International Journal of Information Management 36 (2016) 883–899

Contents lists available at ScienceDirect

International Journal of Information Management

journal homepage: www.elsevier.com/locate/ijinfomgt

UML models consistency management: Guidelines for software quality manager

Raja Sehrab Bashir[a,*], Sai Peck Lee[a], Saif Ur Rehman Khan[a], Victor Chang[b], Shahid Farid[c]

[a] Department of Software Engineering, University of Malaya, Kuala Lumpur, 50603, Malaysia
[b] International Business School Suzhou, Xi'an Jiaotong Liverpool University, Suzhou, China
[c] Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan

ARTICLE INFO

Article history:
Received 28 May 2016
Accepted 28 May 2016

Keywords:
UML model consistency
UML model transformation

ABSTRACT

Unified Modeling Language (UML) has become the de-facto standard to design today's large-size object-oriented systems. However, focusing on multiple UML diagrams is a main cause of breaching the consistency problem, which ultimately reduces the overall software model's quality. Consistency management techniques are widely used to ensure the model consistency by correct model-to-model and model-to-code transformation. Consistency management becomes a promising area of research especially for model-driven architecture. In this paper, we extensively review UML consistency management techniques. The proposed techniques have been classified based on the parameters identified from the research literature. Moreover, we performed a qualitative comparison of consistency management techniques in order to identify current research trends, challenges and research gaps in this field of study. Based on the results, we concluded that researchers have not provided more attention on exploring inter-model and semantic consistency problems. Furthermore, state-of-the-art consistency management techniques mostly focus only on three UML diagrams (i.e., class, sequence and state chart) and the remaining UML diagrams have been overlooked. Consequently, due to this incomplete body of knowledge, researchers are unable to take full advantage of overlooked UML diagrams, which may be otherwise useful to handle the consistency management challenge in an efficient manner.

1. Introduction

Software maintenance is the most important phase of a software development life cycle as maintenance consumes almost 40–80% of the total software development cost (Fernández-Sáez, Genero, Caivano, & Chaudron, 2016). Moreover, 60% of the total maintenance cost is spent on enhancing the existing functionality of the software. Therefore, it is of vital importance to prepare appropriate software artifacts at each phase to reduce the maintenance cost. Maintenance cost can be reduced by improving the comprehension of a software system. Comprehension of a software system consumes about 50% of time of the maintenance phase. Different modeling languages have emerged to help represent the software system in a graphical notations and increase the system comprehension (Dzidek, Arisholm, & Briand, 2008).

Model-Driven Software Engineering (MDSE) is a discipline aimed at promoting the models for software development and maintenance. Model-based representation of a software system provides better understanding about underlying concepts (e.g., classes, methods, aggregation, association, multiplicity, and operations) (Misbhauddin & Alshayeb, 2015). Moreover, models provide a complete and detailed specification to better communicate the structure and behavior of a software system under development (Brambilla, Cabot, & Wimmer, 2012). Unified Modeling Language (UML) is the most popular modeling language and become the de-facto standard to design today's large object-oriented systems. UML offers 14 different diagrams to represent the structure and behavior of a software system (Holt, 2004). The structural diagrams represent the static aspect of a software, whereas behavioral diagrams are used to represent the dynamic aspects of an under development software.

UML has gained popularity in the last 15 years due to its multi-view support. A good quality model requires associated UML diagrams, which should be consistent with each other since they represent the same system but from different viewpoints. Change in one diagram (e.g., due to change in user requirements) can ulti-

* Corresponding author at: University Malaya, Faculty of Computer Science and Information Technology, Malaysia.
E-mail addresses: sehrabraja@siswa.um.edu.my (R.S. Bashir), saipeck@um.edu.my (S.P. Lee), saif.rehman@siswa.um.edu.my (S.U.R. Khan), ic.victor.chang@gmail.com (V. Chang), shahidfarid@bzu.edu.pk (S. Farid).

http://dx.doi.org/10.1016/j.ijinfomgt.2016.05.024
0268-4012/

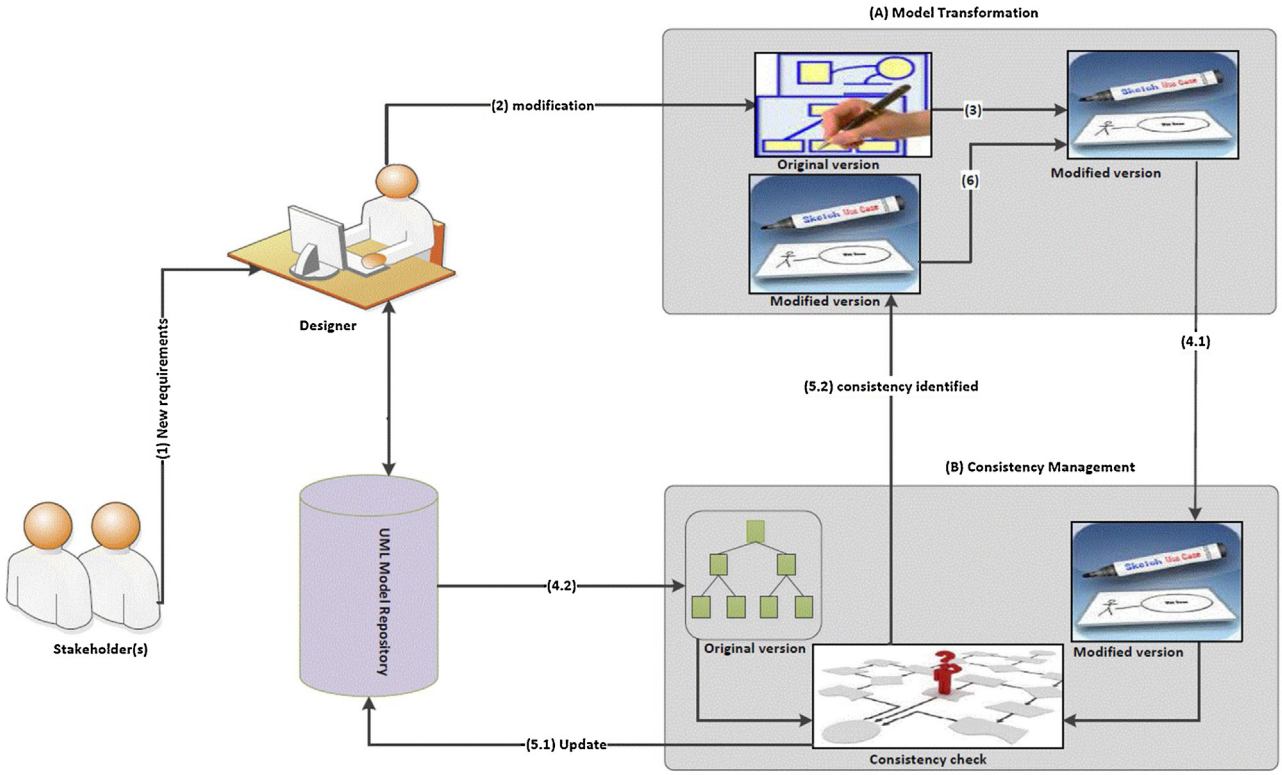

Fig. 1. Abstract representation of model consistency management.

mately affect the underlying model, and as a result, it may cause unwanted changes in other associated UML diagrams of the same model. The term model consistency is defined by Spanoudakis et al., as "*a state in which two or more elements, which overlap in different models of the same system, have a satisfactory joint description*" (Spanoudakis & Zisman, 2001). For example, if a method (mi) is used in a sequence diagram and mi is also mentioned in the class diagram, then we can conclude that both sequence and class diagrams are consistent with respect to mi (Object, 2007). UML model consistency is heavily affected by the number of UML views that may cause a number of errors in the developed system (Muskens, Bril, & Chaudron, 2005). UML model consistency ensures two types of correct transformations: (i) model-to-model: one UML diagram is transformed into other diagrams (e.g., UML sequence diagrams are transformed into the corresponding UML interaction diagrams), and (ii) model-to-code: code is generated automatically from the UML diagram (Lucas, Molina, & Toval, 2009).

Model consistency management includes a number of activities for finding and managing the consistency problems. In order to identify the consistency problems, the software engineer focuses on finding the ambiguities between UML models. In UML models, consistency problems can be occurred in the naming convention, integration of two models, or interaction of models, etc. (Spanoudakis & Zisman, 2001). Fig. 1 depicts the abstract overview of model consistency management stages. At the top level, it contains two main stages including model transformation and consistency management (Fig. 1). First of all, the developer edits the original source model based on the users' given new requirements. After that, the developer performs model analysis (in the consistency management stage), where two possible scenarios can be occurred: (i) models are consistent, and (ii) consistency problems are identified. In the case of consistent models, the original model will be updated. On the other hand, the consistency problem is analyzed to determine the type of consistency (i.e. syntactic, semantic, horizontal and vertical).

After identifying the consistency problems, forwarded to the model transformation stage. Subsequently, perform most suitable transformation based on the developer feedback and type of the consistency problem. Finally, consistency between the transformed model and the original model is checked to ensure consistency.

The Model-Driven Architecture (MDA) has been attracting more attention for the last 15 years. The main objective of MDA is to promote system modeling using UML diagrams. In the scope of MDA, consistency management becomes one of the most important research issues, especially when a software system evolves. In this regard, an extensive amount of research has been conducted for proposing new techniques to improve model quality and also maintain consistency between models. However, current state-of-the-art consistency management techniques lack in exploring and analyzing consistency problems in a complete manner. The existing studies mainly focused on a single view (i.e., structural or behavioral) and horizontal (e.g., intra model) consistency management problems. Moreover, A few studies considered multi-aspects of the system during model transformation. However, prior work overlooked vertical (e.g., inter model) and semantics model consistency problems. The observations and contradictions indicate that the current research on consistency management needs more attention on vertical and semantic model consistency problems. Hence, new approaches need to be developed to support enhancement in UML model with new concepts and ideas. However, new approaches cannot be developed unless the deficiencies in current approaches are analyzed, compared, and research gaps are identified. Motivated by this, we conducted an extensive survey of state-of-the-art approaches focusing on UML model consistency management.

We nevertheless acknowledge that prior work provides an overview of the consistency management techniques. There are two main surveys published in this field of study (Lucas et al., 2009; Spanoudakis & Zisman, 2001). Spanoudakis and Zisman (2001) conducted a first ever survey that focuses on consistency management. The survey broadly focused on consistency management