Ontology Based Transformation and Verification of UML/OCL Constraints

Dr. Abdul Hafeez Khan

Summary
In the Software Engineering(SE) the graphical models specify architecture, connection and characteristics of the system. New SE methods such as MDA utilize graphical models as a nucleus of all development activities. In the continuity of our research on integration of UML and ontology, this paper presents the transformation and verification of class diagram and Object Constraint Language (OCL) and transformation algorithm from Class model to ontology. The class diagram is transformed into ontology and constraints specified through OCL are transformed into SPARQL.

Keywords:
UML, OCL, Ontology, SPARQL

1. Introduction

Software are everywhere they controls the stock exchange, manage patient record, taking decision etc. However, software failure causes losses either human life and economical. Therefore, it is very important that 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 which drive from the test cases 2. testing activity is executed after completion of code and the cost of bug identification and rectification is much higher in the later phases as compare to earlier phases [1]. Furthermore, nowadays more complex and large software are required in the industry which need 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 are play key roles in the development. MDA uses Unified Modeling Language (UML) as 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 which deal with various facets of software [5][6]. The UML Class diagram is very important part of UML [5][7][8][9]. It represented the real world model through via classes, collaboration, and constraints [10]. Object Constraint Language (OCL) is a constraint speciation language and it small scripts are attached with UML for defining constraints, conditions and business rules [11]. However, MDE approach has some limitations and it also has some limitations such as it is not free from error risk. For Example, model may be created with bugs which can indirectly transported into the code. Verification of model can be a possible solution of the problem.

Current UML Class/OCL models transformation and verification techniques are great and offer good support to verify the correctness of model. But, formal and semi-formal methods are used in them for formalization of model and their notation based on mathematics which is numerously diversified from the UML and very had to understand by the software engineer. On the other side, UML class model and ontology have so many common components and both are developed for representing real-world concepts [12]. The OCL is an important element of UML. it is used to specify constraints which has additional information about, or constraints to, UML model. it can access attribute of objects, operation and navigate object to object through associations and query operation calls. it is specially used for applying integrity constraints on class model however, it is also used with other model of UML such as state chart model.

SPARQL Protocol and RDF Query Language (SPARQL) is a sematic query language for manipulating data from ontology languages it is not only used for query, it is also used for various other functionality e.g. ASK and CONSTRUCT element of SPARQL are used for checking constraint consistency. The ASK construct is used verification of constraints consistency and for inferring new information a CONTRUCT command is used. In the proposed method the OCL constraint are transformed into SPARQL ASK in form of Negation as failure (NAF).

Manuscript received August 5, 2025
Manuscript revised August 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.8.20

2. Proposed Solution

In [12] we have proposed ontology-based approaches for transformation and verification UML class model and presented how OCL can be efficiently represented by the SPARQL. This work presents extension of our ongoing research ontology-based verification [12] [13] [14] of Class diagram and OCL, and presents the detail mapping of different OCL elements into SPARQL. Ontology and class diagram with OCL have various similar elements e.g. classes, collaborations, constraints, instances, and generalization. But, ontology has additional benefit such as it has reasoning capability and on the other side UML does not has appropriate formal foundation and it does not has reasoning ability. The UML model and ontology have a difference: Open World Assumption (OWA) which is supported by ontology and Close Work Assumption (CWA) which is supported by UML. In OWA, currently unknown assumption is treated true, and in CWA unknown assumption is treated false. For supporting CWA in ontology we proposed representation of UML and OCL constraint into the SPARQL negation as failure (NAF) queries.

2.1. Class Model Transformation

In this work we propose UML classes transformation into ontology classes and Class properties are transformed into the Ontology datatype property. Associations are transformed 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 [12]. In this work we presented the transformation algorithm of UML class model to Ontology. According the algorithm, the proposed method will take UML class model in XMI format and read the entire file and transformed the model element according the proposed method.

Algorithm 1: 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.name,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

2.2. OCL to SPARQL Transformation

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

Table 1: Primitive Types

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

The basic 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 number such as find ceil and floor of number, roundof the number, find absolute and etc. the transformation of numeric function into the SPARQL shown in table 3.

Table 2: Operation on primitive type

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

OCL string function are transformed into the SPARQL string function. Table 3 shows the mapping of OCL and SPARQL function. OCL conversion functions convert data from one datatype to other such as integer to real. SPARQL also support conversion function and mapping shown in table 3.

Remaining collection

2.3. Transformation of Collection Operations

OCL provides various operations on the collection types. They are specially designed for projecting new collection from existing ones. this section discus 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 which contains the elements where the Boolean-expression evaluates to true. in SPARQL the select operation is mapped into select query as shown in table 4.
The reject operation is just inverse of the select. it rejects all the elements where the Boolean-expression evaluate true. in SPARQL the are mapped into inverse of select.

2.3.2. Include and Exclude

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

2.3.3. Include All and Exclude all

Include All testify the existence of all elements in the collection if all elements exist in the collection it returns true. The Exclude All work as inverse of Include all. In SPARQL the include All and exclude All are also mapped into *Exits* and *Not Exit* in *Filter* statement but with little extra query parameters as shown in example presented in the table 4.

2.3.4. ForAll , Exit and Collect

The ForAll operation is a wide spread variant which can declare multiple iterator which iterate

Table 3: OCL Function Transformation

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

Table 4: 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.Researchproject)	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 Company inv: self.Faculty->reject(isMarried)->isEmpty()	Inverse of reject

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 return 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

Class diagram with OCL constraints are an essential element of UML. It is used for graphically representing real-world entities. This paper presents a new method for transformation and verification of OCL constraints into SPARQL. OCL and SPARQL have many common elements such as datatypes. 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.

References

[1] KyriakosAnastasakis, BehzadBordbar, Geri Georg and Indrakshi 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] IssaTraore, Demissie B. Aredo "Enhancing Structured Review with Model-Based Verification", IEEE Transactions on Software Engineering, Volume 30 Issue 11 PP. 736 - 753, 2004.

[3] Marcin 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] Kyriakos Anastasakis, Behzad Bordbar, Geri Georg, Indrakshi Ray,"On challenges of model transformation from UML to Alloy", Software & Systems Modeling Volume 9, Issue 1, pp 69-86 Springer, 2010

[5] Marco Cadoli, Diego Calvanese, Giuseppe De Giacomo, Toni 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] Carlos A. Gonzlez, Jordi 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

[8] MIRA BALABAN, AZZAM 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

[9] Alessandro 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

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

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

[12] HAFEEZ, Abdul; ABBAS MUSAVI, Syed Hyder; REHMAN, Aqeel -ur-. Ontology-Based Verification of UML Class/OCL Model. Mehran University Research Journal of Engineering and Technology, [S.l.], v. 37, n. 4, p. 521-534, oct. 2018. ISSN 2413-7219. Available at:

[13] Hafeez, Abdul, S. Hyder Abbas Musavi, A. Rehman and A. Shaikh, "Ontology-Based Finite Satisfiability of UML Class Model," in IEEE Access, vol. 6, pp. 3040-3050, 2018. Doi: 10.1109/ACCESS.2017.2786781

[14] Hafeez, Abdul, S. Hyder Abbas Musavi, A. Rehman, " Ontology-Based Transformation and Verification of UML Class Model," in IAJIT, vol. 17, issue 5, 2020 (Accepted)