groenda. 2016. Survey on Textual Notations for the Unified Modeling Language In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires,

B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 28–39. DOI:10.5220/0005686900870098, http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1
12. Truyen Frank. 2006. The Fast Guide to Model Driven Architecture. The Basics of Model Driven Architecture (MDA). Cephas Consulting Corp, 2006. http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf
13. Wu Hao. 2016. Generating Metamodel Instances Satisfying Coverage Criteria via SMT Solving In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 40–51. DOI:10.5220/0005686900870098, http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1

## Література

1. Chebanyuk Elena and Krassimir Markov. 2015. Software model cognitive value. International Journal "Information Theories and Applications", Vol. 22, Number 4, ITHEA 2015 http://www.foibg.com/ijita/vol22/ijita22-04-p04.pdf
2. Chebanyuk Elena and Krassimir Markov. 2016. Model of problem domain "Model-driven architecture formal methods and approaches" International Journal "Information Content and Processing", Vol. 22, Number 4, ITHEA 2016. P. 202–222.
3. Chebanyuk ELena.2014. Method of behavioural software models synchronization. International journal Informational models and analysis. – 2014, № 2. P 147–163. http://www.foibg.com/ijima/vol03/ijima03-02-p05.pdf
4. Chomsky, Noam. 1957. Syntactic Structures. Mouton publishers, Eilenberg: Mac Lane The, Hague, 1945 - 1957. ISBN 90 279 3385 5. p.107. http://ewan.website/egg-course-1/readings/syntactic_structures.pdf
5. Czarnecki, Krzysztof and Simon Helsen. 2006. Feature-based survey of model transformation approaches. IBM Systems Journal Vol. 45 N 3: 2006. P. 621–645. ISSN :0018-8670, DOI: 10.1147/sj.453.0621. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=5386627
6. Favre Liliana and Daniel Duarte. 2016. Formal MOF Metamodeling and Tool Support. In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 99–110. DOI:10.5220/0005689200990110, http://www.scitepress.org/DigitalLibrary/ProceedingsDetails.aspx?ID=j1i7qrX33Ns=&t=1
7. Greiner Sandra, Thomas Buchmann, Bernhard Westfechtel. 2016. Bidirectional Transformations with QVT-R: A Case Study in Round-trip Engineering UML Class Models and Java Source Code. In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 15–27. DOI:10.5220/0005644700150027 http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=efZXth7Zbbg=&t=1
8. IBM, 2016. http://researcher.ibm.com/researcher/files/zurich-jku/mdse-07.pdf
9. Natschlager Christine, Verena Geist1, Christa Illibauer1 and Robert Hutter. 2016. Modelling Business Process Variants using Graph Transformation Rules In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 65–74. DOI:10.5220/0005686900870098. http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1
10. Rabbi Fazle, Yngve Lamo, Ingrid Chieh Yu, Lars Michael Kristensen. 2016. WebDPF: A Web-based Metamodelling and Model Transformation Environment. In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 87–98. DOI:10.5220/0005686900870098, http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1
11. Seifermann, Stephan and Henning Groenda. 2016. Survey on Textual Notations for the Unified Modeling Language In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 28–39. DOI:10.5220/0005686900870098, http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1
12. Truyen Frank. 2006. The Fast Guide to Model Driven Architecture. The Basics of Model Driven Architecture (MDA). Cephas Consulting Corp, 2006. http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf
13. Wu Hao. 2016. Generating Metamodel Instances Satisfying Coverage Criteria via SMT Solving In: MODELSWARD 2016, Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. Edited by S. Hammoudi, L.F. Pires, B. Selic and P. Desfray. SCITEPRESS – Science and Technology Publications, Lda. Portugal, 2016. ISBN: 978-989-758-168-7. P. 40–51. DOI:10.5220/0005686900870098, http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=lzjjeczBZuA=&t=1

***About author:***

*Chebanyuk Elena,*
PhD, accosiate professor of software engineering departement.
Number of publications approaximatelly 65:
in foreign journals – approximately 25, in Ukrainian 35.
ttps://orcid.org/0000-0002-9873-6010

***Affiliation:***

National Aviation University,
03680, Ukraine, Kyiv, Kosmonavta Komarova ave. 1.
Phone: (044) 406 7641,
E-mail: chebanyuk.elena@gmail.com,
chebanyuk.elena@ithea.org