Method on Specifying Consistency Rules among Different Aspect Models, expressed in UML

R. Dubauskaite[1], O. Vasilecas[1]
[1]*Department of Information Systems, Vilnius Gediminas Technical University, Sauletekio al. 11, LT-10223 Vilnius, Lithuania*
ruta.dubauskaite@vgtu.lt

***Abstract*—Unified Modelling Language allows modelling different aspects of information system through the various diagrams it supports. Expression of an information system through class, state, and other models is related to the problem of checking consistency among different aspect Unified Modelling Language models. Consistency means that two or more overlapping elements of different aspect models match each other. Approaches of checking Unified Modelling Language models are based on rules. Most of consistency rules are ambiguous, do not conform OMG Unified Modelling Language metamodel and sometimes are meaningless. In order to improve consistency of different aspect models, the approach of checking consistency is proposed, paying a special attention on requirements of consistency rules. Example of consistency rule and experiments are presented.**

***Index Terms*—Consistency, modelling, rules, UML.**

I. INTRODUCTION

UML (*Unified Modelling Language*) is a general-purpose modelling language that can be used with all major object and component methods. It was chosen for detail analysis because:

1) UML is likely to be the most popular modelling language;
2) It is considered as the standard for the object-oriented modelling [1];
3) There are many modelling tools supporting UML [2].

UML allows modelling different aspects of information system through the various diagrams it supports. *Aspect* is a projection into a model, which is seen from a given perspective and omits entities that are not relevant to this perspective [3]. *Aspect model* means elements of IS model that can be visualised by several the same aspect diagrams.

Sometimes the models of class, state, and other aspects are not interrelated and even more, contradictory information can be provided in them. For example, it is possible that elements created in class model, are not used when modelling states of class. Expression of an IS through various models is related to the problem of consistency ensuring of different aspects models. *Consistency* means that the structures, features and elements that appear in one model are compatible and in alignment with the content of other models [3]. Sometimes consistency concept is misused for expressing well-formedness of IS models. *Well-formedness* is concerned with a correct use of notations to describe one aspect model, consistency among diagrams usually are not classified to well-formedness [4].

Model consistency issue is particularly important within the scope of model-driven architecture (MDA). Unambiguous models are necessary for the successful accomplishment of the tasks of model transformation and finally for code generation. The goal of our research is to improve methods of checking consistency of different aspects UML models. *UML model* is an abstraction of the physical system, created with a certain purpose and expressed in UML.

The rest of this paper is organized as follows: section "Related works" presents approaches of checking UML models and detailed results of analysis of consistency rules. The proposed method of checking consistency of UML models, including requirements of consistency rules are provided in section "Proposal". Section "Case Study" illustrates the evaluation and usage of proposed method. Finally, conclusions are provided.

II. RELATED WORKS

Different approaches to check UML models and their consistency rules are researched and presented in the following subsections.

A. Approaches of Checking UML Models

Initial researchers on checking of UML models appeared in 2002. Liu *et al.* [5] were the first ones to suggest the paradigm of checking of UML models based on reasoning mechanism of a formal language. Its idea is translating UML models and their consistency rules to any formal language. Then inconsistencies are detected using reasoning mechanism (e.g., forward chaining algorithm or/and engine that implement it). Rash and Wehreim [6] suggest using Process Algebra, Object-Z, Mokhati *et al.* [1] propose rewriting logic, Miloudi *et al.* [7] prefer Z language for formal models. The main advantage of these approaches is ease of check consistency – availability of inconsistency detection algorithms of formal systems and inference

Manuscript received July 25, 2012; accepted December 4, 2012.

engines of a design tool that implement these algorithms. One of the disadvantages of these approaches is that formal languages despite they are more precise, they are not popular in practice for initial models.

Kotulski [8], Wang *et al.* [9] refines the approaches by suggesting usage of formal languages, which have visual expression, for example, Node-label-controlled (NLC) graphs, OWL-DL (Ontology Web Language). The main disadvantage of these approaches is that their models, rules are not defined for UML metamodel provided by OMG (Object Management Group). They are mapped to descriptions of UML models, defined by Kotulski [8] and other researchers. Morover, translation of UML models to formal models requires additional resources. More information about rule-based, inference mechanism and other knowledge-based systems is provided in [10].

Another group of approaches of UML models checking, which is evolved almost in parallel with approaches based on UML models translating to formal models, are constraint-driven approaches. The main idea of them is suggestion to the check semi-formal UML models according to defined constraints. Semi-formal model is created using language, which syntax is defined formally, but most of semantics is described using natural language [11]. Studies of this group differ in checked property (consistency or correctness/well-formedness) and language for expressing rules. Chiorean *et al.* [12], Pakalnickiene *et al.* [13] propose checking correctness of UML models according to OCL rules that constrain one aspect model. Chen and Motet [14] propose controlling grammar C-Control for expressing correctness rules. Other analysed works propose approaches of checking consistency of UML models. Sapna and Mohanty [15] provide several examples of OCL consistency rules and their translation to SQL, Chanda *et al.* [16] suggest several consistency rules expressed in context free grammar. The main disadvantage of works is that algorithm of checking consistency of IS models is not presented explicitly. More details about the research of related approaches are provided in our previous paper [17].

Both groups of methods uses rules, therefore consistency rules are researched in detail in the following section.

B. Consistency Rules

A Full, non redundant, clear and meaninglful set of consistency rules is necessary for method of UML models consistency checking and especially for automation of consistency checking process.

Therefore 50 consistency rules were elicited from 10 related researches and examined in order to (a) find out whether the provided rules may be understood unambiguously; (b) determine whether they conform to specification (metamodel) of UML provided by OMG [13]; (c) find out whether they are meaningful. It means if they really show conflict of consistency. Further these issues are presented in detail.

Rules expressed in natural language can be interpreted ambiguously, for example:

Rule 1: *Swimlines in Activity diagram (represented as className in activity state) must be present as a unique class in class diagram* [16]. What is an activity state? Swimline is a partition? The formal expression for rule 1 is provided below:

From the grammar proposed for the activity diagram
activity_state →
act_ID event_ID activity_node act_desc className
pre_element post_element
From the grammar proposed for the class diagram
classes→ cname attribute* method_class*
$\forall$ className $\in$ className_act and
$\forall$cname$\in$ className_class then
className_act $\subseteq$ className_class

Chanda *et al.* [16] do not provide mapping of metaelements of OMG UML metamodel. Therefore it is unclear exactly how to map elements from their description of UML models to OMG UML metamodel.

The analysis of consistency rules also reveals that there are rules contradicting model requirements expressed in OMG UML specification. Example of such rule is:

Rule 2: *Each object and message in a sequence diagram must have a corresponding class and method in the class diagram* [15]. According to OMG UML specification [11] only calling messages have to be defined in class (in case *messageSort* is either *synchCall* or *asynchCall* then message have to refer to an Operation).

Sometimes rules are meaningless and necessity of them is doubt, for example:

Rule 3: *A specification consisting of an Object-Z class and an associated state machine has the property of method executability if in the corresponding process in the semantic model every method is executed at least once* [6]. What is the origin of the rule? Is it an IS development methodology or OMG UML metamodel? E.g. method *getClientData()* is it really have to be used in State model? May be the rule is valid in some conditions but they are not provided.

Analysis of consistency rules shows that most rules are expressed in natural and formal language. The main reasons of ambiguity are: (a) incompleteness, different structure of rules e. g. associated elements or models are not defined explicitly (b) synonyms for the same elements. Formal rules usually use their own description of UML models. Therefore it is unclear what elements of OMG UML metamodel they conform. Besides some consistency rules do not conform to OMG UML metamodel and their practical necessity is doubt.

In summary existing works shows that issue of UML models consistency is important, but there is a need to improve the approaches of consistency checking.

III. PROPOSED APPROACH ON SPECIFYING OF CONSISTENCY RULES AMONG DIFFERENT ASPECTS UML MODELS

In general a method of consistency ensuring of UML models, the processes of UML models checking and removing inconsistencies is presented in our paper [18].

This section presents a proposed method of checking consistency of UML models, especially concentrating on requirements of consistency rules. The necessity of these requirements origins from results of related work analysis. It reveals that most rules are ambiguous, do not conform OMG UML metamodel and they are meaningless sometimes. Therefore it has negative impact on reusing rules and developing more comprehensive set of rules.

The proposal is presented in Fig. 1, the essence of proposal is:

1. Check consistency of semi-formal IS models using consistency rules;
2. Define consistency rules among different aspects IS models according to these requirements:
   2.1. Define consistency rules at three abstraction levels: metamodel independent, metamodel specific and formal/program code;
   2.2. Verify consistency rules according to a metamodel of modelling language;
   2.3. Motivate the necessity of rules defining its origin.

Assigning enforcement level according to their scope of application.

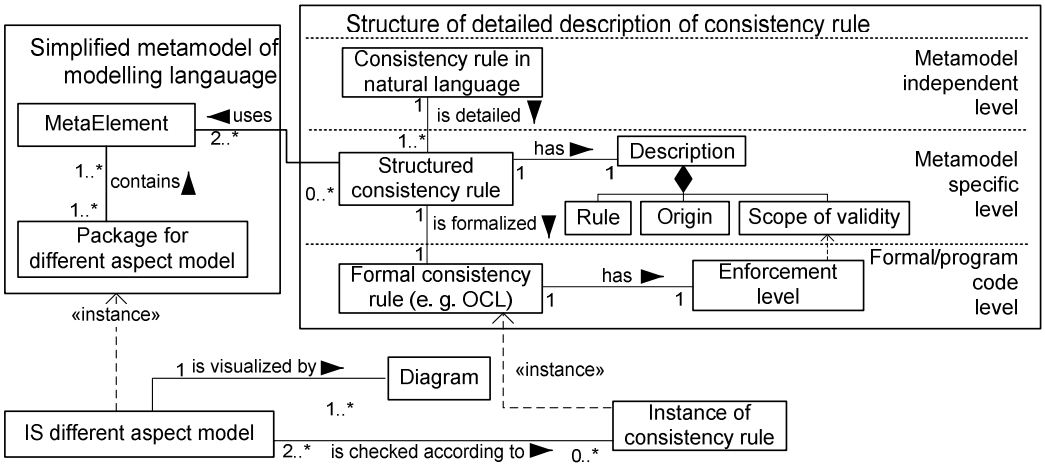

Fig. 1. Structure of detailed description of consistency rule its associations with metamodel of modelling language.

The idea of modelling is based on three levels applied from OMG MDA standard. A platform is changed to a metamodel in adapted MDA transformation schema between different abstraction levels. According to the adapted MDA it is required to model consistency rules in series. Every consistency rule has to be expressed at three levels: metamodel independent, metamodel specific and formal/program code.

At the metamodel independent level a rule is expressed in natural language. It is necessary for general understanding of the rule, even for developer, who has not special knowledge of modelling languages. Rules expressed in a natural language can be interpreted variously. In order to reduce ambiguity it is required to elaborate a consistency rule, expressed in a natural language.

At the metamodel specific level a structured consistency rule refers to OMG UML metamodel metaelements. It is important to emphasise that it is required to associate metaelements from UML specification developed by OMG. Because the reviewed related researches show that UML models descriptions provided by various authors use different concepts for the same objects. At this level it is also required to define an aspect model, which contains an instance of the associated metaelement. To simplify a metamodel specific rule it is recommended to divide it into two parts (Table I).

The third level of a consistency rule is formal or program code level. Expressing consistency rule in formal or programming language it is not mandatory, because formal rules or program code are seldom provided in specification of software system. On the other hand, formal rules can be interpreted unambiguously and rules of program level can reduce time of IS development.

The analysis of existing consistency rules shows that constraint, which is valid always, is too strict in practice. Moreover, consistency rules are defined at metamodel level and, it means that they are sufficiently general. General rules usually do not include specific cases. Hence there are situations when the detected violations of consistency rules do not mean consistency conflict. Therefore, it is proposed to define an enforcement level of a consistency rule. It indicates the necessity of reaction (if it is necessary to modify models) to the detected consistency conflict. If the detected violations of rules show consistency conflicts depend on specific situations, then IS engineer or a knowledge expert can decide whether the situation is inconsistent. A consistency rule has to be assigned with one of three enforcement levels that are presented in Table II.

TABLE I. SPECIFICATION OF CONSISTENCY RULE 4.

Rule ID		R4	
Rule at a metamodel independent level		The class which states are modeled has to be known in Protocol states model	
Rule at meta-model specific level	Rule	Context of protocol states has to be defined by the class.	
	Associated meta-elements	Context of Protocol State Machine	Classifier of Class model
Enforcement level		High	
Descrip-tion		A protocol state machine presents possible and permitted transitions on the instances of its context classifier, together with the operations that carry the transitions. In this manner context – class, which operations can be called, and their execution that determines changes of states of the object have to be defined. The origin of this constraint is the analysis of UML superstructure specification provided by OMG [19].	

Consistency rule R4 can be expressed as OCL invariant:

```
context ProtocolStateMachine inv protocol
States_without_context:self.oclAsType(State
Machine).region.context->notEmpty()
```

In order to prove the necessity of a rule, to reduce number of meaningless rules it is required to provide a description of consistency rule. The description has to include an

explanation of the rule, a definition of origin and a scope of validity (explanation why one or another enforcement level is chosen). The origin of the rule can be OMG UML specification, IS development methods, e.g. RUP, ICONIX, Newton, practical work analysis, etc. Example of consistency rule specification is provided in Table II.

TABLE II. ENFORCEMENT LEVEL OF CONSISTENCY RULES.

Enforcement level	Type of message about violation of the rule
Low	Information
Medium	Warning
High	Error

IV. A CASE STUDY

The first experiment is aimed at the evaluation of the proposed requirements for consistency rules. We demonstrate how various consistency rules from different papers ([4], [15], [16]) and our rules (specified using the proposed requirements) are understood by analysts, designers, programmers, and quality engineers.

The researches of Egyed [4], Sapna, Mohanty [15] and Chanda *et al.* [16] are selected for the experiment because their approaches are the most similar to our proposal compared to other analyzed related researches. In this study the questionnaire is filled by 14 specialists that have theoretical or/and practical knowledge about UML. The questions were about knowing of semantic, associated metaelements, conformance to the metamodel, knowing the origin of rule. Analysis of collected data was performed using *paired t-test* [21] method (Table III).

The second experiment is the extension of UML tool with the UML models consistency checking module prototype *ConsistencyConstraint4UML*. The purpose of the investigation is to test consistency of specific IS models using a developed software prototype. The usage of the implemented software prototype is provided in Fig. 2.

TABLE III. APPLICATION OF PAIRED T-TEST FOR COMPARING PROPOSED AND PREVIOUS [5, 7, 18] METHODS.

Paired t-test method [24]		Application of Paired t-test
Input	Paired samples: (x_1, y_1), (x_2, y_2) … and (x_n, y_n), in this case n-number of participants that provide answers to questionnaire; x-count of positive ('yes) answers about proposed method. y-count of positive ('yes) answers about previous methods	The 14 paired samples obtained after calculating total number of answers 'yes' (to questions about unambiguity and reliability of consistency rules from our and previous methods) provided by 14 participants (13, 4), (10, 7), (14, 10), (12, 9), (10, 8), (8, 9), (8, 10), (11, 9), (9, 7), (14, 8), (12, 7), (8, 6), (11, 9) and (12, 5).
H_0 (Null hypotheses) H_1 (Alternative hypotheses)	H_0: The expected mean of differences ($d_i = x_i - y_i$) is 0 ($\mu_d = 0$); H_1: The expected mean of differences is more than 0 ($\mu_d > 0$)	H_0: The proposed method has the same quality (unambiguity and reliability) as previous methods. H_1: The proposed method has better quality (more answers 'yes' about unambiguity and reliability) compared with previous methods.
Calculations	Calculate $t_0 = \frac{\bar{d}}{S_d\sqrt{n}}$, where $S_d = \sqrt{\frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n-1}}$	Based on the data it can be seen that n = 14 The mean of differences is $\mu_d = 3{,}143$ (Formulas are provided in first column of the table). It can be found that $S_d = 3{,}931$ and $t_0 = 4{,}011$.
Criterion	If $t_0 > t_{\alpha,n-1}$ reject H_0 (H_0: $\mu_0 > 0$) and acce pt H_1 (H_1: $\mu_0 > 0$), where $t_{\alpha,f}$ is the upper α percentage point of the t distribution with f degrees of freedom, which is equal to n – 1. The distribution is tabulated in Table A1 from [21], which presents critical values two-tailed t-test (5 %).	The number of degrees of freedom is f = n -1 = 14 – 1 = 13. In Table A1 from [21], $t_{0.5,\,13} = 2.160$. $t_0 = 4{,}011 > 2.160 = t_{0.5,\,13}$ therefore H_0 is rejected and H_1 is accepted with 95% (100%-5%) confidence level.

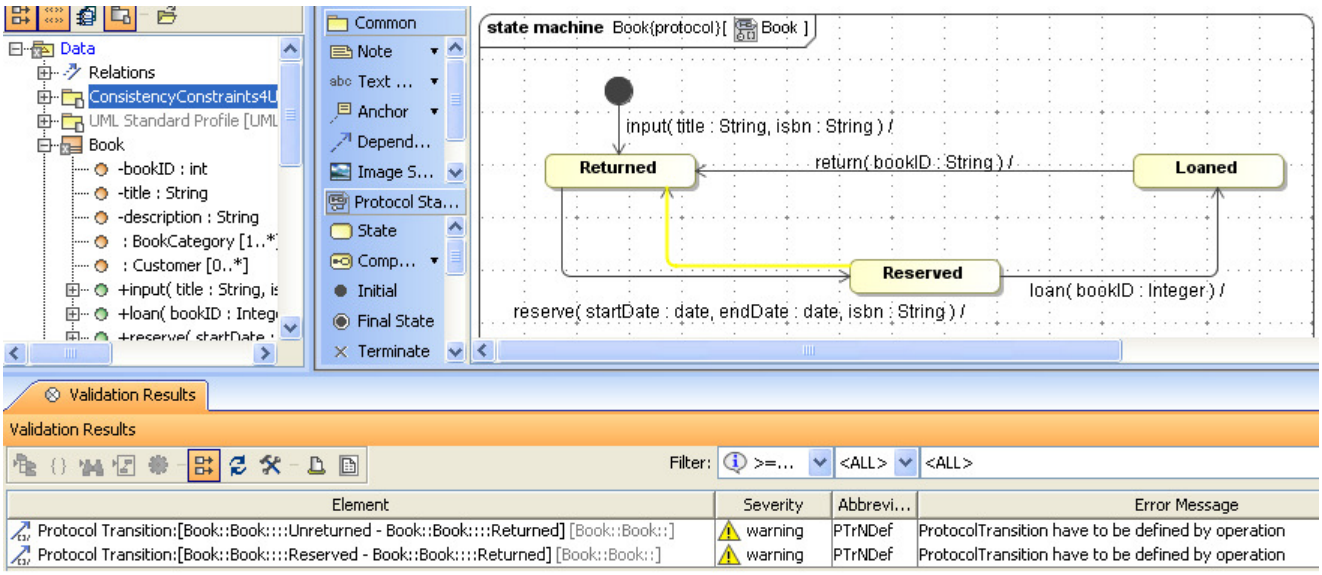

Fig. 2. Checking of IS models using developed module *ConsistencyConstraints4UML*.

UML models are validated according to every consistency rule. The detected consistency conflicts are shown at the bottom of the right column in validation results section of Fig. 2. Left column of Fig. 2 provides part of UML models,

developed using MagicDraw UML 17.0 tool. In the right column of Fig. 2 a states model is visualized using protocol states diagram. The diagram represents possible states of a class Book. Book is a part of library IS.

The last step of ensuring consistency of UML models is modifying of IS models according to detected consistency conflicts. If method cancelReservation() is associated with transition from Book state Reserved to state Returned then consistency of library system models would be improved.

V. CONCLUSIONS

The analysis of methods for IS models consistency checking and their consistency rules shows the relevance of that the solved problem. Various methods using constraints for IS models or using algorithm/engine of a formal language for detecting inconsistencies are proposed. However there is no any comprehensive method how to check consistency and to create a more understandable and more reliable set of consistency rules.

A method of IS different aspects models not related with a specific modelling language is proposed. The feasibility of the proposed method is illustrated creating a set of consistency rules for UML models according to the proposed requirements. The rules are defined at the metamodel level; therefore, they can be implemented in any design tool that supports UML 2.2 metamodel.

The evaluation of the results obtained during the experiment showed that the proposed requirements for consistency rules improve the quality of a set of the rules (less ambiguity, more reliability) by approximately 41% in comparison with other similar methods. The consistency rules that are specified according to the proposed requirements are also more understandable by IS engineers compared with the rules provided by other researches.

The experiment performed demonstrated that the usage of the developed module ConsistencyConstraints4UML allows detecting consistency conflicts among different aspects models.

REFERENCES

[1] F Mokhati, P. Gagnon, M. Badri, "Verifying UML Diagrams With Model Checking: A Rewriting Logic Based Approach", in *Proc. of the Seventh International Conference on Quality Software (QSIC)*, Portland, 2007, pp. 356–362. [Online]. Available: http://dx.doi.org/10.1109/QSIC.2007.4385520

[2] R. Dubauskaite, O. Vasilecas, "Tool Support for Checking Consistency of UML Model", in *Proc. of the 20th International Conference on Information Systems Development (ISD 2011)*, Edinburgh, Scotland, 2013.

[3] N. Rozanski, E. Woods, *Software System Architecture*. London, 2005, p. 546.

[4] A. Egyed, "F*ixing inconsistencies in UML design models",* in *Proc. of the 29th International Conference on Software Engineering (ICSE 2007),* New York, 2007, pp. 292–301. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.38

[5] W. Liu, S. M. Easterbrook, J. Mylopoulos, "*Rule-based detection of inconsistency in uml models"*, in *Proc. of the 5th International Conference on the Unified Modelling Language (UML 02)*, London, 2002, pp. 106–123.

[6] H. Rasch; H. Wehrheim, "Checking Consistency in UML Diagrams: Classes and State Machines. Formal Methods for Open Object-Based Distributed Systems", *LNCS* 2884, pp. 229–243, 2003.

[7] K. E. Miloudi, Y. E. Amrani, A. Ettouhami, "An Automated Translation of UML Class Diagrams into a Formal Specification to Detect UML Inconsistencies", in *Proc. of the Sixth International Conference on Software Engineering Advances (ICSEA 2011)*, Barcelona, 2011, pp. 432–438.

[8] F. L. Kotulski, "Assurance of system consistency during independent creation of UML diagrams", in *Proc. of the International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX 2007)*, Szklarska Poreba, 2007, pp. 51–58.

[9] Z. Wang, et al., "Ontology Based Semantics Checking for UML Activity Model", *Information Technology Journal,* vol. 11, no. 3, pp. 301–306, 2012. [Online]. Available: http://dx.doi.org/10.3923/itj.2012.301.306

[10] R. Butleris, A. Lopata, M. Ambraziunas, S. Gudas, "The Main Principles of Knowledge-Based Information Systems Engineering", *Elektronika ir Elektrotechnika (Electronics and Electrical Engineering),* no. 4, pp. 99–102, 2012.

[11] OMG. 2010a. *Unified Modelling Language (OMG UML), Superstructure, v2.3*, OMG Document: formal (2010-05-05). [Online]. Available: http://www.omg.org/docs/formal/10-05-05.pdf

[12] D. Chiorean, et al., "Ensuring UML models consistency using the OCL Environment", in *Proc. of the ENTCS*, 2004, pp. 99–100.

[13] E. Pakalnickiene, L. Nemuraite, "Checking of conceptual models with integrity constraints", *Information technology and control*, vol. 36, no. 3, pp. 285–294. 2007.

[14] Z. Chen, G. Motet, "A Language-Theoretic View on Guidelines and Consistency Rules of UML", *LNCS* 5562, pp. 66–81, 2009.

[15] P. G. Sapna, H. Mohanty, "Ensuring consistency in relational repository of UML models", in *Proc. of the 10th International Conference on Information Technology (ICIT 2007)*, Rourkela, 2007, 217–222.

[16] J. Chanda, et al., "Traceability of Requirements and Consistency Verification of UML UseCase, Activity and Class diagram: A Formal Approach", in *Proc. of International Conference on Methods and Models in Computer Science 2009 (ICM2CS 09)*, New Delhi, 2009, pp. 1–4.

[17] O. Vasilecas, R. Dubauskaite, R. Rupnik, "Consistency Checking of UML Business Model", *Technological and Economic Development of Economy,* vol. 17, no. 1, pp. 133–150, 2011. [Online]. Available: http://dx.doi.org/10.3846/13928619.2011.554029

[18] R. Dubauskaite, O. Vasilecas, "Ensuring Models Consistency in the OMT, Booch, and OOSE Object-Oriented Methods", *Information Sciences,* no. 50, pp. 160–167, 2009.

[19] A. G. Kleppe, J. B. Warmer, W. Bast, *MDA Explained– The Model Driven Architecture: Practice and Promise*. Addison-Weley, 2004, p. 171.

[20] D. Vavpotic, M. Bajec, "An approach for concurrent evaluation of technical and social aspects of software development methodologies", *Information and software technology,* vol. 51, no. 2, pp. 528–545, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2008.06.001

[21] C. Wohlin, et al., *Experimentation in Software Engineering: An Introduction*. United Kingdom, 2000, p. 204.