ISSN 2278-3091
Volume 10, No.1, January - February 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse221012021.pdf
https://doi.org/10.30534/ijatcse/2021/221012021

Ontology-Based Transformation and Verification of UML Qualified Association

Abdul Hafeez[1], Asif Wagan[2], Samreen Javed[3] , Imtiaz Hussain[4]
[1]Department of Software Engineering SMI University Karachi, Pakistan, ahkhan@smiu.edu.pk
[2]Department of Computer Science SMI University Karachi, Pakistan, asif.wagan@smiu.edu.pk
[3]Department of Computer Science SMI University Karachi, Pakistan, SamreenJaved@smiu.edu.pk
[4]Department of AI & Mathematical Sciences SMI University Karachi, Pakistan, imtiaz@smiu.edu.pk

ABSTRACT
UML class model is an essential element of today's software development process. In modern software development methodologies, it is considered a key contributor in every phase of software development. It may be automatically converted into other UML models and even in programming code. However, the erroneous model generates other erroneous models. The model verification technique checks the presence of error in the UML class model. This paper's main objective is to introduce a technique for the completely automatic and expressive transformation of the UML class model's qualified association into ontology. Because the current verification method does not support the transformation and verification of qualified associations. Later on, the ontology-based reasoning method is presented to verify qualified associations and their constraints.

Key words: Ontology, Model Verification, Class model, Verification Tool, MDE

1. INTRODUCTION

Software are everywhere, and they are not only in the computer but also in home appliances, mobile phone, car, and other devices. But the failure of software causes economic and lives losses. For example, Cloudflare sensitive customer data such as passwords, cookies leaked from customer websites in 2017, due to a bug in the software [1]. Due to wrong software calculation, 3200 US prisoners were released early from 2003 to Dec 2015 [2]. However, sometimes software failure not only cost in terms of money, e.g., in Saudi Arabia, the US missile defense system failed to identify an attack due to the erroneous calculation in the software in 1991 and 28 American soldiers losses their life. These are few examples of misfortunes due to software errors [3]. Hence, software correctness is a vital problem in the industry, and precise software testing is crucial before deployment. Although testing has some limitations: it checks the absence of errors, and it is performed in later stages (after coding). Error correction cost is higher in later stages as compared to the initial stages [4]. The identified testing problems can be easily tackled through model verification. Innovative software development methods such as MDA consider UML models as an integral part of the entire development process. [5,6]. In MDA, automated model-to-model conversion provides systematic reuse of existing software artifacts. However, the automatic transformation may generate some problems, e.g., models can be developed with bugs, and ultimately these bugs can implicitly be shifted in the code. Hence, the model must be verified for better software [7,8].

The UML class model is an essential UML component and specifies static aspects of the system [9]. The class model comprises classes and various kinds of relationships (dependency, association, and generalization) [10,11,12]. Existing verification methods of the UML class model are sufficiently good. However, support of some important elements of the UML class model is missing. For example, a comparison of the different verification methods of the class model presented in [13] claims that none of the verification methods has the support of qualified associations. Previously many researchers have presented work on verification of the UML class model. A detailed formal transformation of the UML meta-model in Z notation presented by France et al. [14]. Object-Z is an extension of Z notation, which has object-oriented capability has also been used to represent the UML meta-model's abstract syntax [15, 16]. B method has also been applied to formalize and verify the UML class model [17,18]. They used B prover to verify consistency against the well-formedness rules [18]. They transformed the well-formedness rules through the B abstract machine invariants.

Many works have also employed various semi-formal techniques to verify the UML class models, e.g., Constraint Satisfaction Problem (CSP) and Alloy. A linear inequality-based method proposed by Cadoli et al. [19] for UML class mode verification. They used CSP for representing and solving linear inequalities. Alloy, a semi-formal method, has also been used to transform and verify the UML class model. Bordbar et al. [20] presented the UML Class model's

transformation with OCL into the Alloy. This approach transformed the meta-model into the Alloy and class model into Alloy's signature as a meta-model instance. Maoz et al. [21] presented formalization of advanced UML model elements into the Alloy, such as interface and multiple inheritances.

Different researchers also used ontology to verify the UML class model, such as a comparison between UML and Web Ontology Language (OWL) presented by Xu et al. [22]. They presented that UML and OWL have many resemblances, such as classes, relationships, and attributes.

2. QUALIFIED ASSOCIATION

Association between classes also annotated by additional properties called qualifier. The qualifier imposes an additional constraint called a qualifier constraint. It allied with multiplicity and created a partition on the target class instances. The qualified association can be divided into two types. In type 1, multiple instances of target class related to a single instance of source class under a category as shown in figure 1 where the association "enroll" connecting department and students through" session". This association is specifying that one or many departments enroll many students under a session. If we semantically investigate this association, we easily found that department class instances are linked with student class instances through the session. For clarity, figure 2 shows the semantic representation of the UML class model shown in figure 1. Figure 2 have two departments {D1, D2} and both have two sessions, such as the D1 has (Session Fall 19 and Session Spring 20), each batch has students, and D2 has (Session Fall 19 and Session Fall 20). In type 2, A key-value is attached with the source class, which uniquely related instance of the target class. This association is trivial and can be achieved through unique key constraints.

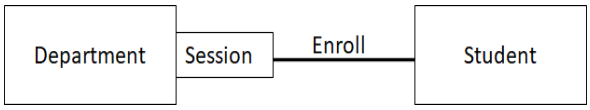

Figure1: Department Student class model

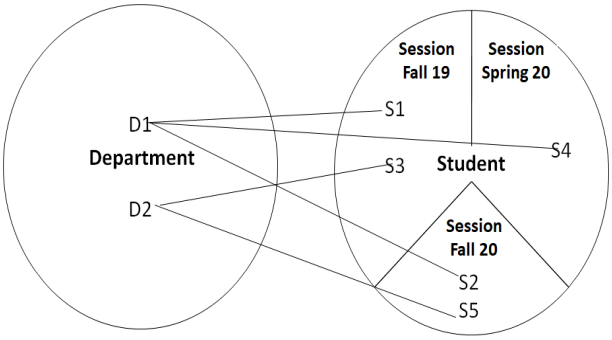

Figure 2: Qualified association partition

3. SOLUTION

A new class is introduced for the qualifier in between the source and target classes in the proposed solution. The source class is connected with the newly introduced class through association, and the new class is connected with the target class through a new association called "hold" (shown in figure 3) and formalized in ontology as :

Department ⊑⊤
Session ⊑⊤
Student ⊑⊤
Hold ⊑ObjectProperty
Enroll ⊑ObjectProperty
Holdby⊑ObjectProperty
Enrollby⊑ObjectProperty
Hold(Department,Session)
Holdby(Session,Department)
Enroll(Session ,Student)
Enrollby(Student,Session)
Hold ≡ Holdby-
Enroll ≡ Enrollby-

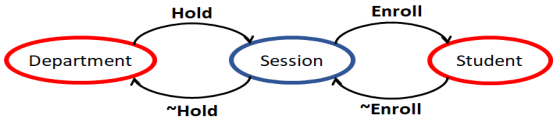

Figure 3 : Department student class model ontology

4. EXPERIMENTS AND RESULT

Numerous ontology tools are available in the market. However, protégé is a widely accepted, and open-sourced tool. Initially, it was developed for biomedical projects. It currently supports various formats to represent ontology, such as RDF/XML, Turtle, OWL/XML, and OBO. Furthermore, it also supports many reasoners such as pellet, racer, fact++, and HermiT.

The results illustrate that the method proposed in this work can efficiently formalize and verify qualified association. For evaluation, we implement the UML class model shown in figure 1 in Protégé.

In the ontology model development, the top-level UML class model elements such as classes are transformed into the ontology concepts, as shown in figure 4. Further, UML class model associations are converted into ontology object properties. Then, the domain and range of object property are set. Finally, proposed constraints are applied in the ontology, and the model's verification is performed through the Protégé reasoner as shown in figure 5.

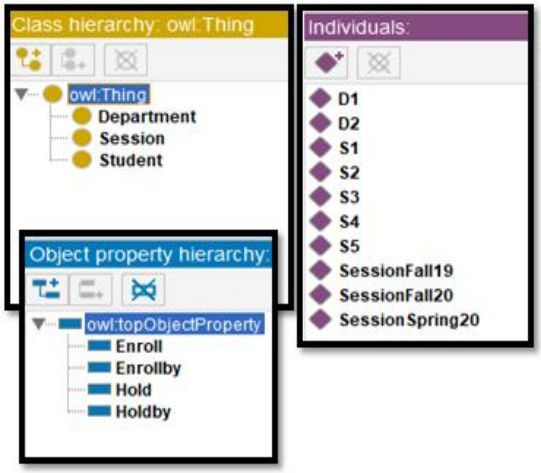

Figure 4: Ontology of Qualified association

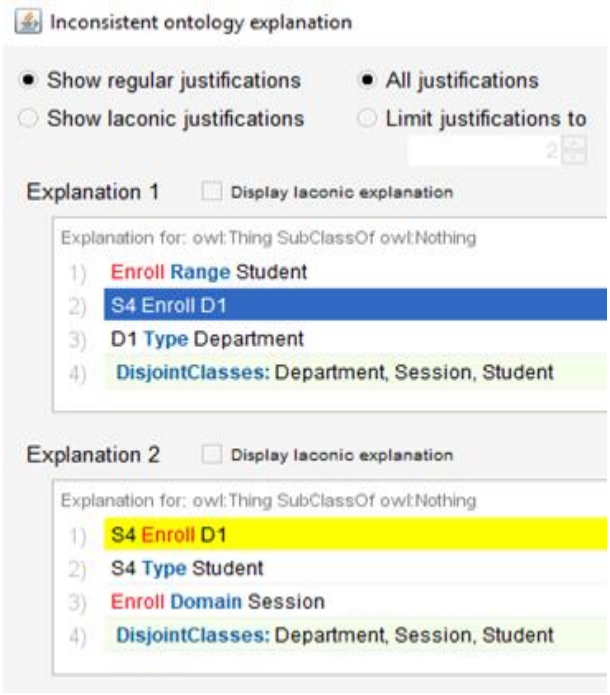

Figure 5: Verification result obtained in Protégé

5. CONCLUSION

UML class model formalization and verification are very important in modern software practices, such as Model Driven Architecture. In previous work, several UML class model elements were formalized and verified through different techniques. However, some crucial elements, such as qualified association was never checked. The proposed method performs transformation and verification of qualified association of UML class model through ontology. This transformation map qualified associations into the new ontology class, added in the middle of the source and target class, and examine various correctness features such as consistency. The method proposed in this work has many advantages because ontology supports various efficient reasoners, which can perform reasoning on large models very efficiently. In the future work, we transformed the OCL constraints and other UML class model unsupported elements.

REFERENCES

1. N. H. Hussein and A. Khalid. **A survey of cloud computing security challenges and solutions**, *International Journal of Computer Science and Information Security*, Vol. 14, p. 52, 2016.
2. Technica . **Software bug granted early release to more than 3,200 us prisoners**. Available on https://arstechnica.com/tech-policy/2015/12/software-bug-granted-early-release-to-more-than-3200-us-prisoners/. Accessed: 2020-05-28.
3. M. Defense. **Software problem led to system failure at Dhahran**, *Saudi Arabia, US GAO Reports*, report no. GAO/IMTEC-92-26, 1992.
4. K. Erdil, E. Finn, K. Keating, J. Meattle, S. Park. **Software maintenance as part of the software life cycle,** *Comp180: Software Engineering Project*, pp. 1–49, 2003.
5. M. Kardoš and M. Drozdová. **Analytical method of CIM to PI, transformation in Model Driven Architecture (MDA)**, *Journal of Information and Organizational Sciences*, Vol. 34, pp. 89–99, 2010.
6. S. Kent. **Model driven engineering**, *in proc. International Conference on Integrated Formal Methods*, Berlin, Heidelberg, 2002, pp. 286–298
7. A. Shaikh and U. K. Wiil. **Overview of slicing and feedback techniques for efficient verification of UML/OCL class diagrams**, *IEEE Access*, Vol. 6, pp. 23864–23882, 2018.
8. J. L. F. Alemán and A. T. Álvarez. **Can intuition become rigorous? foundations for UML model verification tools**, *in Proc. 11th International Symposium on Software Reliability Engineering*, 2000. pp. 344–355,
9. H. Malgouyres and G. Motet. **A UML model consistency verification approach based on meta-modeling formalization**, *in Proc. of the 2006 ACM Symposium on Applied Computing*, 2006, pp. 1804–1809.
10. M. Singh, A. K. Sharma and R. Saxena. **An uml + z framework for validating and verifying the static aspect of safety critical system,** *Procedia Computer Science*, 2016, Vol. 85, pp. 352–361.
11. R. Clarisó, C. A. González and J. Cabot. **Towards domain refinement for UML/OCL bounded verification**, *in Proc. International Conference on Software Engineering and Formal Methods Collocated Workshops*, York, UK, 2015, , pp. 108–114.
12. M. H. Awaad, H. Krauss and H.D. Schmatz. **Advanced praise for the unified modeling language reference manual reading**, *Zentralblatt für Bakteriologie*,

Parasitenkunde, Infektionskrankheiten und Hygiene. Erste Abteilung Originale. Reihe A: Medizinische Mikrobiologie und Parasitologie, Vol. 240, 1978.

13. A. Shaikh, U. K. Wiil and N. Memon. **Evaluation of tools and slicing techniques for efficient verification of UML/OCL class diagrams**, *Advances in Software Engineering*, Vol. 2011, 2011.
14. R. France, A. Evans, K. Lano and B. Rumpe. **The UML as a formal modeling notation**, *Computer Standards and Interfaces*, Vol. 19, pp. 325–334, 1998.
15. S. K. Kim and D. Carrington. **A formal mapping between UML models and object-Z specifications**, *in Proc. International Conference of B and Z Users*, York, UK, 2000, pp. 2–21.
16. S. K. Kim and D. Carrington. **A formal v&v framework for UML models based on model transformation techniques**, *in Proc. In the Proceedings of the 2nd MoDeVa Workshop - Model Design and Validation*, Montego Bay, Jamaica, Springer, 2005, pp. 1-7.
17. H. Ledang and J. Souquières. **Integrating UML and B specification techniques**, *in Proc. The Informatik 2001 Workshop on Integrating Diagrammatic and Formal Specification Techniques*, Vienna, Austria, 2001, pp. 1-8.
18. H. Ledang. **Automatic translation from UML specifications to B**, *in Proc. 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, 2001, San Diego, USA, IEEE, pp
19. M. Cadoli, D. Calvanese, G. De Giacomo and T. Mancini. **Finite satisfiability of UML class diagrams by constraint programming, CSP Techniques with Immediate Application**, *(CSPIA)*, vol. 2, 2004.
20. B. Bordbar and K. Anastasakis. **UML2alloy: A tool for lightweight modelling of discrete event systems**., *in Proc. International Conference Applied Computing*, Algarve, Portugal, IADIS, 2005, pp. 209–216.
21. S. Maoz, J. O. Ringert and B. Rumpe. **Cd2alloy: Class diagrams analysis using alloy revisited**, *in Proc. International Conference on Model Driven Engineering Languages and Systems*, Wellington, New Zealand, 2011 pp. 592–607.
22. W. Xu, A. Dilo, S. Zlatanova and P. van Oosterom. **Modelling emergency response processes: Comparative study on OWL and UML**, *Information Systems for Crisis Response And Management*, Harbin Engineering University, pp. 493–504, 2008.