Baltic J. Modern Computing, Vol. 1 (2013), No. 1-2, 63-76

Ensuring Consistency in Different IS Models – UML Case Study

Diana KALIBATIENE, Olegas VASILECAS, Ruta DUBAUSKAITE

Vilnius Gediminas Technical University, Sauletekio al. 11, LT-10223 Vilnius, Lithuania

diana.kalibatiene@vgtu.lt, olegas.vasilecas@vgtu.lt, ruta.dubauskaite@vgtu.lt

Abstract: Information systems (IS) design is often modelled as a collection of diagrams (e.g. UML diagrams), to depict different aspects of a system such as behaviour, structure, functionality, etc. Refinement of models and the evolving nature of software may lead to inconsistencies in these diagrams. Inconsistent IS model specification might be transformed to an incoherent and conflicting system. Current tools lack of support for maintaining consistency between diagrams. This paper shows that the currently existent methods are insufficient for consistency checking in IS models. Therefore, authors of this paper propose a rule based method for consistency checking in IS models, which is implemented to check consistency in UML diagrams. The proposed method was evaluated using comparative analysis and questionnaires.

Keywords: consistency, UML, model, rule, constraint.

1. Introduction

Information systems (IS) are often modelled as collection of different diagrams, which depict system's processes, states, structure, etc., e.g. certain aspects of a system. *Aspect* model is an abstraction of IS, developed with a certain goal. Elements of one aspect model can be visualised by one or several diagrams. For example, the structure of an IS can be presented by several entity-relationship diagrams or UML class diagrams. Every different aspect model can be analysed separately; however, it is a view of the same system. Therefore, it is natural, that some elements of models overlap and express the same things, only from different aspects. For example, in a UML sequence diagram when an object sends a message to another object, it implies that in a UML class diagram the two classes have a relationship that must be shown on this diagram. Consequently, there is a possibility to create different IS aspect models with inconsistencies.

Consistency means that the structures, features and elements that appear in one model are compatible and in alignment with the content of other models (Rozanski and Woods, 2005). Unambiguous and consistent models are necessary for the successful accomplishment of the tasks of model transformation and finally for IS program code generation. Therefore, the issue of models consistency is particularly important within the scope of model-driven architecture (MDA).

The problem of consistency checking of IS different aspects models arises when several analysts and/or designers model the same system, since they can use different terms for the same object of a domain. If the IS is large and complex, the risk of consistency conflicts in the models is bigger. Therefore, the issue of ensuring consistency is even more relevant. Moreover, even one IS engineer often creates models having consistency conflicts, because of (a) iterative process of IS development, (b) lack of knowledge and practice, etc.

The problem of consistency checking of IS different aspects models in a design phase is important and it has been widely discussed in the publications of recent years. However, none of the analysed methods has been accepted as a standard yet. Ambiguous, not conforming to meta-model of modelling language, sometimes meaningless consistency rules reduce the reusability and practical applicability of the proposed methods. Therefore, it is relevant to propose a method for consistency checking of IS different aspects models using rules and paying special attention to the requirements for consistency rules.

This paper is structured as follows. Section 2 presents related works on consistency checking of IS models. Section 3 introduces the suggested method of ensuring consistency in IS models. Section 4 presents the experiment performed to evaluate the suggested method. Section 5 concludes the paper.

2. Related works

According to Simmonds and Bastarrica (2005), *consistency* is a state in which two or more elements, which overlap in different models of the same system, have a satisfactory joint description. The task is to ensure consistency of a model, consistency of diagrams depends on accuracy of a model. A model can be visualised by several diagrams.

Consistency can be classified to vertical (inter-model), horizontal (intra-model), evolution, semantic or syntactic. *Vertical* or *inter-models* consistency is checked at different levels of abstraction between different aspects models (Lucas et al., 2009; Usman et al., 2008). *Horizontal* or *intra-models* consistency can be defined as a matching ratio between models at the same level of abstraction (ISO/IEC 1997). *Evolution* consistency is validated between different versions of the same aspect model (Straeten et al., 2003). All mentioned types of consistency can express syntactic or semantic conformance of different aspects models. *Syntactic* consistency expresses matching of models to the specifications of a meta-model. *Semantic* consistency requires that a model would be compatible to semantic meanings defined by a meta-model (Lucas et al., 2009; Usman et al., 2008). In this paper, we concentrate on improving models syntactic and semantic horizontal consistency of IS different aspects models expressed by a semi-formal language.

Semi-formal models are widely used; therefore, they are of interest for us. For the detailed study we choose semi-formal *Unified Modelling Language* (UML) (Matta et al,. 2004; Cavarra et al., 2004; Cheng, 2001). Moreover, UML allows us to model different aspects of IS. It is likely to be the most popular modelling language (Silingas and Butleris, 2009). There are many modelling tools supporting UML (Shen et al., 2002).

UML was developed by Object Management Group[1] (OMG), which also introduced MDA (Lucas et al., 2009). Consistency of UML model is especially important in MDA, for automatic transformation of initial model to specific model and finally code generation tasks (Rozanski and Woods, 2005; Berkenkötter, 2008).

Our research gives more attention to *consistency of UML models*. Therefore, the related works that analyse conformance of different aspects models (expressed by consistency rules) are selected for a more detailed analysis. As presented in (Ha and Kang, 2008), there are several trends for consistency checking in UML diagrams: meta-model based methods (Paige et al., 2007), graph-based methods (Taentzer, 2004; Shuzhen and Shatz, 2006), scenario-based methods, constraint-based methods (are the most popular) and knowledge-based methods (like (Wang et al., 2012; Wang et al., 2006)). We are concentrated on meta-model and constraint based methods. The results of analysis are presented as follows.

Table 1. Results of consistency rules analysis

Consistency rules		**Associated different aspects models**										**Expressed in a natural language**	**Associated OMG UML metamodel metaelements are specified**	**Expressed in a formal language**
		Class–State	**Class–Sequence**	**Class–Activity**	**Class–Use Case**	**Sequence – State**	**Sequence – Activity**	**Sequence – Use case**	**Activity-State**	**Activity-Use case**	**Total per source:**			
Author, Year [reference]	Egyed, 2007		1			1					2	+		
	Sapna and Mohanty, 2007	2	2	1	1	1	2	1	1	1	12	+	1 formal rule	1
	Paige et.al., 2007		1	1			1		1		4	+		+
	Chanda et al., 2009			3						1	4	+		+
	Liu et al., 2002							1			1	+		+
	Rasch and Wehrheim, 2003	5									5	+		+
	Straeten et al., 2003; Straeten, 2004; Simmonds and Bastarrica, 2005a	2	4			1					7	+		+
	Shinkawa, 2006						3			2	5	+		
	Kotulski, 2007				1						1	+		
	Borba and Silva, 2010	1	1	1	1	1	2	1	1	1	10	+		+
	Ibrahim et al., 2011									3	3	+		+
Total per diagram:		10	8	5	3	4	7	3	2	8	50			
Different rules:		7	6	3	2	3	5	2	2	3	32			

[1] http://www.omg.org/

For the detailed study 50 consistency rules were elicited from 11 related researches (see Table 1) and examined in order to:

1. evaluate consistency rules, excluding redundant rules;
2. find out whether the provided rules may be understood unambiguously;
3. determine whether they conform to specification of a model – OMG UML metamodel;
4. find out whether they are meaningful, i.e. whether they really show a conflict of consistency.

The count of consistency rules associating UML models of specific aspects provided

in the specific research is presented in Table 1, Part "Associated different aspects models". The line "Different rules" demonstrates how many various rules are presented in different approaches among the same two aspects models.

The three last columns in Table 1 indicate whether the rule expressed in a natural language or/and a formal language and whether the associated metaelements from OMG UML metamodel (OMG, 2009; OMG, 2009a) are defined. A plus sign (+) indicates that all the rules provided in the paper have specific expression; otherwise, a number shows the count of rules expressed in a natural language, containing metaelements from OMG UML specification or having a formal expression. The analysis shows that all the analysed rules are expressed in a natural language, and most rules have a formal expression. E.g., rules having formal expressions understood unambiguously. Moreover, according to the analysis these rules really show a conflict of consistency.

Table 2 gives a summary of the analysed NoMagic MagicDraw, Sybase PowerDesigner, Gentleware Poseidon for UML, IBM Rational System Architect and Microsoft Visio tools.

Table 2. Comparison of the design tools Magic Draw 17.0, Power Designer 16.1, Poseidon for UML 8.0, Rational Software Architect 11.3.1 and Visio 2010

Compared design tools / **Comparison criteria**	**Magic Draw 17.0**	**Power Designer 16.1**	**Poseidon for UML 8.0**	**Rational Software Architect 11.3.1**	**Visio 2010**
1. Model in conformity with metamodel	+	+	partially	+	partially
2. Correctness checking	+	+	-	+	-
3. Consistency checking	partially	-	-	partially	-
4. Language for expressing/ implementation rules	OCL, Java	Visual Basic	Java	Visual Basic	Visual Basic (for Macros), .NET (for plugin)
5. Technique of tools extension with new rules	module, plug-in	plug-in	plug-in	macros	macros, plug-in

The criteria for comparison are:

1. *Model in conformity with a metamodel*. Possible values are "+" (almost conform) and "partially" conform. If a model is in conformity with UML metamodel, it is checked according to one rule: *name of a class has to be unique*. If the tool does not allow creating a class with the same name in the model, then it is assumed that the model almost conforms to the metamodel. It is said "almost" because it is not checked whether all constraints defined in a metamodel are implemented in the tool. If the tool allows creating two classes with the same name, it is assumed that the model partially conforms to the metamodel. It is said "partially" because a tool does not implement all the constraints defined in the specification of UML; however, it provides metaelements defined in a metamodel.
2. *Correctness checking* – constraints are defined at the metamodel level for one aspect model, e.g. for a class diagram.
3. *Consistency checking* – constraints are among 2 and more different aspect models at the metamodel level. Value "partially" means that there are only several rules that constrain two different aspect models, e.g. Class and Sequence. Meanwhile, other aspects models and their relationships (e.g. a class and states) are not included.
4. *Language for expressing/ implementation rules.*
5. *Technique of tools extension with new rules*. Examples are developing a module or plug-in, or macros or using other techniques for the extension of the tool with new rules.

Despite the existence of many tools, it is not easy to develop models that conform to UML metamodel. Moreover, not all available tools have a facility to check consistency of IS models, and almost all defined constraints are for one aspect models.

3. The Rule-Based Method for Consistency Checking in IS Models: UML Case Study

According to the results obtained during the analysis of the related works, 12 rules for consistency checking is defined and *the rule-based method for consistency checking in IS models* is proposed. This method describes how to apply the defined rules. Although, Table 1 presents 32 rules, some of them overlap. Therefore, our 12 rules are inference of Table 1. The defined rules are presented in (Dubauskaite and Vasilecas, 2013).

An example of Rule 1 is as follows: *The class which states are modelled has to be known in the Protocol states model*. The formal expression using OCL invariant for rule 1 is provided below:

```
context ProtocolStateMachine inv
protocolStates_without_context:
    self.oclAsType(StateMachine).region.context-
    >notEmpty()
```

The motivation of necessity of Rule 1 is as follows. A protocol state machine presents possible and permitted transitions on the instances of its context classifier, together with the operations that carry the transitions (OMG 2009). Only classifier of a

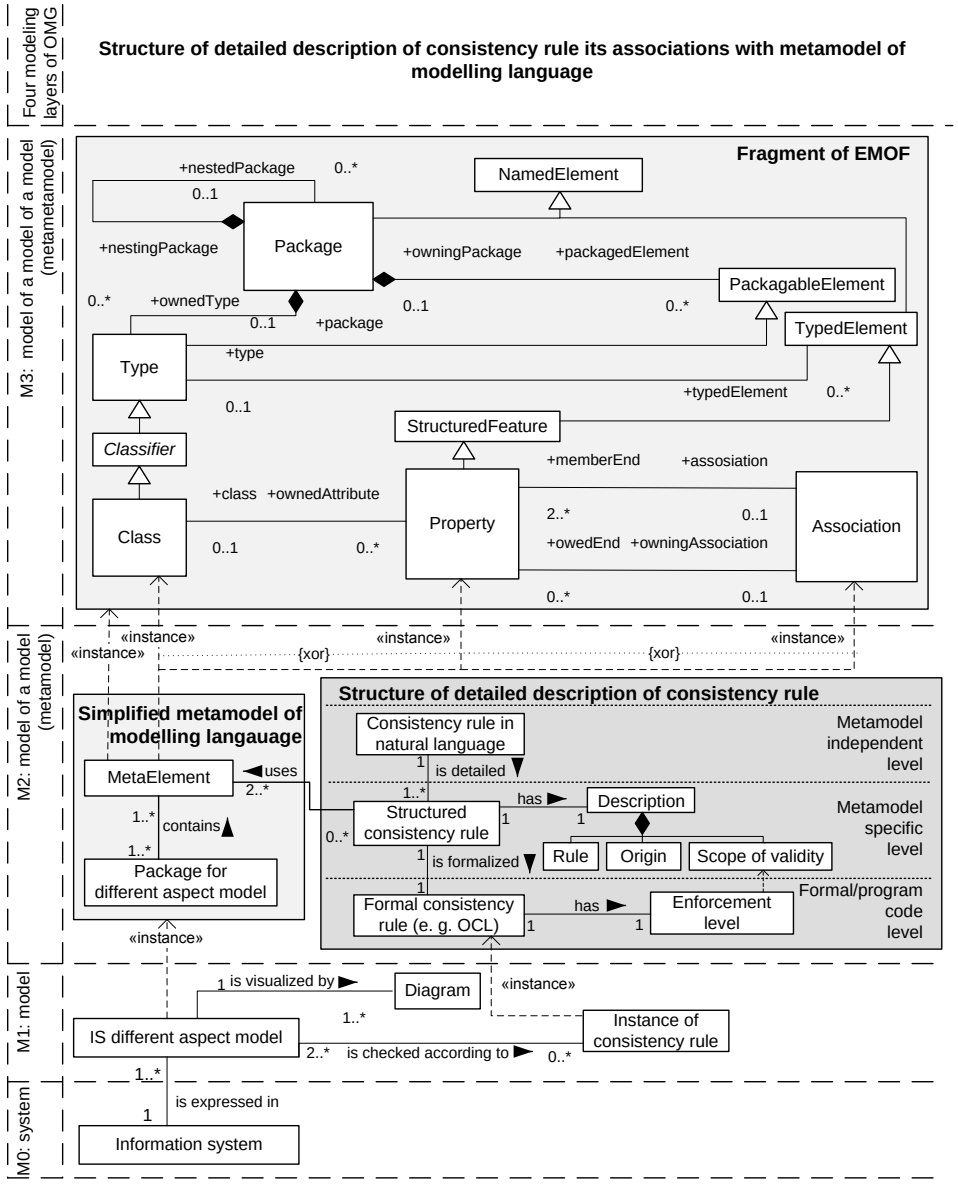

Fig. 1. Structure of detailed description of consistency rule its associations with metamodel of modelling language

class has operations; therefore, it can be derived that a protocol state machine is used to model states of classes. In this manner context – the 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 (OMG 2009) and OOAD method. According to OOAD, significant

changes of the state of the object (described by the class) are modelled using state diagrams (Bennet *et al.* 2010).

The scheme of the proposal is presented in Fig. 1. Description of consistency rule does not belong to metamodel level (it is not metamodel of instances of consistency rules). But it associates elements of metamodel of modelling language (Fig. 1).

The main ideas of the proposal are as follows:

1. Check consistency of semi-formal IS models using consistency rules;
2. Define consistency rules among different aspects IS models according to these **requirements**:
   - 1.1. Define consistency rules at three abstraction levels: metamodel, independent, metamodel specific and formal/program code.
   - 1.2. Verify consistency rules according to a metamodel of modelling language.
   - 1.3. Motivate the necessity of rules defining its origin.
   - 1.4. Assign enforcement level to consistency rules according to their scope of application.

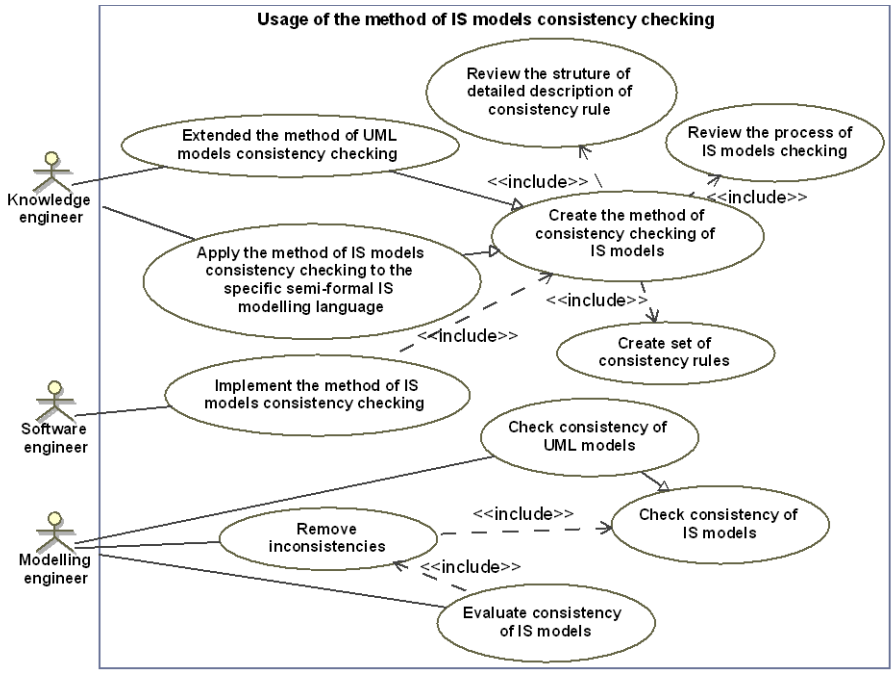

Fig. 2. Use case diagram of the proposed method

Having achieved that the proposed method would better correspond to the existing OMG standards, it is defined using:

- *Four modelling layers architecture* (M0, M1, M2, M3), which OMG uses for its standards, like MDA.
- *Essential MOF* (EMOF), which is one of two compliance points (EMOF and CMOF (Complete MOF)) of MOF. MOF is an OMG standard (OMG, 2011) that defines the language to define a modelling language. A primary goal of EMOF is

to allow simple metamodels to be defined using simple concepts while supporting extensions for more sophisticated metamodelling using CMOF.

- The idea of modelling is based on *three levels* applied *from OMG MDA* standard (OMG, 2003).

The usage of the method is presented in Fig. 2 by use case diagram.

Below in Table 3 the main use case is described.

Table 3. Use case "Apply the method for IS models consistency checking to the specific semi-formal IS modelling language" description

Use case name	Apply the method for IS models consistency checking to the specific semi-formal IS modelling language
Unique ID	UC2
Actor(s)	Knowledge engineer
Brief description	Knowledge engineer creates a method for checking consistency of semi-formal models (except UML models because they are included in a separate use case UC1).
Preconditions	There is a necessity to check consistency of specific semi-formal models. Knowledge engineer has enough knowledge about the specific language. The chosen specific language allows modelling a system from various perspectives. Knowledge engineer knows any formal modelling language.
Main flow	1. Examine the method for IS models consistency checking. 2. Collect data about new consistency rules using elicitation methods. 3. Specify rules at a metamodel-independent level. 4. Define elements of a metamodel related by the rule. 5. Specify a rule at the metamodel specific level. 6. Express simpler rules in a formal language. 7. Define the enforcement level of the rule. 8. Explain the necessity, enforcement level (see the step above) of the rule). 9. Repeat 3–8 steps for each rule. 10. If it is necessary, modify the proposed process of IS models checking.
Alternative flows	If the selected semi-formal language is UML, then forward to task "Extend the method of UML consistency checking". Consistency rules can be defined on class or/and attribute or/and association of metamodel. Enforcement level of the rule can be low, medium, high.
Post conditions	The method for checking consistency of IS models expressed in a semi-formal language is created.

4. Application of the Proposed Method

This section presents the experiment for checking understandability of the proposed method. Some parts of this experiment are published in scientific publications (Vasilecas et al., 2011; Dubauskaite and Vasilecas, 2010).

In the experiment we demonstrate how various consistency rules from different papers (Egyed, 2007; Sapna and Mohanty, 2007; Chanda et al., 2009) and our rules (specified using the proposed requirements) are understood by analysts, designers, programmers, and quality engineers. The researches of Egyed (2007), Sapna and Mohanty (2007) and Chanda et al. (2009) are selected for the experiment because their approaches are the most similar to our proposal compared to other analysed related researches. The experiment is performed using questionnaires. The questionnaire consists of 9 rules from the researches (Egyed, 2007; Sapna and Mohanty, 2007; Chanda et al., 2009) and our proposed rules without saying which rule is from which source, and questions as presented in Table 4.

Table 4. Questionnaire

1.1 Do you understand semantic of the rule?	*1.2 Do you know what metaelements are associated by the rule?*	*1.3 Does the rule conform to OMG UML metamodel?*	*1.4 Is the rule reliable (known origin) and necessary (description of application)?*
☐ Yes ☐ May be ☐ No	☐ Yes ☐ May be ☐ No	☐ Yes ☐ May be ☐ No	☐ Yes ☐ May be ☐ No

In this study the questionnaire is filled in by 14 specialists that have theoretical or/and practical knowledge about UML. The participants are from various companies.

Table 5. Application of the paired t-test

Input	The 14 paired samples obtained having calculated a total number of answers 'yes' (to the questions about unambiguity and reliability of consistency rules from two 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	H_0: The proposed method has the same quality (unambiguity and reliability) as the previous method.
H_1	H_1: The proposed method has better quality (more answers 'yes' about unambiguity and reliability) compared with previous method.
Calculations	Based on the data it can be seen that $n = 14$. The mean of differences is $\bar{d} = 3{,}143$ (Formulas are provided in (Wohlin et al. 2000)). It can be identified that $S_d = 3{,}931$ and $t_0 = 4{,}011$.
Conclusions	A number of degrees of freedom is $f = n - 1 = 14 - 1 = 13$. In Table A1 from (Wohlin et al., 2000) book, $t_{0.025,\ 13} = 2.160$. $t_0 = 4{,}011 > 2.160 = t_{0.5,\ 13}$ therefore H_0 is rejected and H_1 is accepted with 95% (100%-5%) confidence level.

The study is based on the initial hypothesis that the proposed method is not better than the previous methods of specifying consistency rules.

The collected data were processed using t-test method (Table 5).

Fig. 3 demonstrates that IS engineers with different qualification understand the proposed consistency rules better compared to the previous rules.

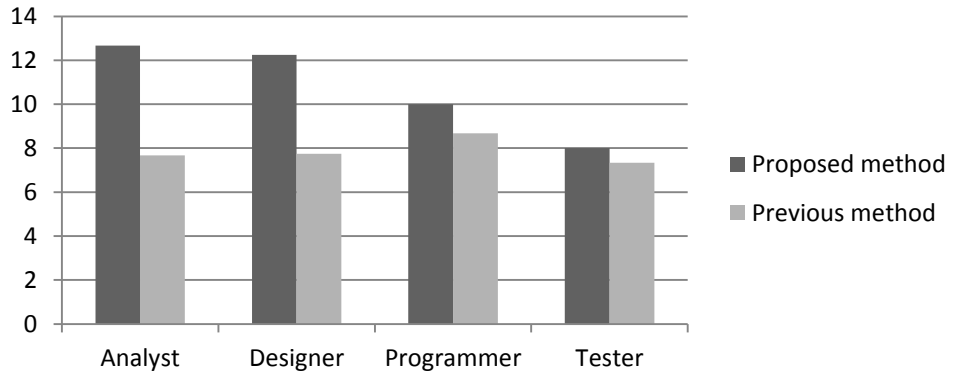

Fig. 3. Understanding of different methods by specialists with various qualifications – counting of "yes" answers

Additionally the comparison of our proposed method with the three most similar methods of other researchers is provided in Table 6. As can be seen, our method fulfil more requirements than other three methods.

According to the results of evaluation of consistency rules specifications, the proposed method is better than the other method of specifying rules. It allows understanding rules less ambiguously because their semantic is more understandable and the associated metaelements are known. The rules are also more reliable because their origin is known and they conform to the metamodel.

Table 6. Evaluation of the proposed and three similar methods according to specific features

Feature/Comparison Criteria	**Methods**			
	Egyed, 2007	**Sapna and Mohanty, 2007**	**Chanda et al., 2009**	**Our method**
1. Technique of checking consistency of IS models	Consistency rules hard coded to UML Analyzer tool	OCL, SQL	Context free grammar	OCL, java and other executable language
2. Language for expressing IS models	UML	UML	UML	Semi-formal modelling language, UML is included

Feature/Comparison Criteria		Methods			
		Egyed, 2007	**Sapna and Mohanty, 2007**	**Chanda et al., 2009**	**Our method**
3. Is a process of checking consistency defined?		Partially	Partially	Partially	Yes
Specification of consistency rules	4. Are requirements of consistency rules defined?	No	No	No	Yes
	5. Are examples of UML consistency rules provided?	Yes	Yes	Yes	Yes
	6. Is a set of consistency rules qualitative[2]?	No	No	No	Partially
Implementation of consistency rules	7. Is there a tool for automating process of IS models consistency checking?	Yes	No	No	Yes
	8. Is an enforcement level of the detected violation of a consistency rule provided[3]?	No	No	No	Yes

Comparison of total count answers 'yes' shows that the quality of the proposed rules is by approximately 40,74% better than the quality of consistency rules specifications provided in previous researches (Egyed, 2007; Sapna and Mohanty, 2007; Chanda et al., 2009).

5. Conclusions

Analysis of consistency rules shows that most rules are expressed in natural and formal language. Rules expressed in natural language may be interpreted ambiguously. Formal rules usually use their own description of UML models. Therefore, it remains unclear what elements of an OMG UML metamodel they conform to. Moreover, some

[2] Unambiguous, known origin and practical necessity, conformance to the metamodel of the modelling language.

[3] It indicates the necessity of performing changes of models according to a consistency rule.

consistency rules do not conform to an OMG UML metamodel, and their practical necessity is doubtful.

The analysis of UML design tools demonstrates that most of them allow developing models that do not conform to the UML metamodel. It means that consistency rules have to associate metaelements from different aspects of models despite the fact that they are directly associated in a metamodel.

The rule-based method for consistency checking in IS models is created. It is free from modelling language and is applied to UML. The feasibility of the proposed method is illustrated creating 12 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 a 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.

References

Bennet, S., McRobb, S., Farmer R. (2010). *Object-Oriented Systems Analysis and Design Using UML*. 4th ed. London.

Berkenkötter, K. (2008). Reliable UML Models and Profiles. *ENTCS*, 217, 203–220.

Borba, C.F., Silva, A.E. (2010). Knowledge-Based System for the Maintenance Registration and Consistency among UML Diagrams. *LNCS*, 6404, 51–61.

Cavarra, A., Riccobene, E., Scandurra, P. (2004). A framework to simulate UML models: moving from a semi-formal to a formal environment. In: *Proc. of the 2004 ACM Symposium on Applied Computig (SAC'04)*, New York, pp. 1519–1523.

Chanda, J., Kanjilal, A., Sengupta, S., Bhattacharya, S. (2009). 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 (ICM2CS09)*, New Delhi, pp. 1–4.

Cheng, H.C. (2001). A Metamodel-Based Approach to Formalizing UML. In: *Proc. of the 25th Annual International Computer Software and Applications Conference (COMPSAC'01)*, Wasshington.

Dubauskaite, R., Vasilecas, O. (2010). The approach of ensuring consistency of UML model based on rules. In: *Proc. of the 11th International Conference on Computer Systems and Technologies (CompSysTech'10), Sofia*, ACM Press, 471, pp. 71–76.

Dubauskaite, R., Vasilecas, O. (2013). Method on specifying consistency rules among different aspect models, expressed in UML. *Electronics and electrical engineering*, 19(3), 77–81.

Egyed, A. (2007). Fixing inconsistencies in UML design models. In: *Proc. of the 29th International Conference on Software Engineering (ICSE 2007)*, New York, pp. 292–301.

Ha, I., Kang, B. (2008). Cross checking rules to improve consistency between UML Static diagram and Dynamic Diagram. In: Fyfe, C et al. (Eds.): IDEAL 2008, *LNCS*, 5326, pp. 436–443.

Ibrahim, N., Ibrahim, R., Saringat, M.Z., Mansor, D., Herawan, T. (2011). Consistency Rules between UML Use Case and Activity Diagrams Using Logical Approach. *International Journal of Software Engineering and Its Applications*, 5(3), 119–134.

ISO/IEC 1997. *Information Technology – Software quality characteristics and metrics* – Part 3: Internal Metrics. International Organization for Standardization and International Electrotechnical Commission.

Kotulski, F.L. (2007). 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, pp. 51–58.

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

Lucas, F.J., Molina, F., Toval, A. (2009). A systematic review of UML model consistency management. *Information and Software Technology*, 51, 1631–1645.

Matta, A., Furia, C., Rossi, M. (2004). Semi-formal and formal models applied to flexible manufacturing systems. *LNCS*, 3280, 718–728.

OMG (2003). *MDA Guide Version 1.0.1*, `http://www.omg.org/cgi-bin/doc?omg/03-06-01`.

OMG (2009). *Unified Modelling Language (OMG UML), version 2.2*, `http://www.omg.org/spec/UML/2.2/Superstructure/PDF`.

OMG (2009a). *Common Variability Language (CVL)*, `http://www.omgwiki.org/variability/doku.php`.

OMG (2011). *Meta Object Facility (MOF) Core Specification*, `http://www.omg.org/spec/MOF/2.4.1`.

Paige, R.F., Brooke, Ph.J., Ostroff, J.S. (2007). Metamodel-based model conformance and multiview consistency checking. *Transactions on Software Engineering and Methodology*, 16(3), 11.

Rasch, H., Wehrheim, H. (2003). Checking Consistency in UML Diagrams: Classes and State Machines. Formal Methods for Open Object-Based Distributed Systems. *LNCS*, 2884, 229–243.

Rozanski, N., Woods, E. (2005). *Software System Architecture*. London, 546 p.

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

Shen, W., Compton, K., Huggins, J.K. (2002). A Toolset for Supporting UML Static and Dynamic Model Checking. In: *Proc. of the 26th International Computer Software and Applications Conference, Prolonging Software Life: Development and Redevelopment (COMPSAC 2002)*, Oxford, pp. 147–152.

Shinkawa, Y. (2006). Inter-Model Consistency in UML Based on CPN Formalism. In: *Proc. of the 8th Asia Pacific Software Engineering Conference (APSEC'06)*, Washington, pp. 411–418.

Shuzhen, Y., Shatz, S.M. (2006). Consistency Checking of UML Dynamic Models Based on Petri Net Techniques. In: 15th International Conference on Computing (CIC '06), pp. 289–297.

Silingas, D., Butleris, R. (2009). Towards Implementing a Framework Modelling Software Requirements in MagicDraw UML. *Information Technology and Control*, 38(2), 153–164.

Simmonds, J., Bastarrica, C.M. (2005). A tool for automatic UML model consistency checking. In: *Proc. of the 20th IEEE/ACM International Conference on Automated Software Engineering*, New York, pp. 431–432.

Simmonds, J., Bastarrica, C.M. (2005a). *Description Logics for consistency checking of architectural features in UML 2.0 models*. DCC Technical Report TR/DCC-2005-1, Chile.

Straeten, R.V.D. (2004). Inconsistency detection between UML models using racer and nRQL. In: *the Third International Workshop on Applications of Description Logics (ADL'04)*, Germany.

Straeten, R.V.D., Simmonds, J., Mens, T., Jonckers, V. (2003). Using Description Logic to Maintain Consistency between UML Models. *LNCS*, 2863, 326–340.

Taentzer, G. (2004). AGG: A Graph Transformation Environment for Modeling and Validation of Software. In: *Pfaltz, J.L., Nagl, M., Böhlen, B. (Eds.) 2nd International Workshop (AGTIVE 2003)*, Charlottesville, USA, *LNCS*, 3062, pp. 446–453.

Usman, M., Nadeem, A., Tai-hoon K., Eun-suk C. (2008). A Survey of Consistency Checking Techniques for UML model. In *Proc. of the 2008 International Conference on Advanced Software Engineering & Its Applications (ASEA 2008)*, Hainan, pp. 57–62.

Vasilecas, O., Dubauskaite, R., Rupnik, R. (2011). Consistency checking of UML business model. *Technological and economic development of economy* 17(1), 133–150.

Wang, Sh., Jin, L., Jin, Ch. (2006). Ontology Definition Metamodel based Consistency Checking of UML Models, In: 10th International Conference on Computer Supported Cooperative Work in Design (CSCWD '06), pp. 1–5.

Wang, Z., He, H., Chen, L., Zhang, Y. (2012). Ontology based semantics checking for UML activity model. *Information Technology Journal*, 11(3), pp. 301–306.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. United Kingdom, Springer.

Authors' information

D. Kalibatiene, dr., is a full time Associate Professor at the Department of Information Systems and a researcher at the Information Systems Research Laboratory of Vilnius Gediminas Technical University. She participated in the project "Business Rules Solutions for Information Systems Development (VeTIS)" of the High Technology Development Programme. She is the member of the European Committee and Lithuanian Government supported SOCRATES/ERASMUS Thematic Network projects "Doctoral Education in Computing" (ETN DEC) and "Teaching, Research, Innovation in Computing Education" (ETN TRICE). She is the author and co-author of more than 30 papers and 1 book in the field of information systems development. Research interests: development of information systems based on business rules and ontology and conceptual modelling.

O. Vasilecas, Prof. dr., is a full time professor at the Department of Information Systems and a principal researcher and the Head of Information Systems Research Laboratory of Vilnius Gediminas Technical University. He is the author and co-author of more than 250 research papers and 5 books in the field of information systems development. His research interests include knowledge, represented by business rule and ontology, information systems development. He delivered lectures in 7 European universities including London, Barcelona, Athens and Ljubljana. Prof. Vasilecas is constantly invited to give training sessions at universities of Germany, Holland, China, Latvia and Slovenia. He supervised 10 successfully defended doctoral theses and currently is supervising 4 additional doctoral students. He was the leader of a number of international and local projects. The latest project under his management was entitled "Business Rules Solutions for Information Systems Development (VeTIS)", which was carried out under the High Technology Development Programme of Lithuania.

R. Dubauskaite, dr., is a lector at the Information System Department in Vilnius Gediminas Technical University. She participated in the High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)". She is author of 11 papers in the field of information systems development. Research interests: consistency of information system model and consistent conceptual modelling based on rules.

Received May 16, 2013, revised June 27, 2013, accepted July 2, 2013