Vol. 5, No. 2 | July – December 2021

Ontology-Based Transformation and Verification of UML/OCL Constraints

Abdul Hafeez[1], Asif Wagan[2], Aamir Umrani[3]

Abstract:
In Software Engineering (SE), the graphical models specify the system's architecture, connection, and characteristics. New SE methods such as Model Driven Architecture (MDA) utilize graphical models as a nucleus of all development activities. In the MDA, the UML class models are very important and play very significant role in software development. But UML class model did not have support of any formal System. Therefore, it is very difficult to verify the correctness of UML class model. This paper presents the transformation and verification of class diagram and Object Constraint Language (OCL) and transformation algorithm from Class model to ontology in the continuity of our research on UML and ontology integration. The class diagram is transformed into ontology, and constraints specified through OCL are transformed into SPARQL. The benefit of the method presented in the study is that the availability of many efficient reasoners which can perform reasoning on huge ontology models in a very adequate time. This electronic document is a “live” template and already defines the components of your paper

Keywords: *UML, OCL, Ontology, SPARQL*

1. Introduction

In the present time software are part of our daily life; they control the stock exchange, manage patient records, taking decisions, etc. However, software failure causes either losses of human life and economical. Therefore, the correctness of software must be testified before implementation. Testing has two main issues 1. Testing never gives 100% grantee of error-free software. It only checks specifics bugs that drive from the test cases 2. testing activity is executed after completion of code. The bug identification and rectification cost are much higher in the later phases than in earlier phases [1]. Furthermore, more complex and large software is required in the industry, requiring extensive human efforts, and software houses want to agility in release software due to completion with their rivals [2]. Hence, new software development techniques have been developed to tackle the issues, and Model Driven Architecture (MDA) is one of them.

In MDA approach, graphical models play key roles in the development. MDA uses Unified Modeling Language (UML) as the main modeling language. UML is an industry-standard, and currently, it is used in all development activities such as analysis, design and documentation, code generation, and testing [3][4]. It provides many diagrams that deal with various facets of software [5][6][7]. The UML Class diagram is very important part of UML [5][8][9][10]. It represented the real-world model via classes, collaboration, and constraints [11]. Object Constraint Language (OCL) is a constraint speciation language, and its small scripts are attached with UML for defining constraints, conditions, and business

[1] Department of Software Engineering, SMI University, Karachi, Pakistan
[2] Department of Computer Science, SMI University, Karachi, Pakistan
[3] Department of Business Administration, SMI University, Karachi, Pakistan
Corresponding Author: ahkhan@smiu.edu.pk

rules [12]. However, the MDE approach has some limitations, and it also has some limitations, such as it is not free from error risk. For example, the model may be created with bugs that can be indirectly transported into the code. Verification of the model can be a possible solution to the problem.

Current UML Class/OCL model transformation and verification techniques offer good support to verify the model's correctness. However, formal and semi-formal methods are used in them for the formalization of the model and their notation based on mathematics. It is numerously diversified from the UML and very hard to understand by the software engineer. On the other side, the UML class model and ontology have so many common components and are developed to represent real-world concepts [13].

The OCL is an important element of UML. It is used to specify constraints that add additional restrictions in the UML model. It can access objects attributes, operation, and navigate object to object through associations and query operation calls. It is specially used to apply integrity constraints on the class model and also be used in other UML models such as the state chart model.

SPARQL Protocol and RDF Query Language (SPARQL) is a semantic query language for manipulating ontology [14]. It is not only used for the query. It is also used for various other functionality, e.g. ASK and CONSTRUCT, which are used for checking constraint consistency. The ASK command is used to verify constraints consistency, and the CONSTRUCT command is used to inferring new information. The OCL constraints are transformed into SPARQL ASK Negation as failure (NAF) Query in the proposed method.

2. Methodlogy and Proposed Solution

In [14], we have proposed ontology-based approaches for transformation and verification UML class model and presented how the SPARQL can efficiently represent OCL. This work presents an extension of our ongoing research ontology-based verification of the UML Class and OCL [14] [15] [16][17][18]. It presents the detailed mapping of different OCL elements into SPARQL. Ontology and UML class/OCL model have various similar elements e.g. classes, collaborations, constraints, instances, and generalization. However, ontology has additional benefits, such as reasoning, and on the other hand, UML does not have an appropriate formal foundation and reasoning ability. The UML model and ontology have a difference, such as Open World Assumption (OWA), which is supported by ontology UML supports Close Work Assumption (CWA). In OWA, the currently unknown assumption is treated as true, and in CWA, an unknown assumption is treated false. We proposed the representation of UML and OCL constraints into the SPARQL negation as failure (NAF) queries for supporting CWA in the ontology.

2.1. Class Model Transformation

In this work, the UML classes are converted into ontology classes, and Class properties are converted into the Ontology datatype property. Associations are converted into the object properties of Ontology, and their multiplicities are transformed into the Qualified cardinalities. The complete detail of class model transformation can be found in [13]. In this work, we presented the transformation algorithm of the UML class model to Ontology. According to the algorithm, the proposed method will take the UML class model in XMI format, read the entire file, and convert the model element according to the proposed method

2.2. OCL to SPARQL Transformation

SPARQL has similar data types as OCL for example integer, real etc., and both support common operations and functions. OCL has 4 Basic type such as Integer, Real, String, and Boolean and SPARQL support all OCL basic datatype such as for Real SPARQL has a decimal, float, and double as shown in table 1. OCL integer transformed into xsd:integer, string into xsd:string, and Boolean into xsd:boolen.

```
Algorithm : Transformation (f as XMI File)
Pre: Required Class diagram in XMI format
        Post: OWL File
While f <> Eof
    1)  e←getElemet(f)
    2)  if e=UMLclass
          addOWLClass(UMLclass.name)
          while e.Attributes<>null
             a← e.Attributes
             addOWLDataProperty
        (a.ame,e.name as domain ,a.datatye
        as range)
    3)  if e=UMLassociation
          if (e.type = unidirectional)
            addOWLObjectProperty
        (e,e.sourse as domain, e.target as
        range)
          else
            addOWLObjectProperty
        (e,e.sourse as domain, e.target as
        range)
             addOWLInverseObjectProperty
        (e,e.traget as domain , e.sourse as
        range)
    4)  if e=UMLgeneralization
        a.  s ←Call SuperClass
        b.  AddOWLSubCLass(e,s)
             return (OWLModel)
End
```

TABLE I. Primitive Types.

OCL	SPARQL
Real	xsd:float,xsd:double,xsddecimal
Integer	xsd:integer
String	xsd:string
Boolean	xsd:Boolean

The primary computational operator of OCL, such as arithmetic, relational, and logical also supported by SPARQL, as shown in Table2. OCL has many types of functions such as number, string, conversion, and group. The number functions perform different manipulation on a number such as ceiling and floor of a number. The transformation of the numeric function into the SPARQL is shown in table 3.

TABLE II. Operation on primitive type

Arithmetic		Relational		Logical	
OCL	SPARQL	OCL	SPARQL	OCL	SPARQL
+	+	<	<	Or	Or
-	-	>	>	And	And
*	*	<=	<=	Not	Not
/	/	>=	>=		
		<>	!=		

TABLE III. OCL Function Transformation

Integer	
OCL	**SPARQL**
Abs()	Abs()
Floor()	Floor()
Round()	Round()
Ceil()	Ceil()
Mod()	NA
String	
OCL	**SPARQL**
Concat()	Concat()
Substring()	SUBSTR()
ToUpperCase()	UCASE()
ToLowerCase()	LCASE()
Size()	STRLEN()
Conversion	
OCL	**SPARQL**
toInteger()	Xsd:integer()
toReal()	Xsd:float() xsd:double() xsd:decimal()
toBoolean()	Xsd:Boolean
Group	
OCL	**SPARQL**
Max()	Max()
Min()	Min()
Sum()	Sum()
Count()	Count()

2.3. Transformation of Collection Operations

OCL provides various operations on the collection types. They are specially designed for projecting new collections from existing ones. this section discusses the transformation of the collection operation

2.3.1. Select and Reject operation

Select and Reject operations specify a selection from a specific collection. The select specifies a subset of a collection. It returns a collection that contains the elements where the Boolean-expression evaluates to true. In SPARQL, the select operation is mapped into a select query, as shown in table 4.

The reject operation is just the inverse of the select. It rejects all the elements where the Boolean-expression evaluates true. In SPARQL, it is mapped into the inverse of select

2.3.2. Include and Exclude

Include operation returns true if the specified object exists in the collection and exclude returns true when the object does not exist. In SPARQL, the includes and excludes are mapped into Exits and Not Exit, as shown in the example presented in Table 4.

2.3.3. ForAll, Exit and Collect

The ForAll operation declares multiple iterators, which iterate over the complete collection. It returns true if the expression is true for each element. In SPARQL, it mapped into the simple query filter without Exits and Not Exits, as shown in Table 4. The Exits operation returns true if at least expression is true for one element. In SPARQL, it can be map into the filter with No Exists statement, as shown in Table 4.

3. Conclusion

UML Class/OCL model constraints are essential elements of UML. It is used for graphically representing real-world entities. This paper presents a new method for the transformation and verification of OCL constraints into SPARQL. OCL and SPARQL have many common elements, such as data types, operators, and functions. However, different types of collection operations such as select, reject, includes, includes all etc. can be easily mapped into the SPARQL through NAF ASK query with Filter construct

TABLE IV. Equivalences of OCL operations

Includes	
context Employee inv: self.worksFor->includes(self.manages.Department)	Ask where { ?E :Manage ?D. Filter (NOT EXISTS {?E :Workin ?D}) }
Excludes	
context Employee inv: self.subordinates->excludes(self)	ASK where { ?E1 rdf:type Com1:Employee. ?E1 :Workin ?E2 Filter (EXISTS {?E1 :Workin ?E2}) Filter ((?E1 = ?E2))}
IncludesAll	
context Faculty inv: self.works.controls->includesAll(self.worksOn.Research project)	ASK where { ?F rdf:type Com1:Faculty. ?F :Work ?D. ?D :Manage ?RP. Filter (NOT EXISTS {?F :Workon ?RP})}
ExcludeAll	
Inverse of Include	
Exit	
context University inv: self.Faculty->exists(firstName = `Abdul')	ASK where { Filter (!(NOT EXISTS {?F :FName "Abdul"^^xsd:string})) }
Forall	
context University inv: self.Faculty->forAll(age <= 65)	ASK Where { ?F :age ?age. Filter (!(?age >65)) }
Select	
context University inv: self.Faculty->select(Sal > 10000)->notEmpty()	ASKwhere { Filter (NOT EXISTS {?D :iworkin ?F. ?F :Fsal 10000}) {select ?D where { ?D rdf:type :Department}}}
Reject	
context University inv: self.Faculty->reject(isMarried)->isEmpty()	Inverse of reject

REFERENCES

[1] K. Anastasakis, B. Bordbar, G. Georg and I. Ray "UML2Alloy: A Challenging Model Transformation", ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2007), LNCS, Vol. 4735,PP 436-450, 2007

[2] I.Traore, D. Aredo "Enhancing Structured Review with Model-Based Verification", IEEE Transactions on Software Engineering, Volume 30 Issue 11 PP. 736 - 753, 2004.

[3] M. Szlenk "Formal-Semantics-Reasoning-UML-Class-Diagram Dependability of Computer Systems" , DepCos-RELCOMEX '06. International Conference, vol., no., pp.51,59, 25-27 May 2006

[4] K. Anastasakis, B. Bordbar, G. Georg nd I. Ray,"On challenges of model transformation from UML to Alloy", Software & Systems Modeling Volume 9, Issue 1, pp 69-86 Springer, 2010

[5] M. Cadoli, D. Calvanese, G. De Giacomo and T. Mancini, "Finate Satisfibaility of Uml Class Diagram by Constraint Programming", Proc. of the 2004 International Workshop on Description Logics, volume 104 of CEUR Workshop Proceedings. CEUR-WS.org, 2004

[6] H. Malgouyres, G. Motet, "A UML Model Consistency Verification Approach Based on Metamodeling Formalization", SAC '06 Proceedings of the 2006 ACM symposium on Applied computing Pages 1804-1809 ACM 2006

[7] M. Laaziri, S. Khoulji, K. Benmoussa, K. M. Larbi, "Information System for the Governance of University Cooperation", Engineering, Technology &amp; Applied Science Research, Volume 8, Issue: 5, Pages: 3355-3359, October 2018 , https://doi.org/10.48084/etasr.2156

[8] A. Gonzlez, J. Cabot, "Formal verification of static software models in MDE: A systematic review", Information and Software Technology Volume 56, Issue 8, Pages 821–838, Elsevier, 2014

[9] M. BALABAN, A. MARAEE "Finite Satisfiability of UML Class Diagrams with Constrained Class Hierarchy", ACM Transactions on Software Engineering and Methodology (TOSEM) - In memoriam, fault detection and localization, formal methods, modeling and design TOSEM Homepage archive Volume 22 Issue 3, July 2013

[10] A. Artale, Diego Calvanese, Ang elica, "Full Satisfiability of UML Class Diagrams, Conceptual Modeling – ER" Lecture Notes in Computer Science Volume 6412, 2010, pp 317-331, Journal 2010

[11] Asadullah Shaikh, and Uffe Kock Wiil, "A feedback technique for unsatisfiable UMLOCL class diagrams", Software Practice and Experience Wiley Journal, 2013

[12] Jordi Cabot, Ernest Teniente, "Incremental Integrity Checking of UMLOCL Conceptual Schemas" , Journal of Systems and Software Volume 82, Issue 9, Pages 1459–1478, 2009

[13] I. Al Agha, O. El-Radie, "Towards Verbalizing SPARQL Queries in Arabic", Engineering, Technology &amp; Applied Science Research, Volume 6, Issue: 2 , Pages: 937-944 , April 2016, https://doi.org/10.48084/etasr.630

[14] A. HAFEEZ, A. MUSAVI and A. REHMAN, "Ontology-Based Verification of UML Class/OCL Model", Mehran University Research Journal of Engineering and Technology, Volume 37, Issue 4, pages 521-534, 2018

[15] A. Hafeez, S. Hyder, A. Rehman and A. Shaikh, Ontology-Based Finite Satisfiability of UML Class Model," IEEE Access, Volume 6, Pages 3040-3050, 2018. Doi: 10.1109/ACCESS.2017.2786781

[16] A. Hafeez, Abdul, S. Hyder Abbas Musavi, A. Rehman, " Ontology-Based Transformation and Verification of UML Class Model," IAJIT, Volume. 17, issue 5, 2020

[17] S. Asadullah, A. Hafeez, E. MA, A. Alghamdi, A. Siddique and B. Shahzad "Ontology-Based Verification of UML Class Model XOR Constraint and Dependency Relationship Constraints.", in intelligent automation and soft computing vol 27 issue 2, 2021

[18] A. Hafeez, A. Wagan , J. Samreen, H. Imtiaz. . " Ontology-Based Transformation and Verification of UML Qualified Association", International Journal of Advanced Trends in Computer Science and Engineering, Volume 10 , issue 1 , Pages 164-167, 2021