A Survey of Consistency Checking Techniques for UML Models

Muhammad Usman[1], Aamer Nadeem[1], Tai-hoon Kim[2], Eun-suk Cho[2]

[1]*Center for Software Dependability, Mohammad Ali Jinnah University*
Islamabad, Pakistan
m_usman99@yahoo.com, anadeem@jinnah.edu.pk

[2]*Dept. of Multimedia, Hannam University,*
133, Ojeong-dong, Daedeok-gu, Daejeon, Korea
taihoonn@hannam.ac.kr, eunsukk@empal.com

Abstract

UML is the de-facto industry standard to design object-oriented software. UML provides a set of diagrams to model every aspect of an object-oriented application design in sufficient detail, but lacks any mechanism to rigorously check consistency between the models. Today, most of the effort is applied on creating accurate and consistent UML models rather than implementing the design. Automatic code generation from UML models has emerged as a promising area in recent years. The accuracy of generated code in some ways depends on UML models consistency. In this paper, we present a survey of UML consistency checking techniques. To analyze existing techniques, we identify some analysis parameters and construct an analysis table. The analysis table helps us to evaluate existing consistency checking techniques. We conclude that most of the approaches validates intra and inter level consistencies between UML models by using monitoring strategy. UML class, sequence, and statechart diagrams are used in most of the existing consistency checking techniques.

1. Introduction

The launch of UML [3] opened a new way for OO application designing [1]. UML standard of Object Management Group (OMG) [4] contains a set of useful diagrams to express static and dynamic properties of an OO application in design phase [2]. OMG has also introduced Model Driven Architecture (MDA) [5] to provide a standard for automatic translation of UML models to OO code.

Automatic translation of UML models to code minimizes the number of errors and generates more conformed and reliable OO code than manual translation. So it is important to have consistent UML models to get conforming OO code. It requires to have full concentration on UML models consistency such as if a UML sequence diagram calls a method on an object of a class then method signature for that method must exists in UML class diagram in that specific class [4]. UML models consistency is also important because inconsistent UML models result into inaccurate OO code generation. Consistency validation between UML models is useful because it is hard to make changes in source code than in UML models. So whenever UML models are changed, it is very important to assure that UML models are still consistent after the changes. Consistency validation between UML models also helps software vendors financially by minimizing the cost during software development process [6].

In this paper, we present a survey of existing consistency checking techniques between UML models. We have defined some analysis parameters and construct an analysis table to analyze existing techniques on the basis of parameters. Most of the techniques focus on inter and intra level consistency validation between UML models. Nearly all existing consistency checking techniques provide consistency rules to validate consistency between UML models which comes under monitoring strategy.

2. Consistency Types

Following are the types of consistencies which we are focused in context of this survey paper from Mens *et al.* [7].

Inter-model (Vertical) Consistency: Consistency is validated at different levels of abstraction between different diagrams. Syntactic and semantic consistencies are also included in it.

Intra-model (Horizontal) Consistency: Consistency is validated at a same level of abstraction between different diagrams.

Evolution Consistency: Consistency is validated between different versions of a same UML diagram.

Semantic Consistency: Consistency is validated for UML diagrams semantic meanings defined by UML metamodel.

Syntactic Consistency: Consistency is validated for UML diagrams specifications in UML metamodel.

978-0-7695-3432-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ASEA.2008.40

3. Analysis Parameters

This section provides a set of analysis parameters used for analyzing consistency checking techniques.

Nature: It identifies the focused properties of an OO application such as static or dynamic. Possible values are structural, behavioral and both which are based on used UML diagrams.
MDA Based: It identifies that the approach lies in MDA domain or not. Possible values are yes and no.
UML Version: It identifies the version of UML. Possible values are the different available versions of UML such 1.0, 1.4, 1.5, 1.1.1, 2.0, 2.1.1, etc.
UML Diagrams: It identifies the UML diagram(s) use in the presented approach. Possible values are any one or more UML diagrams.
Consistency Type: It identifies the relevant consistency type addressed in the presented approach. Possible values are from the list of consistency types discussed in section 2.
Intermediate Representation: It identifies any temporary representation is used to validate consistency between UML diagrams. Possible values any intermediate representation which varies from technique to technique.
Consistency Strategy: It identifies the strategy used to validate consistency between UML diagrams. There are three types of strategies available in literature as *consistency by analysis* (based on an algorithm), *consistency by monitoring* (based on rules), and *consistency by construction* (generates one artifact from another) [14]. Possible values are the strategies defined above.
Rules Provided: It identifies the level of consistency checking rules presented in a paper. Possible values are high, medium and low which are determined on the basis that how rules are presented and discussed in the paper with their logical correctness.
Case Study: It identifies the real-life example use to check the validity of the presented approach. Possible values are yes and no.
Automatable: It identifies the automation of presented consistency checking technique. Possible values are high, medium and low which are assigned when paper's author mention it or on the probability of approach automation.
Tool Support: It identifies the tool develop to automate the proposed work. Tool support helps in quick validation of the proposed work. Possible values are yes and no.

4. Consistency Checking Techniques

This section provides a discussion on existing consistency checking techniques between UML models. This section also analysis existing techniques with the help of analysis table build from analysis parameters. The consistency checking techniques are divided by intermediate representation into three categories.

4.1. Formally Represented Techniques

This subsection discusses and analysis those existing techniques in which intermediate representation is defined in any formal language or in some formal notation.

Engels et al. : Engels *et al.* [8] use UML 1.3 sequence, collaboration and statechart diagrams to validate consistency between them. The authors focus on representing UML behavioral models formally as it is easy and accurate to determine inconsistencies between formal models. The authors define a step wise approach to extract problems between UML models. The approach is proposed for real time UML (UML-RT) applications. Monitoring strategy is used through which consistency rules are presented in terms of definitions and conditions on formally represented model. A traffic light case study is also demonstrated. A tool named *FDR* is developed to implement proposed work.

Rasch and Wehrheim: Rasch and Wehrheim [9] focus on UML 1.5 class and state machine diagrams for consistency validation. Classes and state machines are translated to CSP-OZ which is semantically powerful. CSP-OZ is used because it helps in complete class definition as well as ordering methods execution. Translation of class and state machine to Object-Z is also discussed. Rules are presented to validate the translation accurately. Some propositions are also provided to ensure the consistent transformation of class and state machine to Object-Z.

Kim and Carrington: Kim and Carrington [10] provide meta-model level rules against consistency validation in statechart. Statechart is represented in Object-Z to apply consistency rules. The main focus of their approach is to define integrity consistency constraints between different UML models. Predicates are used for defining the invariants and integrity constraints. The authors provide translation of statechart meta-level constructs to Object-Z.

Shinkawa: Shinkawa [11] uses UML 2.0 use-case, class, sequence, activity and statechart diagrams for consistency checking. The paper presents UML models classification and the author includes one diagram from every classification. The author provides mapping for converting UML diagrams to Colored Petri Nets

(CPN). The consistency is validated through translation of UML diagrams to CPN. Different examples are used to demonstrate the translation. Colored Petri Nets (CPN) is used as an intermediate representation. Inter-model consistency is validated between UML models. Rules are presented to translate each UML diagram to CPN.

Liu et al. : Liu *et al.* [12] base UML 2.0 class, sequence and statechart diagrams to propose a consistency checking approach. Object Oriented Specification Language (OOL) is used to present software design formally. Theoretical proofs are provided to show the power of OOL to present UML diagrams. Some constraints are defined to ensure consistency between OOL formal representations. UML diagrams are translated to OOL through point-of-sale case study. No tool support is provided. Analysis strategy is implemented by the approach.

Bernardi et al. : Bernardi *et al.* [13] use UML 1.4 sequence and statechart diagrams to validate consistency. UML models are represented in generalized stochastic Petri nets (GSPNs) automatically. GSPN is a formal representation and the authors believe that consistency problems between UML diagrams occur because it lacks formalism. Monitoring strategy is used through which consistency rules are discussed briefly. Analysis of the presented approach is provided on a *watchdog* mechanism example which is mainly used for error detection in fault tolerant systems. No tool support is provided.

Haesen and Snoeck: Haesen and Snoeck [14] use UML 1.5 class and finite state machine diagrams to propose a consistency checking approach. Class diagram is used to represent the static structure while object event table (OET) is used to maintain all the events that can occur during the life time of the system. Finite State machines (FSM) are used to demonstrate the behavior of OET events and it is assumed that interactions between objects are handled through the events. A MERODE methodology provides a formal definition of UML diagrams. UML patterns are used to implement consistency strategies.

Satoh et al. : Satoh *et al.* [15] propose a consistency validation method on UML 2.0 class diagram. UML class is translated into logic program for contradiction finding between two different versions of a class diagram for consistency. Mapping rules are provided to translate UML class diagram into logic program. Consistency checking is applied on logic program. Contradictory parts are deleted from the logic program to get a consistent class diagram. The proposed translation rules are bi-directional such as they support translation between class diagram and logic program in both directions. An algorithm is proposed through which contradictory part between different versions of class diagrams can be recovered.

Straeten and Simmonds: Straeten and Simmonds [16] discusses consistency checking on UML 1.5 class, sequence, and statecharts diagrams. *ALCQI* is used to present UML models formally. The authors use a method for representing class diagram into *ALCQI* by Berardi [17]. Berardi demonstrates the translation of class diagram constructs such as classes, associations, aggregations, generalization, and constraints into *ALCQI*. Berardi also provides experimentation to validation the proposed translation methods.

4.2. Extended UML Representation Techniques

This subsection discusses and analysis those presented techniques in which intermediate representation is defined as an extension in UML diagram(s).

Engles et al. : Engles *et al.* [18] use UML 1.4 sequence, collaboration and statechart diagrams for consistency validation. Dynamic Meta-Modeling (DMM) rules are used for this purpose. The rules are provided as two constraints {new} and {destroyed}. {new} constraint initiates a create call to attached class. {destroyed} constraint deletes the attached instance from the execution space. The tester environment is also provided to validate the consistency.

Mens et al. : Mens *et al.* [7] extend UML 2.0 class, sequence and statechart diagram meta-models to include versioning support for consistency validation. Description logic (DL) is used to detect and resolve inconsistencies formally. Five stereotypes as <<versioned>>, <<horizontal>>, <<evolution>>, <<refine>>, and <<trace>> are also included in UML extension. OCL is used for stereotypes definition.

4.3. No Intermediate Representation Techniques

This subsection discusses and analysis those presented techniques in which no intermediate representation.

Grischick: Grischick [19] uses UML 1.5 class diagram to detect inconsistencies between two versions. An algorithm is proposed for this purpose. Different color codes are used for distinguishing properties of class diagram. The approach not only tells difference between two versions but also provides information that how the difference is produced. A data structure is

proposed for the approach implementing. Evaluation function is used to compare any two elements of class diagram. Critical analysis is presented against proposed algorithm.

Graaf and Deursen: Graaf and Deursen [20] focus on UML 1.4 scenario (sequence, collaboration) and statechart diagrams for consistency checking. The approach generates statecharts from scenarios. Transformation rules are provided in ATL for generating statecharts from scenarios. Scenarios are created from use-cases. Transformation is done in four steps. First step applies domain knowledge. Second step generates flattened statecharts. Third step merges flattened statecharts against their respective class. Last step introduces hierarchy information in merged statecharts.

Briand et al. : Briand *et al.* [21, 22] use UML 1.4 class, sequence and statechart diagrams for evaluating consistency. The approach focuses on change management in UML models. Change affect is also estimating by the approach before actual change implementation. Monitoring strategy is applied through which consistency rules are implemented in the tool. OCL expressions are used for expressing the rules. ATM is used as case study to validate the feasibility of the approach. Experimental analysis and results are also provided.

Feng and Vangheluwe: Feng and Vangheluwe [23] use UML 1.5 class sequence and statechart diagrams for consistency validation. The approach covers client-server applications. Consistency issues are resolved between components. Rules are presented for validating consistency. Output traces are used for this purpose. Rule consists of four parts as *pre-condition*, *post-condition*, *guard (optional)* and *counter-rule property (optional)*.

Egyed: Egyed [24] uses UML 1.3 class, sequence diagrams and statechart diagrams to validate consistency between them. The approach is applied on runtime instances for consistency validation. The issues in runtime instances are resolved through scope of a consistency rule and a logger. Rules are provided in OCL through monitoring strategy. UML/ ANALYZER tool [25] is developed to automate the approach. It contains three components as *consistency checker*, *evaluation profiler*, and *rule detector*. The tool is integrated in IBM Rational Rose for open use. Experimental results are presented with the help of the developed tool.

Bellur and Vallieswaran: Bellur and Vallieswaran [26] use UML 1.5 use-case, class, sequence, statechart, component and deployment diagrams for consistency validation. The paper discusses some consistency issues and provides solution for them. Relational-metamodel is used for consistency checking based on four views as requirement, development, source, and deployment. Consistency is enforced during the construction of UML models. XMI is used to represent UML model. Consistency rules are also applied on an XMI. Two case studies as Document Viewer and ATM are demonstrated. They incorporate their work as a plug-in to an open source tool known as Eclipse.

5. Conclusion

In this paper, we presented a survey of consistency checking techniques for UML models. Intermediate representation is used to classify the existing techniques. The consistency validation mechanism is discussed in existing techniques. A generalized set of parameters is also defined. Analysis table is constructed to analyze the existing techniques on the basis of analysis parameters. The analysis reflects that formalization of UML models is preferable to validate consistency because it helps in removing ambiguities and enforce consistency. Most of the approaches implement consistency validation rule between UML models in a tool which lies under monitoring strategy and help in quick validation. Intra and inter model consistency types are used in nearly all the approaches.

6. References

[1] G. Booch : Object Oriented Design with Applications, Benjamin/Cummings, 1991, Redwood, California. ISBN: 0-8053-0091-0.

[2] I. Jacobson, G. Booch, and J. Rumbaugh : The Unified Software Development Process, Addison-Wesley, Reading, MA, 1999.

[3] G. Booch, J. Rumbaugh and I. Jacobson : The Unified Modeling Language User Guide, Addison Wesley, 1999. ISBN: 0-201-57168-4.

[4] OMG, Unified Modeling Language Specification, Version 2.1.1, (2007-02-07).

[5] MDA Guide (2003), Version 1.0.1 Object Management Group (OMG) on 01-06-2003, http://www.omg.org/docs/omg/03-06-01.pdf.

[6] W-H. Roetzheim and R-A. Beasley : Software Project Code and Schedule Estimating: Best Practices, Prentice Hall PTR; Pap/cdr edition November 1997. ISBN: 978-0136820895

[7] T. Mens, R. V-D. Straeten, and J. Simmonds : A Framework for Managing Consistency of Evolving UML Models, *In H. Yang, editor, Software Evolution with UML and XML,* chapter 1. Idea Group Inc., March 2005.

[8] G. Engels, J. M. Kuster, and L. Groenewegen : A Methodology for Specifying and Analyzing Consistency of Object-Oriented Behavioral Models, *In Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE'01)*, pp. 186-195, September 2001, Vienna, Austria, ACM Press.

[9] H. Rasch and H. Wehrheim : Consistency Checking in UML Diagrams: Classes and State Machines, *In Proceedings of the 6th Formal Methods for Open Object-Based Distributed Systems (FMOODS'03),* pp. 229-243, November 2003, Paris, France, Springer-Verlag, LNCS 2884.

[10] S-K. Kim and D. Carrington : A Formal Object-Oriented Approach to defining Consistency Constraints for UML Models, *In Proceedings of the 15th Software Engineering Conference (ASWEC'04),* pp. 87- 94, April 2004, Melbourne, Australia, IEEE.

[11] Y. Shinkawa; Inter-model Consistency in UML Based on CPN Formalism, *In Proceedings of the 13th Asia Pacific Software Engineering Conference (APSEC'06),* pp. 411-418, December 2006, Bangalore, India, IEEE Computer Society.

[12] Z. Liu, X. Li, J. Liu, and J. He : Integrating and refining UML models. Technical Report 295, UNU/IIST, PO Box 3058, Macao SAR China, 2004. Presented at UML 2004 Workshop on Consistency Problems in UML-based Software Development, October 10-15, 2004, Lisbon, Portugal.

[13] S. Bernardi, S. Donatelli and J. Merseguer : From UML Sequence Diagrams and Statecharts to Analyzable Petri Net models, *In Proceedings of the 3rd International Workshop on Software and Performance (WOSP'02),* pp. 35-45, July 2002, Rome, Italy.

[14] R. Haesen, M. Snoeck : Implementing Consistency Management Techniques for Conceptual Modeling. *In Proceedings of the 3rd International Workshop on Consistency Problems in UML* based *Software Development III Understanding and Usage of Dependency Relationships*, pp. 99-113, October 2004, Lisbon, Portugal.

[15] K. Satoh, K. Kaneiwa, and T. Uno: Contradiction Finding and Minimal Recovery for UML Class Diagrams, *In Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06),* September 2006, Tokyo, Japan. IEEE Computer Society.

[16] R. V-D Straeten, J. Simmonds, and T. Mens : Detecting inconsistencies between UML models using description logic. *In Proceedings of the International Workshop on Description Logics (DL'03),* September 2003. Rome, Italy. CEUR-Workshop Proceedings.

[17] D. Berardi. Using DLS to reason on UML class diagrams. In Proceedings of the Workshop on Applications of Description Logics (ADL'02), September 2002, Aachen, Germany. CEUR-Workshop Proceedings.

[18] G. Engles, J. H. Hausmann, R. Heckel, and S. Sauer, Testing the Consistency of Dynamic UML diagrams, *In* Proceedings *of the 6th Integrated Design and Process Science (IDPT'02),* June 2002, Pasadena, California.

[19] M. Grischick : Difference Detection and Visualization in UML Class Diagrams, Department of Computer Science on Metamodeling and its Application, Technical University of Darmstadt, Technical Report TUD-CS-2006-5, 2006.

[20] B. Graaf and A-V. Deursen, Model-Driven Consistency Checking of Behavioral Specifications, *In Proceedings of the 4th International Workshop on Model-Based* Methodologies *for Pervasive and Embedded Software (MOMPES'07),* pp. 115-126, March 2007, Braga, Portugal, IEEE Computer Society.

[21] L. C. Briand, Y. Labiche, and L. O'Sullivan : Impact Analysis and Change Management of UML Models, *In Proceedings of the 19th International Conference* Software *Maintenance (ICSM'03),* pp. 256-265, September 2003, Amsterdam, Netherlands, IEEE Computer Society Press.

[22] L. C. Briand, Y. Labiche, L. O'Sullivan, and M. M. Sowka : Automated Impact Analysis of UML Models, *Journal of Systems and Software,* vol. 79, issue. 3, pp. 339-352, March 2006.

[23] T-H. Feng and H. Vangheluwe : Case Study: Consistency Problems in a UML Model of a Chat Room, *In Proceedings of the 6th International Conference on* the *Unified Modeling Language (UML'03),* October 2003. San Francisco, USA.

[24] A. Egyed : Instant Consistency Checking for the UML, *In Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, May 2006, Shanghai, China. ACM Press.

[25] A. Egyed : UML/ANALYZER: A Tool for the Instant Consistency Checking of UML Models, *In Proceedings of* the *29th International Conference on Software Engineering (ICSE'07),* pp. 793-796, May 2007, Minneapolis, MN, USA, IEEE Computer Society.

[26] U. Bellur and V. Vallieswaran : On OO Design Consistency in Iterative Development, *In Proceedings of the 3rd International Conference on Information Technology: New Generations (ITNG'06),* pp. 46-51, April 2006.

Ref. No	Nature	MDA Based	UML Version	UML Diagrams	Consistency Type	Intermediate Representation	Consistency Strategy	Rules Provided	Case Study	Automatable	Tool Support
[8]	Behavioral	N	1.3	SD, CLD, SC	Inter-model	CSP-OZ	Monitoring	M	Y	H	Y
[9]	Both	N	1.5	CD, SM	Intra-model	CSP-OZ	Monitoring	M	Y	H	N
[10]	Behavioral	N	1.3	SC	Inter-model	OZ	Monitoring	H	N	H	N
[11]	Both	N	2.0	UCD, CD, SD, AD, SC	Inter-model	CPN	Analysis	H	N	H	N
[12]	Both	N	2.0	CD, SD,SC	Intra-model	OOL	Analysis	H	Y	H	N
[13]	Behavioral	N	1.4	SD, SC	Semantic & Syntactic	LGSPN	Monitoring	M	Y	M	N
[14]	Both	N	1.5	CD, SC (FSM)	Intra-model	MERODE	All	L	N	H	Y
[15]	Structural	N	2.0	CD	Evolution	Logic Program	Monitoring	M	N	L	N
[16] [17]	Both	N	1.5	CD, SD, SC	Intra & Evolution	ALCQI	Monitoring	L	N	L	N
[18]	Behavioral	N	1.4	SD, CLD, SC	Intra-model	Extended UML	Monitoring	M	N	M	N
[7]	Both	N	2.0	CD, SD, SC	All	Extended UML	Monitoring	H	Y	H	Y
[19]	Structural	N	1.5	CD	Evolution	N	Analysis	M	Y	H	Y
[20]	Behavioral	Y	1.4	SD, SC	Intra-model	N	Analysis & Construction	H	Y	H	N
[21] [22]	Both	N	1.4	CD, SD, SC	Intra-model	N	Monitoring	H	Y	H	Y
[23]	Both	N	1.5	CD, SD, SC	Intra-model	N	Monitoring	L	Y	M	N
[24] [25]	Both	N	1.3	CD, SD, SC	Intra-model	N	Monitoring	M	Y	H	Y
[26]	Both	N	1.5	UCD, CD, SD, SC, CPD, DD	Inter-model	N	Construction	M	N	H	Y

Table 1: Analysis of UML diagrams Consistency Checking Techniques

Abbreviation	Value
H	High
M	Medium
L	Low
Y	Yes
N	No

Abbreviation	Value
UCD	Use-case Diagram
CD	Class Diagram
SD	Sequence Diagram
CLD	Collaboration Diagram
SM	State Machine
SC	Statechart

Abbreviation	Value
AD	Activity Diagram
CPD	Component Diagram
DD	Deployment Diagram
FSM	Finite State Machine

Table 2: Abbreviations Used in table 1