# PROGRAMAVIMAS IR PROGRAMINĖ ĮRANGA

## Ensuring Models Consistency in the OMT, Booch, and OOSE Object-Oriented Methods*

**Rūta Dubauskaitė**

Vilnius Gediminas Technical University
Programmer, Doctoral student
Saulėtekio al. 11, LT-10223 Vilnius, Lithuania
E-mail: rutad@isl.vgtu.lt

**Olegas Vasilecas**

Vilnius Gediminas Technical University
Prof., Dr.
Saulėtekio al. 11, LT-10223 Vilnius, Lithuania
E-mail: olegas@isl.vgtu.lt

*Modelling of information systems (IS) involves development of different models that present various aspects of a system, like the static structure, behaviour, etc. Expression of an IS through various models is related to the problem of ensuring of different models consistency, which is very important for IS design, models transformation and finally code generation tasks.*
*Restriction of IS models can help to solve the problem of models inconsistency. However, constraints defined in the OMT, OOSE, Booch object-oriented methods are suitable only for one model and at that point constraints of models relationships are not defined explicitly. Hence the authors of the paper suggest the approach for extension of an ensuring consistency, based on semi-formal models with constraints, by adding the consistency rules to object-oriented IS models. Consistency rules are directed to constrain relationships of different aspect models. The proposed approach is illustrated by a case study in a digital library domain.*

### Introduction

The models of processes, states, structure and other models are created when an information systems (IS) are modelling by various aspects. Zahman Enterprise architecture framework suggested modelling aspects are data (things), function (process), network (location), people, time and motivation (business rules) (McGovern, 2003).

Business rules make an important and integral part of each IS by expressing business logic, constraints on concepts, their interpretation and relationships. Defined rules of structure and behaviour are captured in the models of structures, states, processes and other IS models (Nemuraite, Ceponiene, Vedrickas, 2008). Hence the problem of IS models consistency includes the problem of rules models consistency or inconsistency.

Sometimes the models of different aspects are not interrelated and even more, contradictory information can be provided in them. Expression of an IS through various models is related to the problem of ensuring consistency of 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 (Rozanski, Woods, 2005). According to (ISO/IEC 1997), consis-

* 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

tency expresses the matching ratio. Sometimes an integrity concept is used instead of consistency concept. According to Caplinskas (1997), consistency is part of integrity. Integrity also involves the following quality characteristics: accuracy of facts, accuracy of values, conformity, accuracy of time. In this research, we are concentrated on improving models consistency.

Model-driven architecture (MDA) puts models into the centre of the IS development process as the source of transformation to platform-specific models (PSM), which are used for code generation (Mokati et al. 2007). Unambiguous models are necessary for the successful accomplishment of the tasks of model transformation and code generation (Berkenkotter, 2008; Rozanski, Woods, 2005). Therefore checking the consistency of related models should be central aim of IS development process.

Thus the goal of the paper is to improve the consistency of IS models. The main task is to extend the existing approach for checking models consistency and removal of detected inconsistencies that would allow making a model more consistent. The proposed approach is illustrated by the case study of consistency rules for UML static structure and behaviour models in a digital library domain. In the future the authors are going to apply and extend the proposed approach to a method for checking the consistency of UML static structure and behaviours models, especially concentrating on the consistency of models elements that can be visualised by class and sequence diagrams.

The related work analysis is made by theoretical research and comparative analysis methods. Logical induction and constructive researched methods are used to suggest a solution of ensuring the consistency problem. The suitability of proposed approach for solving the consistency problem is researched by the experimental analysis method.

The remainder of the paper is organized as follows. Section "Related Work" gives a brief overview on approaches for ensuring the consistency of IS models and presents the analysis results of ensuring the consistency in Booch, OMT, OOSE object oriented methods. Section "The Approach of Ensuring the Consistency in Object-Oriented Models" presents the proposed approach for checking the consistency of object-oriented IS models. Section "A Case Study" illustrates the usage of approach proposed for ensuring consistency. Section "Conclusions and Future Works" concludes the paper and presents future research directions.

## Related Work

### *Approaches of Ensuring Consistency*

The problem of ensuring models consistency, for example, can be solved: i) using matrices; ii) modelling a system using a semi-formal language with constraints or iii) modelling a system using a formal language.

A matrix can be used to show the consistency rules among semi-formal models of different aspects (Saulis, Vasilecas, 2008). For example, the usage of the entity-function matrix helps to check the consistency of structure and behaviour models. However this approach does not take into consideration constraints of every aspect model taking into account that the consistency of IS models means not only correctness of one single model, but all related models of the IS used in development process.

IS models can be created using the natural language, semi-formal or formal modelling language. Semi-formal languages, for example, UML, are useful for modelling IS, because these languages are more understandable in comparison with formal languages and more precise in comparison with the formal languages and more precise in comparison with natural languages. But the usage of a semi-formal language cannot ensure that created models were consistent, because the of lack of formal semantics (Mokati et al., 2007). Part of semantics of UML is defined by OCL (Object Constraint Language), and some of constrains are provided in UML specification by the natural language that is claimed to be precise. Berkenkotter (2008) suggested to rectify UML constraints expressed in OCL and formalize UML constrains expressed in the natural language. The major disadvantage of these works (Berkenkotter, 2008; Pakalnickiene, Nemuraite, 2007) is that

they propose constraining only one model, but the consistency rules of different aspects models are not presented. The consistency rules in this context mean constraints on relationships between models. The relationships of models show which elements of a model should have relationships with the elements of another model. These relationships are often not expressed explicitly. For example, in the UML state chart diagrams it should be allowed to model states of a class, which is defined in the model of static structure (classes diagram).

Another approach proposed for ensuring models consistency is based on formal models (Mokati et al., 2007; Van Der Straeten et al., 2003; Simons, Bastarrica, 2005). Formal models are expressed by a formal modelling language, for example, Maude (based on rewriting logic) (Mokati et al., 2007) and a description logic (Van Der Straeten et al. 2003). Most formal languages have inference mechanisms. These mechanisms allow us to reason about the consistencies of knowledge bases. The result of models checking is a formal text. It is difficult enough to understand the formal text even for IS developers. More information about approaches for ensuring models consistency is provided in (Vasilecas, Dubauskaite, 2009).

The analysis of the approaches for ensuring consistency shows that the approaches proposed solve the models inconsistency problem from some specific viewpoint. Formal models are often too complex to be used in practice. Semi-formal models are wide used, but their constraints usually are only applied to one model and the constraints on relationships of models are not defined (Table 1).

Therefore the authors suggest extending the approach based on semi-formal models with constraints by adding the consistency rules for relationships of different aspect models.

### *The Analysis of Ensuring Consistency in the Booch, OMT, OOSE Object-Oriented Methods*

UML is the most popular semi-formal modelling language (Berkenkotter, 2008). It is considered as the standard for the object-oriented modelling (Mokati et al. 2007) and usually object-oriented methods are used for a detailed analysis and following implementation. The initial versions of UML originated from three leading object-oriented methods: Booch, OMT, and OOSE (UML 2007). Therefore these methods are chosen for a detailed analysis. There is a need to mention that there are a lot of object-oriented system development methods, for example, OAA (Object Oriented Analysis), UP (Unified Process), RUP (Rational Unified Process), ICONIX, AM (Agile Modelling) methods, Merode development process, Newton, etc. But this time they are not in the scope of our research.

The original developers of the UML Grady Booch, James Rumbaugh, and Ivar Jacobson provide the core of the language in the Booch (Booch 1991), OMT (Rumbaugh 1991) and OOSE (Jacobson 1992) methods.

The Booch, OMT, and OOSE methods describe, which models should be created for software development (Booch, Rumbaugh, Jacobson 1998). For example, the Booch model consists of three different aspect models: requirements model (describes all objects statically), behaviour model, and architecture model (Conoles et al., 1996).

The analysis of Booch, OMT and OOSE shows that instructions how to develop different aspect models (for example, static or dynamic) are provided in the natural language and expressed in an implicit way. Examples of such instructions for developing a dynamic model of OMT are:

- A process must have at least an input flow and an external flow.
- A flow is between processes or between the file and process, or process and external input, external output (Conoles et al., 1996).

The Booch, OMT, and OOSE methods express instructions for model development in an implicit way, therefore it is difficult to check the consistency and to automate process of checking consistency of the developed models.

The OMT, Booch and OOSE methods define general models relations (flow of models development), but the relationships of elements in different aspect models are not described.

A detailed comparison of approaches for ensuring models consistency and ensuring

*Table 1.* **Comparison on approaches for ensuring models consistency using object-oriented methods**

| Comparison criteria / Compared approaches and methods | | Ensuring consistency of one model | | Ensuring consistency of different aspect models | |
|---|---|---|---|---|---|
| | | Instructions for model development, expressed in an implicit way | Constraints for one model, expressed in an explicit way | General relationships of models (flow of models development) | Relationships of elements of different aspect models expressed in an explicit way |
| Approaches of ensuring consistency of semi-formal models | Using matrices | - | - | - | + |
| | Modelling system using semi-formal language with constraints | - | + | - | - |
| | Transforming semi-formal models to formal models | - | + | | + |
| Object-oriented methods for creating semi-formal UML models | OMT | + | - | + | - |
| | Booch | + | - | + | - |
| | OOSE | + | - | + | - |
| The approach of ensuring semi-formal model consistency proposed in this work | | - | + | - | + |

object-oriented models consistency using OMT, OOSE and Booch methods are presented in Table 1.

Based on the analysis of approaches for ensuring consistency using object-oriented models (OOM), the authors suggest:

- extending the approach based on semi-formal models with constraints, adding the consistency rules on different aspect models relationships (see Table 1, text in a grey shadow);
- to use several consistency rules that can be acquired from object-oriented methods and formalised (see Table 1, text in vertical lines shadow).

The proposed approach is presented in detail in the next section.

## The approach for Ensuring the Consistency in Object-Oriented Models

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

The analysis of Booch, OMT and OOSE methods shows that they express instructions for model development in an implicit way and define only general models relationships. However these OOM can be a source for acquisition of formal constraints and consistency rules.

The authors of the paper suggest extending the IS development approach, based on semi-formal models and constraints, by adding the formal consistency rules to IS models. The assumption is that set of semi-formal models are OOM. The proposed approach for ensuring consistency of OOM is presented in Fig. 1.

Fig. 1 presents sequence of steps that realised proposed approach. First of all a method for checking the consistency of IS models should be created. Next the method for ensuring the consistency of UML models will be created and farther constraints and consistency rules will be defined for a metamodel of modelling language. The novelty of this approach is proposal to define the consistency rules for a relationship of two elements from different aspect models in an explicit way (Fig. 1, activities in gray shadow). The analysis of related works shows that nowadays constraints on the elements of only one aspect model are defined in an explicit way.

A second group of activities of the approach proposed is checking IS models. If constraints are expressed in an explicit way, it is possible to automate the process of checking consistency of the developed models. All models are checked according to the implemented constraints and consistency rules.

And finally, the consistency of IS models is improved by removing the detected conflicts. The usage of the approach proposed is illustrated by a case study in the next section of the paper.

## A Case Study

According to the approach proposed, it is recommended to define constraints and consistency rules in an explicit way. An example of the UML model consistency rule and its usage for detecting inconsistencies in the books library IS models are presented in the chapter. Due to space limitations only one consistency rule is presented.

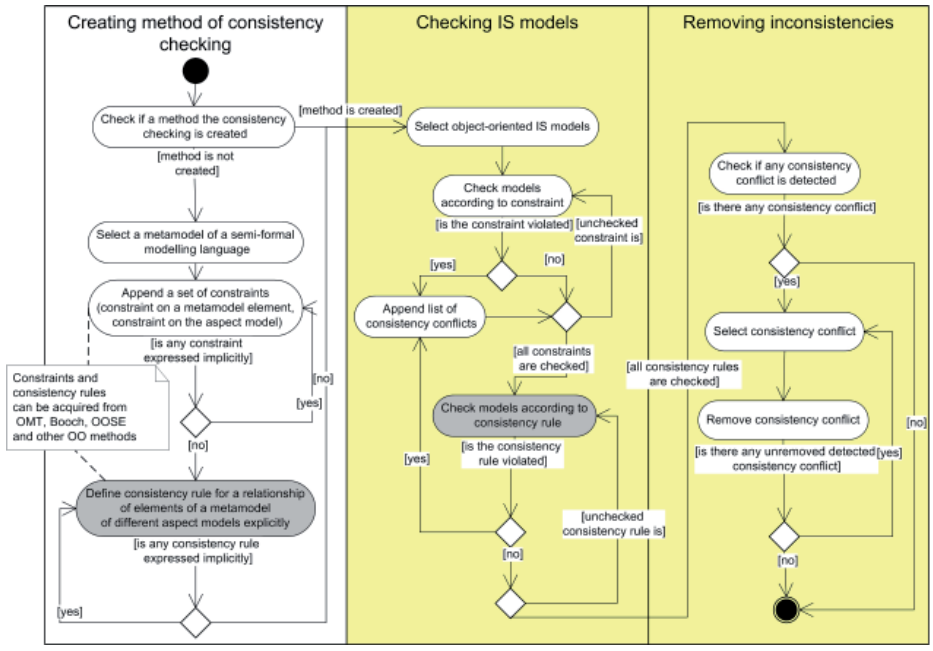



*Fig. 1* ***Approach for ensuring model consistency in object-oriented approach***

In this section, a consistency rule for UML class and sequence models is presented. Classes and their relationships show a static structure of data, while a sequence model presents interactions of classes. The consistency rules are defined for a metamodel.

Let us consider that we analyse a domain of books library and develop models:

The process of books search is presented in the sequence model.

The static structure of books library is presented in the class model.

Recall that the major elements of the class model are classes and their relations. A class consists of three parts: name, attributes, and operations. The class defines the type of object data. The major elements of a sequence model are lifelines of objects and messages, sent and received by lifelines in order to perform the process.

According to UML superstructure specification (OMG, 2007), the lifeline of a sequence model can be related with the class of class model through *ConnectableElement.* The relationship description in (OMG, 2007) is provided in the natural language: "The classifier containing the referenced *ConnectableElement* must be the same classifier, or an ancestor of the classifier that contains the interaction enclosing this lifeline."). In this case, OMG UML specification is the source of the proposed consistency rule ID1 (see the text below).

The authors of the paper suggest defining the consistency rule between operations of the class and messages of lifeline in order to test the approach. The consistency rule consists of a textual description and a formal constraint expressed in OCL. Below the consistency rule between elements of different aspect models is presented.

Consistency rule ID1:

*The operation of a 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.*

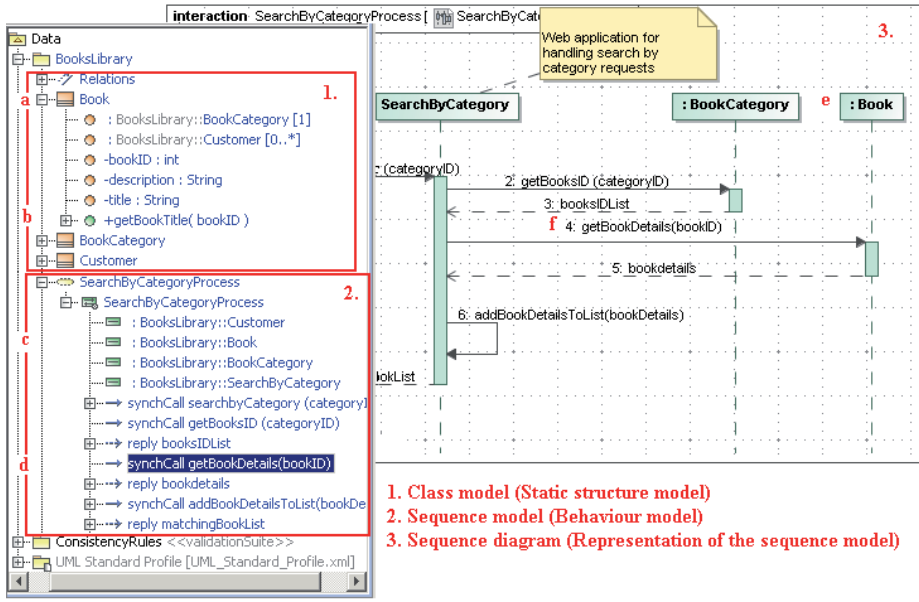
```
context ce : UML::CompositeStructures::InternalStructures::
ConnectableElement inv OperationOfClassMustBeMessageOfLife-
Line :
   ce.class.operation=ce.lifeline.interaction.message
```

It should be noted that not all the lifelines of a sequence model are related with classes in the class model. The type of lifelines can be a domain, control and boundary. An example of the control lifeline is a *Search* lifeline which represents web application, while the domain lifeline *Book* represents class *Book* of the class model. Only the domain lifelines should be connected with the class model. It means that the proposed consistency rule ID1 should be applied only to domain lifelines of the sequence model.

Using the consistency rule *ID1* helps to detect a consistency conflict of models presented in part 1 of Fig 2 and part 2 in Fig. 2. The static structure, classes and their relations can be presented graphically using a UML class diagram, while interactions can be showed using the UML sequence diagram (part 3 of Fig. 2). The lifeline *Book* (part 2.c of Fig. 2) is related with the class *Book* (part 1.a of Fig. 2). The lifeline *Book* (part e of Fig 2) has a message *getBookDetails* (part f of Fig. 2, part 2.d of Fig. 2), while the class *Book*, which should be related with the lifeline *Book*, has only one operation *getBookTitle* (part 1.b of Fig.2), but the operation *getBookDetails* is not presented in this class. It means that the static structure and behaviour models are inconsistent.

If a designer adds the operation *getBookDetails* to the class *Book,* the consistency of static structure and behaviour models will 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 the consistency of UML models by using MagicDraw UML and PowerDesigner tools is presented in (Dubauskaite, Vasilecas, 2009). The authors of the paper intend to extend the proposed approach to the method for consistency checking of UML models and to implement it using the MagicDraw UML tool (Vasilecas, Dubauskaite, 2009).




*F i g . 2* ***The example of models and their consistency conflicts***

## Conclusions and Future Works

The analysis of related works shows that semi-formal models are wide used, but consistency constraints are often applied to only one modelling aspect. Constrains on the relationships of elements of different aspect models usually are not provided in an explicit way. Object-oriented development methods like Booch, OMT and OOSE express the instructions for semi-formal model development in an implicit way but describe only general relationships of models. Despite the fact that some results have been achieved in the field of ensuring of models consistency, the problem of is still open and relevant.

Based on the research performed, the authors of the paper suggest extending of the existing approach, based on semi-formal models by adding the consistency rules for the relationship of elements of different aspect models in an explicit way. Object-oriented methods can be a source for acquisition of formal constraints and consistency rules.

The case study shows that usage of the approach proposed allows detecting inconsistencies of different aspect models.

In future works we are going to elicit the consistency rules expressed explicitly and implicitly from different models used in object-oriented methods and from the UML metamodel, and to extend the proposed approach to a method for ensuring consistency of UML models. The next step is implementing of defined rules in the MagicDraw UML tool which is used in running “Business Rules Solutions for Information Systems Development (VeTIS)” project and testing the method proposed.

### REFERENCES

AYVAZREYS, Z.; UYSAL, M. (2001). Designing an Object-Oriented Application Model by Booch methodology. *Journal of Electrical &Electronics*, vol. 1, no 2. p. 193–200.

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

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. (1998). *The Unified Modelling Language User Guide*. Addison Wesley 512 p.

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)*, vol. 102, p 99–100.

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.

ČAPLINSKAS, A. (1996). Programų sistemų inžinerijos pagrindai [Fundamental of Software System Engineering]. Vilnius: Mokslo aidai, t. 1. 294 p.

DUBAUSKAITĖ, R.; VASILECAS, O. (2009). Checking Consistency of UML Models Using MagicDraw UML and PowerDesigner Tools, Informatics: *Proc. of the 12th conference of Lithuanian's researchers*. Vilnius: Technika (in press).

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

JACOBSON, Y. (1991). *Object-oriented software engineering*. USA: ACM. 400 p.

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.

NEMURAITE L., CEPONIENE L., VEDRICKAS, G. (2008). Representation of business rules in UML&OCL models for developing information systems. In T*he Practice of Enterprice Modeling*: First IFIP WG 8.1 Working Conference, PoEM 2008, Stockholm, Sweden, November 12-13, 2008 : proceedings / Stirna, Janis Persson, Anne (Eds.). Vol. 15, Lecture Notes in Business Information Processing. Berlin; Heidelberg; New York, p. 182–196.

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.

PAKALNICKIENĖ, E., NEMURAITĖ, L. (2007). Checking of conceptual models with integrity constraints. *Information technology and control*, vol. 36, no. 3, p. 285–294.

ROZANSKI, N.; WOODS, E. (2005). *Software System Architecture*. London: Addison-Wesley. 546 p.

SAULIS, A.; VASILECAS, O. (2008). *Informacinių sistemų projektavimo metodai*. Vilnius: Tehcnika. 247 p.

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.

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, p. 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"*, Ruse, Bulgaria, 18–19 June (accepted).

## MODELIŲ DARNOS UŽTIKRINIMAS NAUDOJANT OMT, BOOCH IR OOSE OBJEKTINIUS METODUS

**Rūta Dubauskaitė, Olegas Vasilecas**

S a n t r a u k a

Projektuodami informacinę sistemą sukuriame skirtingus modelius pagal duomenis, procesus ir kitus aspektus. Šiuo atveju yra rizika, kad sistemos specifikacijoje bus darnos pažeidimų dėl tarpusavyje nesuderintų modelių. Darnūs ir neprieštaringi modeliai yra reikalingi tolesniam pradinių modelių transformavimui ir galiausiai programinio kodo generavimui. Taigi yra svarbu užtikrinti modelių darną. Taikant objektinių sistemų kūrimo metodus darnos užtikrinimo problema sprendžiama apribojant kiekvieno aspekto, pavyzdžiui, duomenų, procesų ir kitų, modelius. Tačiau dviejų ir daugiau skirtingų aspektų modelių ribojimai nėra apibrėžti išreikštiniu būdu. Šiame darbe siūloma išplėsti ribojimais grindžiamą modelių darnos užtikrinimo būdą, papildant jį objektinių modelių darnos taisyklėmis. Darnos taisyklė šiame kontekste apibrėžia ryšių tarp skirtingų aspektų modelių elementų ribojimus. Pasiūlytas objektinių modelių darnos užtikrinimo būdas yra iliustruojamas knygų bibliotekos dalykinės srities pavyzdžiu.