The Checking of Information Systems Models Using Rules[7]

Ruta Dubauskaite, Olegas Vasilecas, Algis Saulis

Abstract: *Different models are developed when information system (IS) is designed by data, process and other aspects. The representation of information system through various models is related to the problem of ensuring consistency among different models. The problem of models inconsistency can be solved by using of formal or partially formal models with constraints. However formal models are often too complex to be used in practice. Semi-formal models are widely used, but constraints used in such a models often are suitable only for one model and relationships among models are not defined. Hence the author of paper suggests extending of IS approach based on semi-formal models and constraints, by adding the consistency rules for IS models. The proposed approach is illustrated by a case study.*

Key words: *information systems models, consistency, checking models, ensuring consistency.*

INTRODUCTION

The models of processes, states, structure and other models are created when modelling information system by various aspects. Sometimes the models of different aspects are not related. Even more, contradictory information can be provided in them. The expressing of information system through various models is related to the problem of ensuring consistency among different 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 [1].

Model-driven architecture (MDA) puts models into the centre of the information system development process as the source of transformation to platform-specific models (PSM). PSM is used for code generation [2]. Unambiguous models are necessary for the successful accomplishment of the tasks of models transformation and code generation [3]. Therefore checking consistency among related models is very important. Hence the objective of paper is to improve consistency of information systems models.

The remainder of the paper is organized as follows. Section "Related Work" gives a brief overview on approaches of ensuring information systems models consistency. Section "Checking of Information Systems Models Using Rules" presents the suggested approach of ensuring IS models consistency. Section "A Case Study" illustrates the usage of suggested approach for ensuring consistency. Section "Conclusions and Future Works" concludes the paper.

RELATED WORK

The problem of ensuring models consistency can be solved: i) using matrixes; ii) modelling system using semi-formal language with constraints; iii) modelling system using formal language.

Matrix can be used for showing consistency rules among different aspect semi-models [4]. For example the usage of entity-function matrix helps to check the consistency of structure and behaviour models. But this approach does not take into consideration constraints of every aspect model. Consistency of information systems models means correctness both of one model and among all models of information system.

Information system models can be created using natural language, semi-formal or formal modelling language. UML (Unified Modelling Language) is the most popular semi formal modelling language [3]. But the usage of UML can not ensure that created

[7] The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)" Reg. No. B-07042

models are consistent, because of lack of formal semantics [2]. Part of semantics is defined by OCL (Object Constraint Language) constraints, most of them are provided in UML specification by natural language that is claimed to be precise. Berkenkotter [3] rectifies UML constrains expressed in OCL and formalizing UML constrains express in natural language. The main disadvantage of these works [3, 5] is that constrains only for one model are defined, but consistency rules among different models are not present.

Another suggested approach of ensuring models consistency is based on formal models [2, 6, 7]. Formal models are expressed by formal modelling language, for example Maude (based on rewriting logic) [2] description logic [7]. Most formal languages have inference mechanisms. These inference mechanisms allow to reason about the consistencies of knowledge bases. The approach based on formal models considers both static and dynamic features of object-oriented systems. The result of models checking is formal text. It is difficult enough to understand formal text even for developers of information systems.

More information about approaches of ensuring models consistency is provided in our paper [8].

CHECKING OF INFORMATION SYSTEMS MODELS USING RULES

The analysis of ensuring consistency approaches shows the inconsistency problem can be solved by using of formal or semi-formal models with constraints. Formal models are often too complex to be used in practice. Semi-formal models are widely used, but their constraints often are applied only for one model and relationships among models are not defined. Semi-formal languages are more understandable in comparison with formal languages. Semi-formal languages are also more precise in comparison with natural languages. Therefore it is relevant to improve consistency of semi-formal models.

Based on the research performed the authors of paper suggest to define constraints for aspect model in explicit way and define consistency rules among aspect models in explicit way. The approach of ensuring models consistency in object-oriented models is presented in fig. 1.

The usage of defined constraints and consistency rules allows detecting models consistency conflicts. If constraints are expressed in explicit way then it is possible to automate the process of checking consistency of developed models. When conflicts are detected then they can be removed in such way improving consistency of models. The usage of suggested approach and defined constraints and consistency rules are illustrated by a case study in the following section of the paper.

A CASE STUDY

According to suggested approach it is recommended to define constraints and consistency rules in explicit way. The example of UML model consistency rule and using of it for detecting inconsistencies in models of books library information system are presented in the chapter. Due to space limitations only one consistency rule is presented.

The authors of this paper define consistency rule between UML class and sequence models. Classes and their relations show static structure of data, while sequence model presents interactions of classes. Consistency rules are defined for metamodel. According to UML superstructure specification [9] the lifeline of sequence model can be related with class of class model through ConnectableElement ("The classifier containing the referenced ConnectableElement must be the same classifier, or an ancestor, of the classifier that contains the interaction enclosing this lifeline.").

Let's consider we are analysing books library domain and modelling:

- The process of books search using sequence model.

- The static structure of books using class model.

We remind the main elements of class model are classes and relations among them.

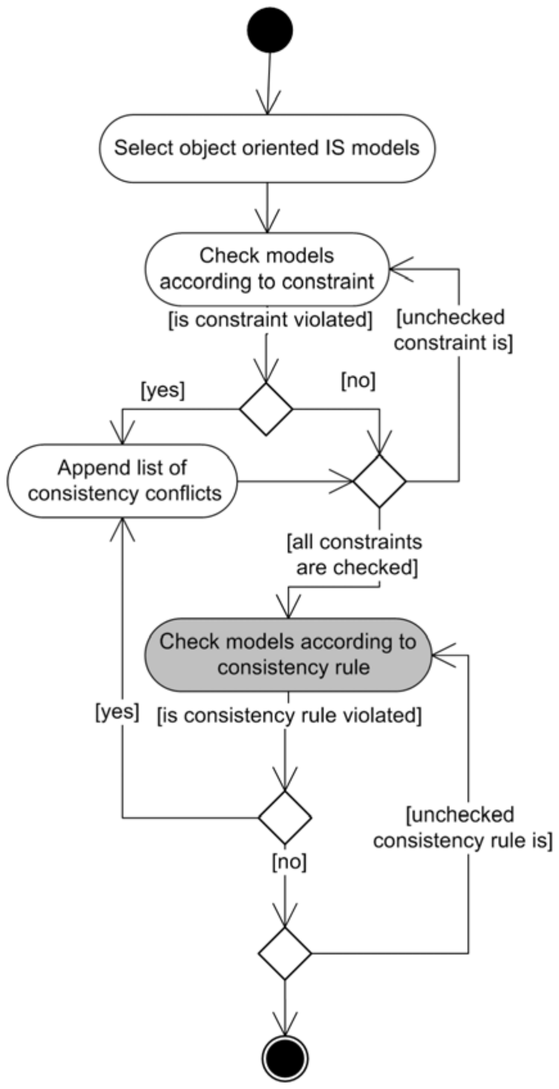
Select object oriented IS models
Check models according to constraint
[is constraint violated]
[unchecked constraint is]
[yes]
[no]
Append list of consistency conflicts
[all constraints are checked]
Check models according to consistency rule
[yes]
[is consistency rule violated]
[unchecked consistency rule is]
[no]

Fig. 1 Approach of checking consistency of IS models

Class consist of three parts: name, attributes and operations. Class defines type of object data. The main elements of sequence model are lifelines of objects and messages, sent and received by lifelines in order to perform process.

The authors of the paper suggest defining consistency rule between operations of class and messages of lifeline in order to test the suggested approach. Consistency rule consists of a textual description and a formal constraint expressed in OCL.

The operation of classifier containing the referenced Message of Connectable element must be the same operation of the classifier that contains the interaction enclosing this message of lifeline.

```
    context ce : UML::CompositeStructures::InternalStructures::
ConnectableElement inv OperationOfClassMustBeMessageOfLifeLine :
    ce.class.operation=ce.lifeline.interaction.message
```

Using consistency rule OperationOfClassMustBeMessageOfLifeLine helps to detect consistency conflict between models presented in fig 2. Static structure, classes and their relations can be presented graphically using UML class diagram. While interactions can be showed using UML sequence diagram. Lifeline Book (fig. 2, right column) is related with class Book (fig. 2, left column). Lifeline Book has message getBookDetails. While class Book, which is equivalent of lifeline Book, has no operation. Operation getBookDetails is also not present is this class. It means that static structure and behaviour models are inconsistent.

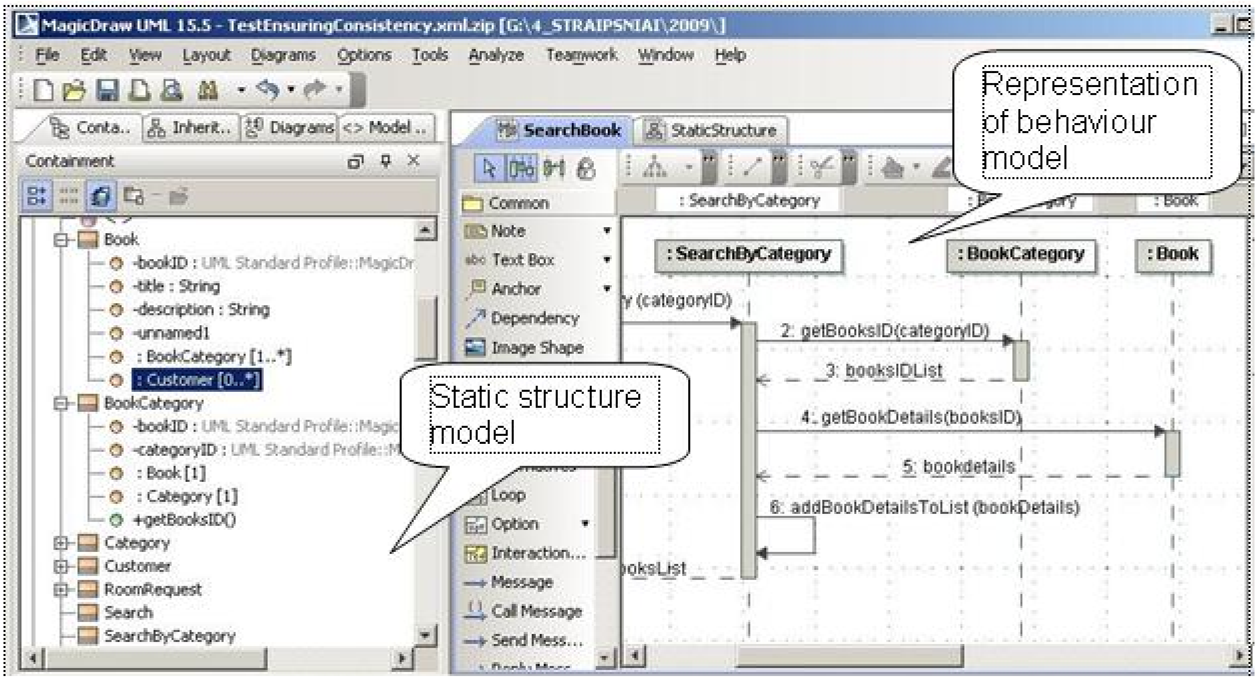
MagicDraw UML 15.5 - TestEnsuringConsistency.xml.zip [G:\4_STRAIPSNIAI\2009\]
Representation of behaviour model
Static structure model
Book
-bookID : UML Standard Profile::MagicDr
-title : String
-description : String
-unnamed1
: BookCategory [1..*]
: Customer [0..*]
BookCategory
-bookID : UML Standard Profile::Magic
-categoryID : UML Standard Profile::M
: Book [1]
: Category [1]
+getBooksID()
Category
Customer
RoomRequest
Search
SearchByCategory
SearchBook
StaticStructure
: SearchByCategory
: BookCategory
: Book
2: getBooksID(categoryID)
3: booksIDList
4: getBookDetails(booksID)
5: bookdetails
6: addBookDetailsToList (bookDetails)

Fig. 2 The example of models and their consistency conflicts

If designer add operation getBookDetails to class Book, then consistency between static structure and behaviour models would be improved.

The process of checking models consistency rules can be automated using CASE tools. Because of space limitations, CASE tools are not detailed here. Checking consistency of UML models using MagicDraw UML and PowerDesigner tools is presented in our paper [12]. The authors of the paper implement consistency rules using MagicDraw UML tool [8].

CONCLUSIONS AND FUTURE WORK

The analysis of related works shows semi-formal models are widely used, but their constraints often are applied only for one aspect and relations among models are not defined. Despite many results that have been achieved on models consistency, the problem of ensuring models consistency is still open and relevant.

Based on the research performed the authors of paper suggest extending of object-oriented methods by defining constraints for aspect model in explicit way and consistency rules among aspect models in explicit way.

A case study shows that the using of suggested approach allows detecting of inconsistencies of different aspect models.

In future works we are going to elicit consistency rules from different object oriented methods and from metamodels of modelling languages expressed in explicit and implicit way. The next step is implementing of defined rules in MagicDraw UML tool and testing the suggested approach with defined consistency rules.

REFERENCES

[1] Rozanski, N.; Woods, E. (2005). Software System Architecture. London: Addison-Wesley. 546 p.

[2] Mokhati, F.; Gagnon, P.; Badri, M. (2007). Verifying UML Diagrams With Model Checking: A Rewriting Logic Based Approach. Proceedings of the Seventh International Conference on Quality Software. USA,p. 356–362.

[3] Berkenkötter, K. (2008). Reliable UML Models and Profiles. Electronic Notes in Theoretical Computer Science (ENTCS), vol. 217, p. 203–220.

[4] Saulis, A.; Vasilecas, O. (2008). Informacinių sistemų projektavimo metodai. Vilnius: Tehcnika. 247 p.

[5] Pakalnickienė, E., Nemuraitė, L. (2007). Checking of conceptual models with integrity constraints. Information technology and control, T. 36, no. 3, p. 285–294.

[6] Van Der Straeten, R.; Simmonds, J.; Mens, T.; Jonckers, V. (2003). Using Description Logic to Maintain Consistency between UML Models. UML 2003: LNCS 2863. Berlin: Springer-Verlag. 326–340.

[7] 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.

[8] Vasilecas, O., Dubauskaite, R. (2009). Ensuring Consistency of Information System Rules Models. Stoilov, T., Rachev, B. (eds.), Proc. of the International Conference on Computer Systems and Technologies "CompSysTech'09", Ruse, Bulgaria, 18-19 June (in press).

[9] OMG. (2007). OMG Unified Modeling Language (OMG UML), Superstructure, v2.1.2 , OMG Document: formal/2007-11-04. Available at <http://www.omg.org /docs/formal/07-11-02.pdf>, last visit 2009 04 01.

[10] Booch, G.; Rumbaugh, J.; Jacobson, I. (1998). The Unified Modeling Language User Guide. Addison Wesley 512 p.

[11] Chonoles, M., J.; Quatrani, T.. Lockheed M. Advanced Concepts Center, Rational Software Corporation. (1996). Succeeding with the Booch and OMT methods: a practical approach, Addison-Wesley, 378 p.

[12] Dubauskaitė, R.; Vasilecas, O. (2009). Checking Consistency of UML Models Using MagicDraw UML and PowerDesigner Tools, Informatics: proc. of 12th conference of Lithuanian‘s researchers, Vilnius: Technika (in press).

ABOUT THE AUTHORS

Ruta Dubauskaite, Doctoral student, Department of Information Systems, Vilnius Gediminas Technical University, Phone: +37052744860, E-mail: rutad@isl.vgtu.lt.

Prof. Olegas Vasilecas, Dr., Department of Information Systems, Vilnius Gediminas Technical University, Phone: +37052744860, E-mail: olegas@fm.vgtu.lt.

Assoc. Prof. Algis Saulis, Dr., Department of Information Systems, Vilnius Gediminas Technical University, Phone: +37052370620, E-mail: asaulis@fm.vgtu.lt.