Taylor & Francis
Taylor & Francis Group

TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
ISSN 2029-4913 print/ISSN 2029-4921 online

2011 Volume 17(1): 133–150
doi: 10.3846/13928619.2011.554029

CONSISTENCY CHECKING OF UML BUSINESS MODEL

Olegas Vasilecas[1], Rūta Dubauskaitė[2], Rok Rupnik[3]

[1, 2]Information System Department, Vilnius Gediminas Technical University,
Saulėtekio al. 11, LT-10223 Vilnius, Lithuania
[3]Faculty of Computer and Information Science, University of Ljubljana,
Trzaska 25, 1000 Ljubljana, Slovenia
E-mails: [1]olegas.vasilecas@vgtu.lt (corresponding author); [2]ruta.dubauskaite@vgtu.lt;
[3]rok.rupnik@fri.uni-lj.si

Received 22 April 2010; accepted 7 December 2010

Abstract. Unified modelling language (UML) is often used in practice for modelling business system (BS) by various aspects. UML model of business system consists of different aspect models and their usage for information system (IS) design is related with inconsistency problem. It arises because ambiguous or even contradictory information are provided in different aspect models. The paper presents approach in ensuring UML model consistency. Several examples of consistency rules are included to the paper to illustrate how approach is working. Developed prototype of suggested approach is applied in a domain of enterprise manufacturing windows and doors. Obtained results are discussed.

Keywords: consistency, UML model, UML diagram, aspect model, rules, consistency checking, consistency ensuring, model validation.

Reference to this paper should be made as follows: Vasilecas, O.; Dubauskaitė, R.; Rupnik, R. 2011. Consistency checking of UML business model, *Technological and Economic Development of Economy* 17(1): 133–150.

JEL Classification: C52, C51, E61, D83, L86.

1. Introduction

UML is dominant modelling language for specifying, designing and documenting artefacts of business system (Chen and Motet 2009). Complete conceptual model includes a series of different aspect models for example, static structure model, states machines and etc. (Gudas 2009). The developed multiple view (aspect models) of IS can contain inconsistent or even contradictory specifications. 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). The consistency problem is widely discussed in the publications of recent years. It shows the importance of the question of ensuring consistency of IS model in design phase. Model consistency issue is particularly important within the scope of model-driven engineering (MDE) (Lucas *et al.* 2009). MDE puts models into the centre of the IS development process as the source of transformation to platform-specific models, which are used for code generation (Mokhati *et al.* 2007). Unambiguous models are necessary for the successful accomplishment of the tasks of model transformation and finally for code generation (Berkenkötter 2008). Correct working of software systems is very important for enterprises, because programs help to manage business systems, e.g. indicating problematic transport zones (Jakimavičius and Burinskienė 2009), monitoring sewage and analysis of water recourses (Dzemydienė and Dzindzalieta 2010; Kalibatienė and Vasilecas 2010) and etc.

The objective of this paper is to improve the consistency of IS models, expressed in UML. The main task is to extend the existing approaches of ensuring UML models consistency. Several consistency rules of UML model are presented. In the paper the tool supporting the proposed approach is introduced and the process of automated UML model checking is demonstrated.

The paper is organised as follows. Section 1 introduces the issue to be analysed. Section 2 briefly presents consistency types and approaches for ensuring UML model consistency. Section 3 introduces the suggested approach of ensuring UML model consistency based on set of rules and its implementation in NoMagic MagicDraw UML tool. Section 4 presents the experiment performed to evaluate the suggested approach. Section 5 concludes the paper.

2. Related Work

Consistency concept, its types and approaches of checking UML model consistency are presented in the section.

2.1. Consistency

Several definitions of model consistency appear in existing literature. In general consistency means that the structures, features and elements that appear in one model are compatible with the content of other models (Rozanski and Woods 2005).

Consistency can be classified to vertical (inter-model), horizontal (intra-model), evolution, semantic or syntactic consistency.

Vertical or inter-model consistency is checked at different levels of abstraction between different aspect models (Lucas *et al.* 2009; Usman *et al.* 2008). Horizontal or intra-model consistency can be defined as 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 (Van Der Straeten *et al.* 2003).

All mentioned types of consistency can express syntactic or semantic conformance of different aspect models. Syntactic consistency expresses matching of model to the specifications

of metamodel. Semantic consistency requires that model would be compatible to semantic meanings defined by metamodel (Lucas *et al.* 2009; Usman *et al.* 2008).

In this paper, we are concentrated on improving models syntactic horizontal consistency. The examples of horizontal consistency requirements are:

- *Inv1*: Call message of sequence model should have operation assigned. It means the operation of object called in sequence model should appear in class model.
- *Inv2*: Private operation of class model can not be called from foreign classifier in sequence model. *Inv2* extends requirement *Inv1*. It requires that called operation in sequence model, that appear in class model would not be private. Because operation of private visibility is visible only to owner of this operation and foreign classes can not see and call it.

In the following section we discuss several issues that are also important for the area of consistency: the way how various consistency requirements can be expressed, the way how IS model is checked and etc.

2.2. Approaches in Ensuring UML Model Consistency

In this section approaches of ensuring UML model consistency are presented. UML was chosen because it is likely the most popular modelling language (Silingas and Butleris 2009) and there are many modelling tools supporting UML (Shen, Compton, Huggins 2002). The second reason UML was chosen it was developed by OMG (Object Management Group), which also introduce MDA (Model Driven Architecture). Consistency of UML model is especially important in MDA, for automatic transformation initial model to specific model and finally code generation tasks.

Approaches of checking UML model consistency are compared according to the following parameters:

- *Paradigm of ensuring consistency* of UML model shows general approach used for ensuring consistency;
- *Technique*, language or approaches used in the analyzed paper to ensure consistency;
- *Scope of constrained model elements* shows if constraint is defined for
  - *one aspect model*,
  - *relationship* of different aspect models. Constraint on relationship requires that element of one aspect model would be in another aspect model too;
- *Abstraction level used for ensuring consistency* defines if consistency is ensured
  - at *metamodel* level of UML,
  - at *formal language level* (consistency conflicts are detected by inference mechanism of formal language).
- *CASE tool* used for the implementation and testing of suggested approach.

The comparison of analyzed approaches on defined parameters is provided in the Table 1.

The analyzed approaches of checking UML model consistency can be divided to two groups. First group of approaches based on constraints. Constraints are defined for UML metamodel and UML models are checked according to these constraints. The authors of papers (Chen

Table 1. Comparison of approaches in ensuring UML model consistency

No.	Paradigm of consistency ensuring	Author Year [reference]	Technique of ensuring consistency	Scope of constrained model elements	Abstraction level used for ensuring consistency	CASE tool
1	Use of constraints	Chen and Motet 2009	Controlling grammar in XMI format	one aspect model	Metamodel of UML	–
2		Berkenkötter 2008	OCL	one aspect model	Metamodel of UML	USE tool
3		Chiorean *et al.* 2004	OCL	one aspect model	Metamodel of UML	OCLE
4		Egyed 2007	Consistency rules hard coded to UML Analyzer tool	relationship of different aspect models	Metamodel of UML	Rational Rose with integrated UML Analyzer
5		Pakalnickiene and Nemuraite 2007	OCL	one aspect model	Metamodel of UML	MagicDraw UML
6	Use of inference mechanism of formal language	Mokhati *et al.* 2007	Rewriting logic (Maude)	relationship different aspect models	Formal language	Maude, modelCheck function
7		Van Der Straeten *et al.* 2003	Description logic	relationship different aspect models	Formal language	Loom
8		Simmonds and Bastarrica 2005	Description logic	relationship different aspect models	Formal language	Poseidon for UML, MCC plug-in

and Motet 2009; Berkenkötter 2008; Shen *et al.* 2002; Egyed 2007) suggest defining constraints for every different aspect model. But constraints among different models are not defined. The Eyged approach (Egyed 2007) uses constraints on relationships of different aspect models. But the defined consistency rules are hard coded to UML Analyzer tool and can not be analyzed. The second group of approaches is based on formal models (Mokhati *et al.* 2007; Van Der Sraeten *et al.* 2003; Pakalnickiene and Nemuraite 2007). Authors of these approaches propose transformation of semi-formal UML models to formal models. In this case it is suggested to detect consistency conflicts using inference mechanism of formal language. Despite of formal models are more precise, but they are less understandable in comparison with semi-formal languages, which can be presented using diagrams (Vavpotič and Bajec 2009).

The analysis of related work shows that the approaches solve inconsistency of models problem in some extent. Formal models are often too difficult to understand to be used in practice. Semi-formal UML models are widely used, but their constraints are proposed only for one model, relationships among models are not defined. The Egyed approach (Egyed 2007) includes constraints on relationships of different aspect models, but the defined consistency rules are hard coded to CASE tool and are not expressed in formal language, which would be independent from platform specific features. Despite of many results that have been achieved in the field of ensuring models consistency is still open and relevant.

The authors of this paper suggest extending the approach based on constraints of UML metamodel by adding consistency rules among different aspect models. Consistency rule means constraint on relationship of different aspect models. It is suggested consistency rules to express in OCL in order the approach would be more general and more applicable in various tools.

The following section presents the suggested approach in detail.

3. The Approach in Ensuring UML Model Consistency Based On the Rules

The following sections describe the suggested general framework of IS model, approach of ensuring UML model consistency, explain several proposed consistency rules in detail and presents the implementation of the approach.

3.1. The Suggested General Framework of IS Model

A review of existing literature indicates a wide variety of consistency definitions. Sometimes consistency term is used for expressing conformance of diagrams, sometimes of models. In this section the authors of paper suggest general framework of IS model. The main purpose of this suggestion is contributing more clearness to concept of consistent model, emphasising IS model relationships with consistency, aspect models and diagrams concepts.

Fig. 1 shows the place of horizontal, evolution and vertical consistency in IS model graphically. Remark that all horizontal relationships of models are not displayed for reasons of clarity.

IS model consists of several different aspect models. Developing models of higher level is based on lower level models. The authors of (Bajec *et al.* 2003; Bajec and Krisper 2005)

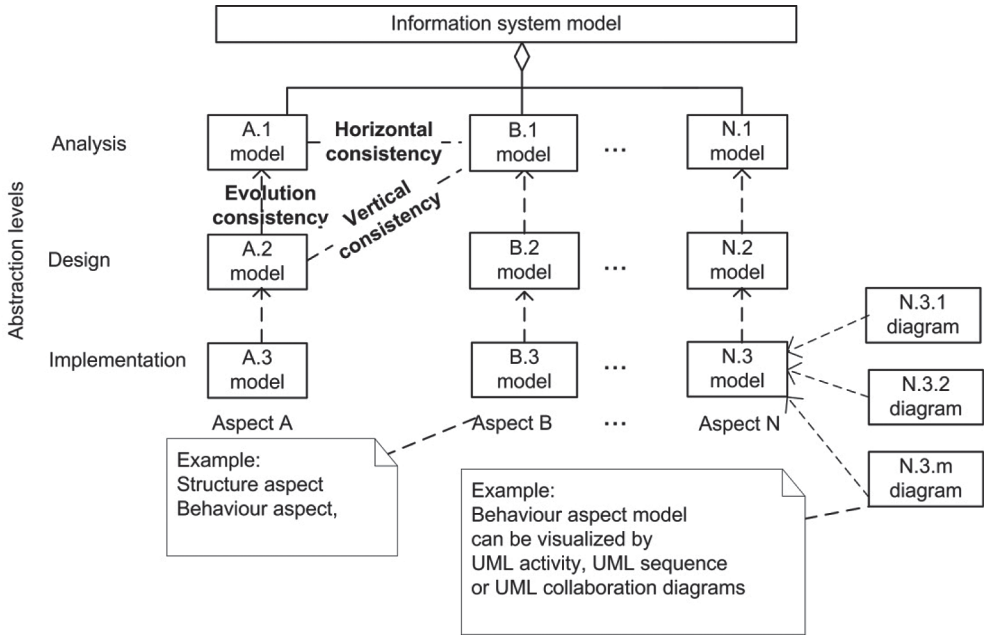

Fig. 1. Horizontal, evolution and vertical consistency of IS model

propose method how to keep motivation aspect model (business rules) at the business level inline with the rules that are implemented at the system level.

Every aspect model can be visualised by several different diagrams. Every diagram is based on model. It means if model is consistent, diagram also is consistent. Therefore the main issue is to ensure consistency of IS model.

The presented general framework of IS model enables clear conception of horizontal, vertical and evolution consistency of IS model and diagrams.

3.2. The Suggested Approach of Ensuring UML Model Consistency

In this section we present our approach of ensuring UML model consistency. The some details of approach are presented in Table 2.

Table 2. Consistency ensuring of UML model

Paradigm of consistency ensuring	Use of constraints
Technique of consistency ensuring	OCL
Abstraction level used for ensuring consistency	Metamodel of UML
Scope of constrained model elements	Relationships of different aspect models and different aspect models
CASE tool (used and extended)	MagicDraw UML

It is necessary to mention that constraints are also defined in specification of UML metamodel (OMG 2007), but they constrains only separate aspect model. In our suggested approach constraints are on relationships of different aspect models and separate aspect models too.

The reasons why we suggested:

- to check consistency of UML model,
- to define constraints on relationships at metamodel level,
- to express consistency rules in OCL are explained bellow.

We proposed checking consistency of IS model, which consists of several different aspect models, presented in semi-formal language, in this case in UML:

- Do not perform additional UML model transformation to formal model task. The formal model allows detecting inconsistencies, by inferences mechanisms of formal language, but transformation task requires additional time,
- Besides UML is more understandable and more usable in practice than formal modelling languages.

It is also suggested defining constraints on relationships of different aspect model at metamodel level (Vasilecas and Dubauskaite 2009). Sometimes constraints are called consistency rules. The ensuring consistency in higher level makes the consistency rules more usable, because meta-level constraints are independent from specific implementation platform (Bajec and Vavpotič 2008). It is necessary to stress that at metamodel level we suggest defining only general constraints on different aspect models and their relationships, while domain specific consistency rules are defined for every model of specific IS. More information about domain specific rules are presented the papers (Vasilecas and Lebedys 2007; Nemuraite *et al.* 2008; Smaizys and Vasilecas 2009). The authors of (Vasilecas and Lebedys 2007) research derivation of domain rules from IS models, and usage them for data validation. While the paper (Nemuraite *et al.* 2008) presents method for checking aspect model based on rules.

We suggested to express consistency rules of UML model in OCL. The main reasons for choosing this language are:

- OCL is part of UML. According to OMG OCL specification Object Constraint Language is used to describe expressions on UML models (OMG 2006),
- OCL is formal language. It means the constraints can be interpreted unambiguously.

The structure, relationships of elements of proposed approach is shown in Fig. 2 using UML class diagram.

Consistency of IS model, expressed in UML, is checked according to defined consistency rules. Consistency rules describe conditions that all UML models must satisfy them to be considered valid (Egyed 2007). Consistency rules, expressed in OCL, constrain every aspect model, and relationships of different aspect model. Aspect model is part of IS model. Consistency rules are defined for UML metamodel. IS model also conforms to UML metamodel. Diagrams, that visualizes all or part of IS model, are based on UML metamodel too. Diagrams are included to our suggested approach of IS consistency, because developer often models information system using diagrams, but in order diagrams would present unambiguous aspects of IS system, first of IS model should be consistent.

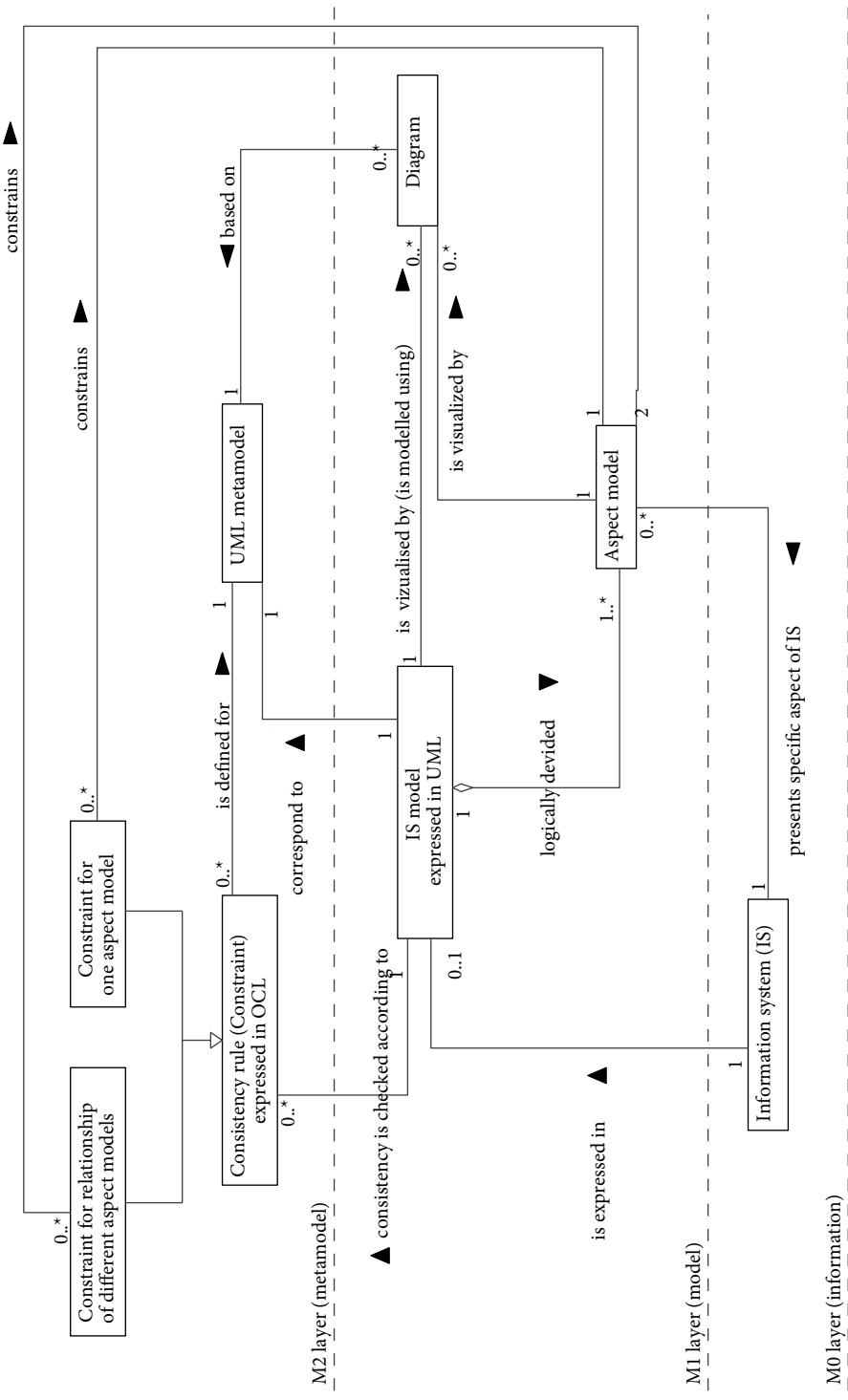

Fig. 2. Approach in ensuring consistency based on rules

The process of checking IS model consistency is presented by authors of this paper (Dubauskaite and Vasilecas 2009b). The main activities are checking model according consistency rules and appending list of violations of consistency. The removing of detected consistency conflicts can allow to improve consistency of UML model.

Detecting consistency conflicts in design phase can help to minimize the cost of correction of inconsistencies during the development process compared with cost of correction of conflicts detected in late phases of IS development.

The novelty of the work lies in the fact we suggest new method for ensuring consistency of information system model:

- we proposed to check consistency of original semi-formal UML model,
- the approach includes both constraints (consistency rules) on relationships of different aspect model and constraint on different aspect models,
- these consistency rules are expressed explicitly in OCL,
- consistency rules are defined for UML metamodel.

3.3. Consistency Rules of UML Model

In this section we provide three examples consistency rules that illustrate the usage of the suggested method. According to our suggested the method of ensuing consistency of IS model consistency rules have to be defined for every aspect model and for relationships of different aspect models.

Examples of suggested consistency rules for relationships of different aspect model elements are presented in the Table 3. Consistency rule for relationship of different aspect model defines coherence of elements from different aspect models. The main aspects

Table 3. Examples of suggested consistency rules

ID	Associated elements by consistency rule		Consistency rule
	Element of UML metamodel for static structure aspect model	**Element of UML metamodel for behaviour aspect model**	
1	*Operation* of class of static structure model. Static structure model can be visualized by class diagram	*Protocol Transition (Protocol-Transition)*, of protocol state model. Protocol state model graphically can be represented by protocol state machine diagram	Protocol transition of protocol state model has to be defined by operation
2	*Class* of static structure model	Element of protocol state model *ProtocolStateMachine*	Context of protocol states has to be defined
3	*Class* of static structure model	*Lifeline* of sequence model, which can be represented graphically by sequence diagram	The type of lifeline should be specified

of information system are static and behaviour aspect. Therefore consistency rules are provided for relationships of static and behaviour models. According to suggested approach consistency rules are defined for metamodel of UML. The related elements of static structure model and behaviour structure models are defined in 2nd and 3rd column of Table 3. The 1st column of Table 3 shows identification number of consistency rule, which is defined in column 4.

Consistency rule ID1 requires defining transition of states by specifying operation, which execution causes the changes of state.

The Fig. 3 bellow shows elements of UML metamodel for IS Class and Protocol State Machines models that are associated by consistency rule ID 1. Dashed lines with arrows present relationships of UML metamodel and IS model. Elements of different aspect models associated by consistency rule 1 are showed using dashed line without arrows. According to UML metamodel, which part is in Fig. 3 operation is not mandatory for protocol transition. But in practical situations it is need to know what operation execution causes the transition of states.

After analysis of UML metamodel specification and specific information system models we come to a conclusion, that it is necessary to define operation, which determine movement of object from one state to another state. Operation is defined in class model. Therefore consistency rule ID1 associates two different aspect models: class and state models.

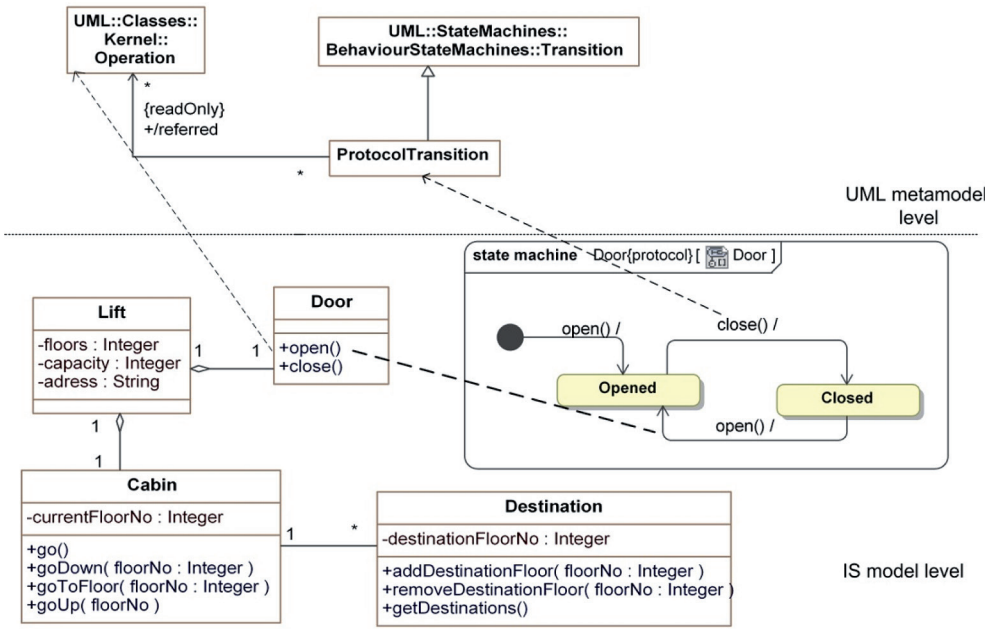

Fig. 3. Fragments of UML metamodel for IS static structure (class) and behaviour (Protocol State Machine) models

Consistency rule ID2 constrains a protocol state machine. This model presents the possible and permitted transitions on the instances of its context classifier, together with the operations that carry the transitions (OMG 2007). In this manner context – class, which operations can be called and their execution determines changes of states of object have to be defined. The source of this constraint is UML superstructure specification provided by OMG (OMG 2007).

The Fig. 4 shows elements of UML metamodel and IS model, associated by consistency rule ID 2.

According to **consistency rule ID3** the type of lifeline should be defined.

Type of lifeline shows associated class. When matching class in known then can be checked:

- If message of lifeline has associated operation of class (*Inv1*),
- If called message has public visibility (only public operations can be called by other objects) (*Inv2*) and etc.

We derived consistency rule ID3 concluding results of analysis of UML metamodel specification and MagicDraw UML tool. The constraints enumerated above this paragraph (*Inv1* and *Inv2*) are implemented in MagicDraw UML tool. They illustrate the necessity of our suggested consistency rule ID3. According to UML metamodel (Fig. 5) lifeline can be associated or not associated with a class (class is type of object). Therefore violation of consistency rule ID3 should be warning, but not error.

More details about our suggested approach and consistency rules are provided in (VeTIS 2009).

The software prototype, which implements our proposed approach and includes consistency rules, is presented in next section.

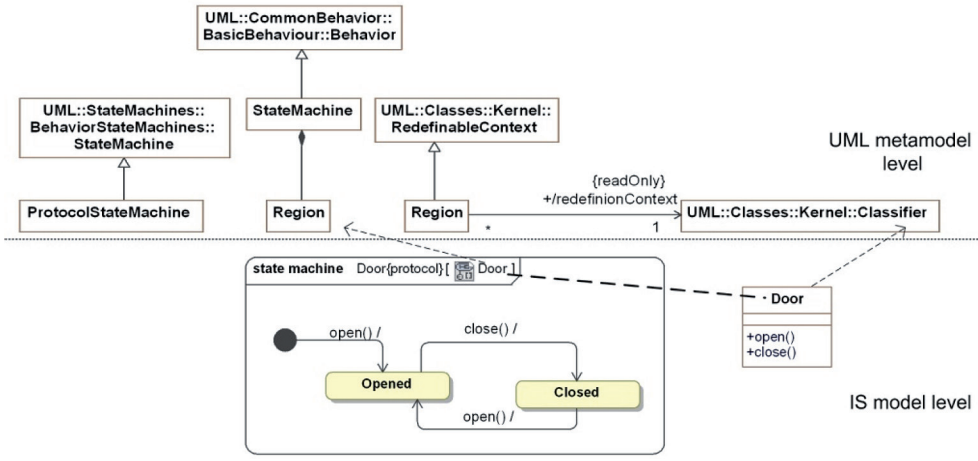

Fig. 4. Fragments of UML metamodel for IS behaviour
(Protocol State Machine, State Machine) and static structure (class) models

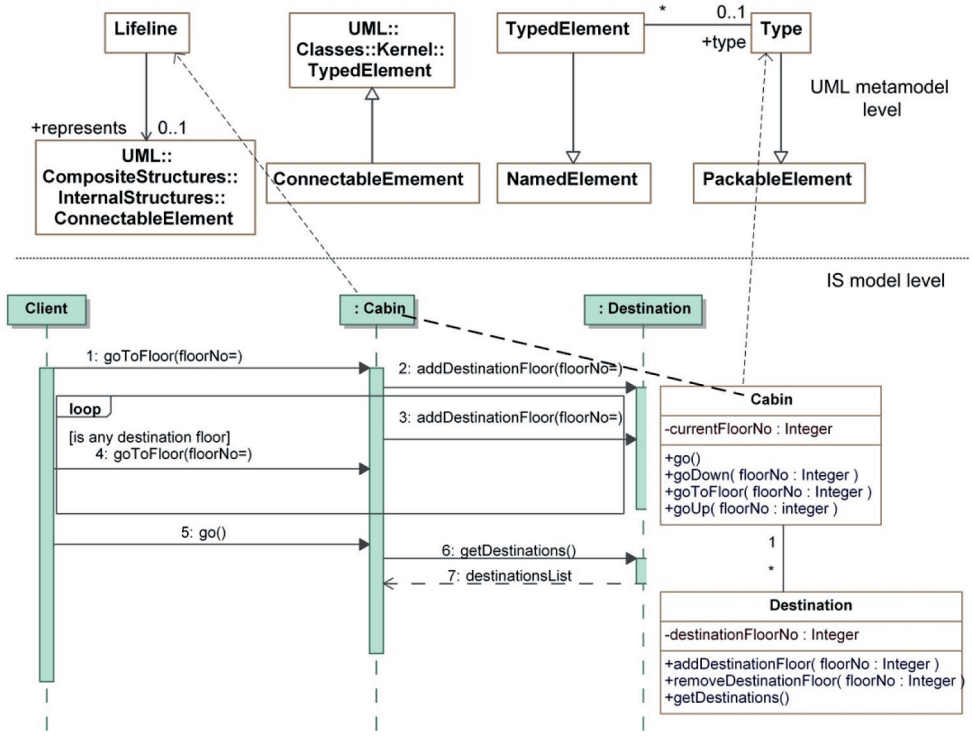

Fig. 5. Fragments of UML metamodel for IS behaviour (Sequence model) and static structure (class) models

3.4. The Implementation of Suggested Approach

There are commercial and non-commercial tools supporting checking of IS models. The known possibilities to check UML model consistency using MagicDraw UML and other CASE tools was presented in our paper (Dubauskaite and Vasilecas 2009a). The functionality of analysed tools is limited to validate only one aspect model and they usually do not allow ensuring consistency among several different aspect models. Therefore MagicDraw UML tool was extended with UML consistency constraints module, which contains consistency rules of different aspect models. MagicDraw UML is extensible tool, it can incorporate new consistency rules not only as module but also as plug-in.

Fig. 6 provides the specification of consistency rule ID1.

Consistency rule is expressed in OCL 2.0 (Specification part of Fig. 6), defined for element *ProtocolTransition* of UML metamodel (Constraint element part of Fig. 4). Implemented consistency rule ID1 relates state machine (more exactly transition of states) and class models (operation of class).

All implemented consistency rules are presented in documentation of project of business rules solutions for IS development (VeTIS 2009).

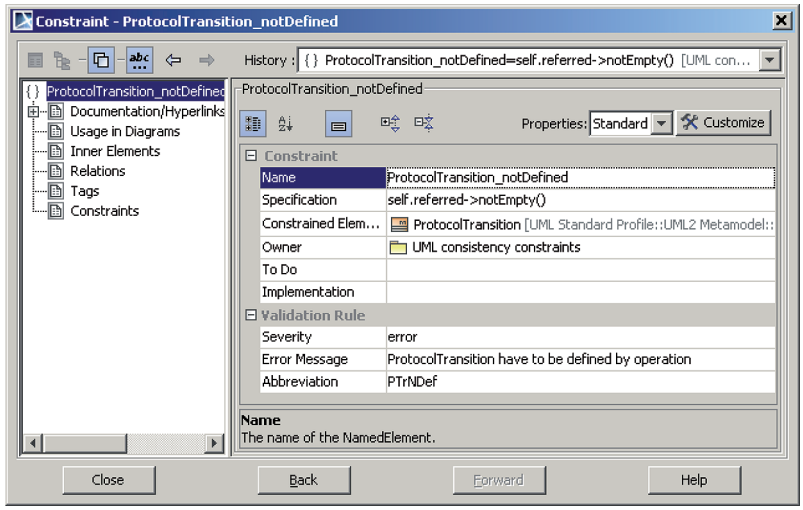

Fig. 6. Implementation of consistency rule

The tool do not automatically resolves consistency conflicts, because it does not know whether an inconsistency is tolerable or what choice of fixing inconsistencies is the best. However a tool can be assistant that help to find violations of consistency.

The usage of implemented prototype of consistency module of MagicDraw UML tool is presented in next section.

4. Experiment

A simple experiment is presented in this section to illustrate usage of suggested approach and demonstrate the functionality of created software prototype.

First activity of UML model consistency checking process is developing of the method with consistency rules and implementation it in a CASE tool (Dubauskaite and Vasilecas 2009b). Our suggested constraints, presented in section 3.3 are implemented as module of MagicDraw UML tool.

The prototype of UML consistency constraints module can be reused in test or real project by importing the developed consistency module to standard MagicDraw UML tool (Fig. 7).

Second step of ensuring consistency of UML model is validating developed concrete IS model according to every consistency rule. If consistency rule is detected list of consistency violations is appended with message of error or warning. Left column of Fig. 8 provides UML model, developed using MagicDraw UML 15.5 tool.

The UML test model consists of static structure model and behaviour model. Behaviour model is visualized by protocol states diagram. The diagram represents possible states of a class Door. Door is a part of lift business system. Accurate modelling of business system is necessary in order software, which is developed or generated automatically according

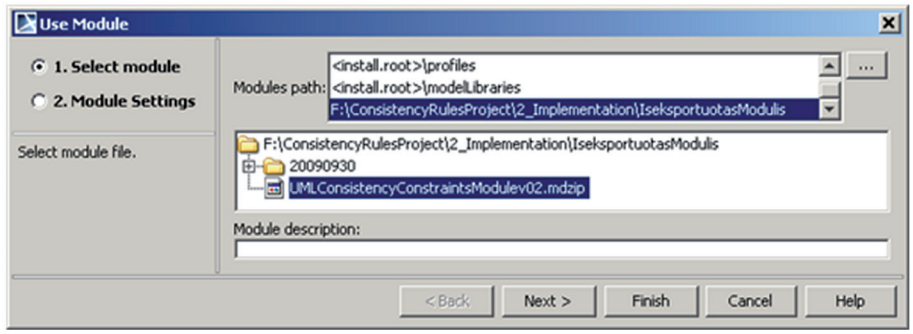

Fig. 7. Importing the developed software prototype for checking model consistency

to model, manages business system correctly. The process of checking of business model, expressed using UML, is executed automatically (activated by command validate model). The detected consistency conflicts are shown in bottom of right column in validation results section of Fig. 8.

The last step of ensuring consistency of UML model is modifying of IS model according to detected consistency conflicts.

If method *close* of class Door is added to transition of door protocol states model then consistency of lift system model would be improved.

The experiment shows that the suggested approach is able to detect inconsistencies automatically in such way makes easer work of designer. Beside detected and fixed consistency conflicts in earlier phases of IS life cycle allows to reduce cost of IS development.

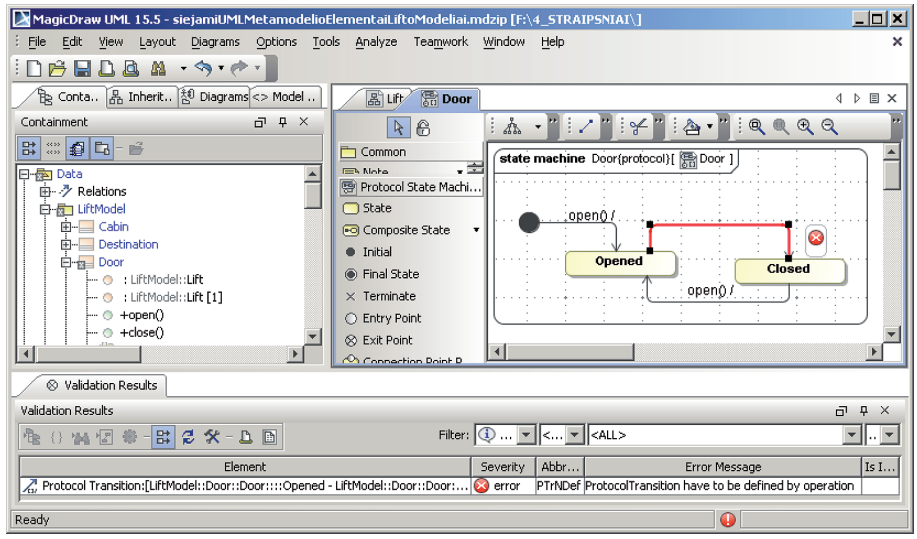

Fig. 8. Checking of IS model using software prototype which implements our approach in ensuring consistency

5. Conclusions and Future Work

The analysis of related work showed that the majority of the existing approaches use constraints for one aspect model. Second biggest group of approaches suggest the using of the formal languages, nonetheless the fact that formal techniques are not yet very popular in the industrial software development community. Formal models are often not enough understandable for software engineers. Semi-formal UML models are widely used, but their constraints usually are for only one model, integrity requirements and constraint for relationships of different aspect models usually are not defined. The approach proposed by Egyed (Egyed 2007) uses some constraints on relationships of different aspect models. But the defined consistency rules are hard coded to the CASE tool UML Analyser and are not formally expressed using language, which would be platform independent.

Despite of many results that have been achieved by different groups of researchers in the field, the issue of ensuring models consistency is still open and relevant.

Based on the researched performer we suggest approach of ensuring UML model consistency, which consist of four elements: (1) checking consistency of original semi-formal UML model, (2) includes constraints on relationships of different aspect models, (3) these consistency rules are expressed explicitly in OCL and (4) defined on metamodel of UML. Checking consistency of original UML model allows not performing additional task of UML model transformation to formal model, in such a way time for detecting consistency conflicts is shortened. UML is more understandable for IS engineers and more usable in practice than formal modelling languages. The suggested approach allows detecting consistency conflicts of different aspect models according to predefined constraints on relationships. Consistency rules expressed in OCL and defined on metamodel are unambiguous, more general and more applicable in various tools.

Our proposed approach was tested using some different consistency rules of static structure and behaviour aspect models and implementing them in consistency control module which was developed for MagicDraw UML tool (No Magic 2010). The experiment showed that usage of the suggested approach allows detecting inconsistencies of different aspects models.

The following step of research is expanding the approach by extending the set of the consistency rules. Next, we intend to create a method for fixing detected inconsistencies of UML model.

References

Bajec, M.; Krisper, M. 2005. A methodology and tool support for managing business rules in organisations, *Information Systems* 30(6): 423–443. doi:10.1016/j.is.2004.05.003

Bajec, M.; Vavpotič, D. 2008. A framework and tool-support for reengineering software development methods, *Informatica* 19(3): 321–344.

Bajec, M.; Krisper, M.; Rupnik, R. 2003. Tracking Business Rule Evolution to Support Is Maintenance, in *ICEIS 2003*, 527–530.

Berkenkötter, K. 2008. Reliable UML Models and Profiles, *Electronic Notes in Theoretical Computer Science (ENTCS)* 217: 203–220. doi:10.1016/j.entcs.2008.06.050

Chen, Z.; Motet, G. 2009. A Language-Theoretic View on Guidelines and Consistency Rules of UML. Model-Driven Architecture Foundations and Applications, *LNCS 5562*: 66–81. doi:10.1007/978-3-642-02674-4_6

Chiorean, D.; Pasca, M.; Carcu, A.; Botiza, C.; Moldovan, S. 2004. Ensuring UML models consistency using the OCL Environment, *Electronic Notes in Theoretical Computer Science (ENTCS)* 102: 99–100. doi:10.1016/j.entcs.2003.09.005

Dubauskaitė, R.; Vasilecas, O. 2009a. UML taisyklių modelių darnos tikrinimo galimybės, naudojant MagicDraw UML ir PowerDesigner įrankius [Checking Consistency of UML Models Using MagicDraw UML and PowerDesigner Tools], in Adomėnas, G. P.; Čeikienė, N.; Kulvietis, G. *et al.* (Eds.). *Informatika*: *12-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis" pranešimų rinkinys* [Informatics: Proc. of the 12th conference of Lithuanian's researchers]. Vilnius: Technika (in press).

Dubauskaite, R.; Vasilecas, O. 2009b. Ensuring Models Consistency in the OMT, Booch, and OOSE Object-Oriented Methods, *Information Sciences* 50: 160–167.

Dzemydienė, D.; Dzindzalieta, R. 2010. Development of architecture of embedded decision support systems for risk evaluation of transportation of dangerous goods, *Technological and Economic Development of Economy* 16(4): 654–671. doi:10.3846/tede.2010.40

Egyed, A. 2007. Fixing inconsistencies in UML design models, in *Proc. of the 29th International Conference on Software Engineering (ICSE)*, 292–301. doi:10.1109/ICSE.2007.38

Gudas, S. 2009. Enterprise Knowledge Modelling Domains and Aspects, *Technological and Economic Development of Economy* 14(2): 281–293. doi:10.3846/1392-8619.2009.15.281-293

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

Jakimavičius, M.; Burinskienė, M. 2009. A GIS and multi-criteria-based analysis and ranking of transportation zones of Vilnius city, *Technological and Economic Development of Economy* 15(1): 39–48. doi:10.3846/1392-8619.2009.15.39-48

Kalibatienė, D.; Vasilecas, O. 2010. Ontology axioms for the implementation of business rules, *Technological and Economic Development of Economy* (16)3: 471–486. doi:10.3846/tede.2010.29

Lucas, F. J.; Molina, F.; Toval, A. 2009. A systematic review of UML model consistency management, *Information and Software Technology* 51: 1631–1645. doi:10.1016/j.infsof.2009.04.009

Mokhati, F.; Gagnon, P.; Badri, M. 2007. Verifying UML Diagrams with Model Checking: A Rewriting Logic Based Approach, in *Proc. of the Seventh International Conference on Quality Software*, 356–362. doi:10.1109/QSIC.2007.4385520

Nemuraitė, L.; Čeponienė, L.; Vedrickas, G. 2008. Representation of business rules in UML&OCL models for developing information systems, *Lecture Notes in Business Information Systems* 15: 182–196. doi:10.1007/978-3-540-89218-2_14

No Magic. 2010. Magic Draw tool [cited 16 February 2010]. Available from Internet: <http://www.magicdraw.com/>.

OMG. 2006. OCL 2.0 Specification [cited 29 October 2009]. Available from Internet: <http://www.omg.org/spec/OCL/2.0/PDF>.

OMG. 2007. OMG Unified Modelling Language (OMG UML), Superstructure, v2.1.2, *OMG Document*: *formal/2007-11-04* [cited 1 April 2009]. Available from Internet: <http://www.omg.org/docs/formal/07-11-02.pdf>.

Pakalnickiene, E.; Nemuraite, L. 2007. Checking of conceptual models with integrity constraints, *Information Technology and Control* 36(3): 285–294.

Rozanski, N.; Woods, E. 2005. *Software System Architecture*. London: Addison-Wesley. 546 p.

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 (COMPSAC 2002), Prolonging Software Life: Development and Redevelopment, Oxford, England, IEEE Computer Society*, 147–152.

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, M. C. 2005. *Description Logics for Consistency Checking of Architectural Features in UML 2.0 Models*. DCC Technical Report TR/DCC-2005-1, Departamento de Ciencias de la Computacion, Santiago, Chile.

Smaizys, A.; Vasilecas, O. 2009. Business Rules Based Agile ERP Systems Development, *Informatica* 20(3): 439–460.

Usman, M.; Nadeem, A.; Kim, T.; Cho, E. 2008. A Survey of Consistency Checking Techniques for UML model, in *Proc. of 2008 Advanced Software Engineering and Its Applications*, 57–62. doi:10.1109/ASEA.2008.40

Van Der Straeten, R.; Simmonds, J.; Mens, T.; Jonckers, V. 2003. Using Description Logic to Maintain Consistency between UML Models, in P. Stevens *et al.* (Eds.). "*UML" 2003 – the Unified Modeling Language. Modeling Languages and Applications. 6-th International Conference*. San Francisco, CA, USA, October 2003. Proceedings *2863*: 326–340.

Vasilecas, O.; Dubauskaite, R. 2009. Ensuring Consistency of Information System Rules Models, in Stoilov, T.; Rachev, B. (Eds.). *Proc. of the International Conference on Computer Systems and Technologies "CompSysTech'09"*, VI.6.1–VI.6.8.

Vasilecas, O.; Lebedys, E. 2007. Application of Business Rules for Data Validation, *Information Technology and Control* 36(3): 273–277.

Vavpotič, D.; Bajec, M. 2009. An approach for concurrent evaluation of technical and social aspects of software development methodologies, *Information and Software Technology* 51(2): 528–545. doi:10.1016/j.infsof.2008.06.001

VeTIS. 2009. *Business Rules Solutions for Information Systems Development.* Project reg. No. B-07042, Lithuanian State Science and Studies Foundation.

VERSLO MODELIO, IŠREIKŠTO UML KALBA, DARNOS TIKRINIMAS

O. Vasilecas, R. Dubauskaitė, R. Rupnik

Santrauka. Modeliuojant verslo sistemą įvairiais aspektais dažnai naudojama UML kalba. Verslo sistemos UML modelis yra sudarytas iš keleto skirtingų aspektų modelių, kurių naudojimas informacinei sistemai projektuoti yra susijęs su darnos pažeidimų problema. Ją lemia dviprasmiška ar netgi prieštaringa informacija, pateikta skirtingų aspektų modeliuose. Straipsnyje pristatomas UML modelio darnos užtikrinimo metodas. Pateikiama keletas darnos užtikrinimo taisyklių, siekiant iliustruoti pasiūlyto metodo veikimą. Sukurtas siūlomą darnos užtikrinimo metodą įgyvendinantis programinės įrangos prototipas taikomas langų ir durų gamybos įmonei modeliuoti. Straipsnyje pristatomi gauti testavimo rezultatai.

Reikšminiai žodžiai: darna, integruotumas, UML modelis, UML diagrama, aspektinis modelis, taisyklės, darnos tikrinimas, darnos užtikrinimas, modelio validavimas.

Olegas VASILECAS. Prof., Dr Habil O. Vasilecas is full time Professor at the Information System Department, and senior researcher, and head of Information Systems Research Laboratory in Vilnius Gediminas Technical University. He is author of more than 200 research papers and 5 books in the field of databases and information systems development. His research interests: knowledge, including business rule and ontology, based information systems development. He delivered lectures in 7 European universities including London, Barcelona, Athens and Ljubljana. O. Vasilecas carried out an apprenticeship in Germany,

Holland, China, and last time in Latvia and Slovenia universities. He supervised 6 successfully defended doctoral theses and now is supervising 7 doctoral students. He was leader of many international and local projects. Last time he leaded VGTU part of "Business Rules Solutions for Information Systems Development (VeTIS)" project carried out under High Technology Development Program.

Rūta DUBAUSKAITĖ. Doctoral student and full time 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.

Rok RUPNIK. Dr. assistant professor of Information Systems at University of Ljubljana, Faculty of Computer and Information Science. He received his Master in Information Systems Engineering from University of Ljubljana, Slovenia, in 1998, and his Ph.D. in Mobile applications from University of Ljubljana, Slovenia, in 2002. His research interests include IT governance, Information Systems Development Methodologies, Mobile Applications, Data Mining, and project management. His articles have appeared in proceedings of many international conferences and journals. He had an important role in various information systems development and other projects. He has strong practical and teaching interests in project management, Information Systems Strategic Planning, developing Data Mining applications and other areas. He is a member of AIS (Association of Information Systems), senior member of PMI (Project Management Institute) Slovenian Chapter, member of ACM and a member of IEEE. In 2009 he received PMP (Project Management Professional) certificate by PMI (Project Management Institute).