International Journal of Future Generation Communication and Networking
Vol. 13, No. 1, (2020), pp. 940-951

Ontological Representation of the UML/OCL Models and Their Verifications

[1]Shikha Singh, [2]Manuj Darbari
[1&2]*Department of Computer Science and Engineering,*
School of Engineering, BBD University,
Lucknow, Uttar Pradesh, India
[1]*shikhasingh2816@gmail.com* , [2] *manujuma@gmail.com*

Abstract

Software model describes constructional layouts with all its detailed components and capabilities. These software models are being used by the latest software development methodologies like as MDE (Model Driven Engineering) as their main components. Most of the modern software development methodologies use these models, as their main components. Using these Software models, the codes can be automatically generated. In case if they have any errors in them, they are directly passed on to the codes. Rectifying such errors at later stages of software development is havoc. Model Verification could be one of the promising solutions to this problem. But again, it has got a variety of features and including them all at one place could be another hurdle of the path. Keeping these problems in view, this work demonstrates an ontological mapping of Software model (UML/OCL in specific) and their verification. Benefits of Ontological portrayal of modeling over OCL have been utilized to perform different levels of code verification. This work mainly focuses on bringing out the best from utilizing the concepts from the two above mentioned approaches, namely UML/OCL and Ontology, to effectively verify software models.

Keywords: *Unified Modeling Language (UML), UML Class diagrams, Object Constraint Language (OCL), Software Model Verification, Ontology*

1. Introduction

Designing is one of the most important part in building of a real time system. A proper and efficient design ensures the seamless behavior of the system in future. There are different approaches for designs. Object-oriented is one of the popular approaches used. One of the popular choice of language by designers for developing real time systems is UML, for enhancing UML notation in modeling of real time application several approaches have been developed. OCL acts as a cherry on the top of UML to ensure system behaves correctly. Software Modeling has always been the first choice for most of the programmer. As they describes almost all the aspects of any software before it actually is made.

In this work mainly a kind of comparison between UML/OCL with ontology is conducted, so that these concepts could be used for model verification effectively. The gap between the two methodologies is closely analyzed. This work focuses especially on bridging the gap between the UML/OCL and Ontology, so as to achieve the maximum benefits of both of them at one place. The UML Class model with OCL (Unified Modeling Language Class with Object Constraint Language) is chosen and a verification strategy based on ontology is introduced onto it. In the planed strategy, a class diagram(UML) is taken and changed into ontology indicated in OWL (Web Ontology Language) and the constraints (OCL) involved is

ISSN: 2233-7857 IJFGCN

converted into SWRL (Semantic Web Rule Language) rules. This task attempts to exhibit that the proposed technique productively covers entire parts of UML Class/OCL model verification and proofed to be beneficial if the two of them are incorporated together.

2. Related Work

Modern software is turning out to be increasingly perplexing and huge. Most of us are dependent on some or the other software to perform our daily routine jobs and hence failure of any such software will definitely turn lives into hell or will lead to various categories of losses be it in terms of lives or money [1]. These failures are basically the results of the errors those were probably not taken care of in the process of making software. The delay in identifying them comes with cost, the sooner we catch them the lesser are the losses [2]. These software need a ton of human endeavors plus time, as software making industries require to release them as right on time as could be expected under the circumstances [3]. Therefore, new methodologies for software development have been presented for quickening of the software making process. One such technique in which software models are taken as a center of software development has been MDE. In MDE, Unified Modeling language is commonly used, which is a graphical modeling language [4]. UML is used in almost all phase of software development activity namely software requirement specification, software analysis, software design, software documentation, and for its code writing as well [5]. For managing different sides of software (structure diagram and behavior diagram), they are being managed by a number of models which UML comprises of [6-7]. The class diagram of UML model is one such significant part and depicts the framework through concepts, relationships, and constraints [8].A class diagram portrays static structure of the given scenario. A class diagram (UML model) together with Object Constraint Language is used for enforcing various limitations, for demonstrating and implementing the integrity constraints and business rules. But MDE approach too isn't likewise liberated from the danger of mistakes. In MDE, software models are created in prior to the software development and at the early stages, it is not an easy task for team developing the software to have a clear vision of all business domain/rules and their constraints. Subsequently, models can get created with mistakes/errors, and these blunders can certainly move into the code [9].

A promising answer for this issue is model Verification. Software Model testing team keeps an eye on the rightness of model and ensures that the model layout is without error. These Software Models are created during the initial phases of software development; therefore, performing error checking is economical when they are conducted in the early phases [10]. Current UML (Class diagram along with OCL) models verification techniques are sound and give extraordinary endeavors to check the accuracy. However, as they are mostly based on formal and semi-formal techniques, therefore, their notations are mostly mathematical [11]. They are altogether not the same as the UML class model and is difficult to understand by the software experts. They likewise have a few lackings, for example, they don’t support basic data types like (string and date), graphical modeling constraints (xor and dependency relationships) and backing of logical consequences. On the other hand, ontology and UML class model has numerous comparable components and both are utilized for modeling real-world concepts [12].

This task introduces a verification method which is based on ontology, especially for UML Class diagram and OCL model and demonstrates how the performance

ISSN: 2233-7857 IJFGCN

efficiency of them are enhanced. At present, the proposed method underpins OCL invariants and doesn't bolster OCL operations. In any case, right now, as the objective documentation is spurred by the way that the present ontology reasoners bolster thinking more than a huge number of ontological things inside a sensible time [13], and all the confirmation viewpoints referenced in existing benchmarks can be viably cultivated through the technique based on ontology. In the proposed work a class diagram is taken and changed into ontology indicated in OWL-DL and OCL constraints into SQWRL (Semantic Query-Enhanced Web Rule Language; articulated squirrel) is a SWRL-based query language that gives SQL-like operators to extricating data from OWL ontologies.

Mahmud [14] brought to light ReSA for the embedded systems. The ReSA is a domain specific language which incorporates assertions of Ontology for spitulating an embedded system. The scalable verifications of Simulink models can be performed via these axioms. Nguyen et. al. [15] manufactured a tool named GUITAR, which accepts the text requirements and converts them into a fortified specification for automated reasoning. Hence, an ontology- based integrated framework can be utilized to generate goal centric verification and case modeling techniques. Corea et. al. [16] proposed an ontology-based technique for affirmation of the Business processes. They modeled ontology reasoning with the logic program as business rules. Liao et al. [17] developed an oncology-based prototype, i.e notification-oriented data intensive EIS (Enterprise Information System). This particular prototype has a new approach for software and hardware characteristics. The crucial disputes faced by IES amidst this fourth industrial revolution are discussed along with their ontology equipped solutions. Another renowned researcher utilized ontology-based verification of UML models.

He et al. [18] used transformation methods to study the behavior of the model. This method includes division of the model into static and dynamic elements. The static ones are used to transform this model into the OWL-DL whereas the dynamics elements transform into the DL-safe rules thereafter being verified by the reasoner. In-depth study of UML versus web ontology language by Dilo et al. [19] fostered various common elements in fields such as classes, the relationships or attributes. They also identified distinguishing patterns of both languages e.g. the relationships in UML class model include association, generalization and its composition, on the other hand and OWL solely has the object property. To conclude, both the languages were proved compatible. Bahaj et. al. [20] discovered a UML class model which serves as an alternative translation into the ontology. It categorizes composition and aggregation as a unique association.

Belghiat et. al. [21] presented statistical graph data inclusive of transformation of class diagram meta-model into the ontology. Parreiras et. al. [22] represented the software models as a combination of UML and ontology and structured MOF (Meta Object Facility) meta-model as their backbone. Exhaustive study provides verification of UML Class/OCL model by formal/ semi-formal notations. The UML components represent graphical elements with less formal notation [23]. UML determines well-versed rules by the meta-model. The OCL however, has no proof. Therefore, different methods (like Z notation, B methods) comprise the major part or early research formalized by well-formedness rules and UML meta-model France et. al. [23] made use of Z notation to formalize the core meta-model thereby translating it into a compositional schema. This schema is chunked into many sub-schemas which relate to each component in the core UML meta-model. Different formal methods correspond to different areas with its own specific strength and thus a single method refrain verification and validation of the UML model. Kim et. al.

ISSN: 2233-7857 IJFGCN

[24] proposed an integrated framework in which verification and validation sustains a formalized environment .The designer can suitably select as per the need. Truong et. al. [25] presented B method for UML class model. In this approach, the model is transformed into the consistency of a verified class model against well-formedness rules. UML well-formedness rules are transformed into the invariant of an abstract machine named B. Semi-formal methods like CSP (Constraint Satisfaction Problem) and Alloy are also greatly used.

Cabot et. al. [26] suggested incremental verification of UML Class/OCL model through CSP. They raised a discussion in verification of constraints, post each structure event (process to Insert Entity, Update Attribute, Delete Entity, and so on) can be costly. They introduced PSEs (Potential Structure Events), which basically are events causing probable Constraint violations. In this method, PSEs are recorded for each integrity constraint following which the instances of entities and relationships are incrementally verified. Complete automation and scalable solutions for restrained verification of UML Class/OCL model is carried out. Moaz et. al. [27] renovates the paradigm of the advanced UML class model (multiple inheritance or interface) into alloy specification. They put forth various analyses such as refinement inspection and intersection of two or more classes.

Shaikh et. al. [28] gave techniques to minimize the complexity of UML Class or OCL model verification by dividing this model into many sub-models, or Model slicing. The useless parts are rejected from the slices (sub-models). This approach reduces verification time of a model with large size but few constraints. However, if there are disjoint slices in the model, less partition will be made, compromising efficiency. The department of UML Class/OCL model verification is broadly focused by Gogolla et al.[29]. They presented detailed guidelines covering almost all areas of UML Class/OCL model verification which act as a functional preface to the new verification system. The core functional requirements are condensed in Table 1. Apart from the ones listed below, there exist a few partly overlapping requirement. The next section elaborates all the functional requirements.

2.1. UML Class/OCL Model Verification Method Requirements

Table1. Requirements Summarization [29]

Req.No	Requirements	Details
Req-1	To Verify Consistency	Global & Local, Complex & Simplex Constraints
Req-2	Requirement Conflicts Detection	Some requirements result into a conflict. This deals with the figuring out of the conflicting requirements.
Req-3	Intensive Arithmetic Calculation	Integer and float number functions and operations support
Req-4	Processing String	String values and string function support
Req-5	Consequences	acquire new knowledge
Req-6	Instances in huge number	10-30 instances per class

ISSN: 2233-7857 IJFGCN

3. Proposed Solution

Requirement:- 1

This requirement guarantees that a non-empty model should be created without infringement of any constraint, irrespective of the type of constraints; it could be global or local and complex or simple. On a single class, the local constraint is applied and on many classes, the global constraints are applied. Easy computations are performed on simple constraint and enormous computations are performed on complex constraint. The verification technique must be able to handle all types of constraints be it the local/global or simple/complex constraints.

Consistency Constraint is the first requirement. Proposed technique shows that verification of complex and global constraints could be easily performed with the help of SQWRL queries. Table 2 shows Product_ID local constraints presented through SQWRL query. Table 2 represents global constraint illustration, where the constraint involves two classes.

Table2: Transformation of OCL Constraint

OCL Constraint	SQWRL
context PRODUCT ProductIDLength: self.wordCount < 12	PRODUCT(?p) ^ has_Product_ID(?p, ?pid) ^ swrlb:matches(?pid, "[0-9]*") ^ Length(?pid) ^ swrlb:lessThan(?pid, 12)->sqwrl.select(?p)

Requirement-2:

Sometimes the requirements of the system being developed result into a conflict and as a result all the conflicting requirements cannot be met. The UML does not provide any modeling of the conflicted requirements. But the author in this research has utilized the SQWRL to query the ontology for requirements conflict diagnosis and proper application or suggestion of the conflict resolution strategy to cater a solution for the arisen conflict.

SWRL is a formal language used to specify rules. The policies of an organization can be framed as rules that the whole system has to abide. There is a whole library of functions supplied with SWRL called SWRLB. SWRL can be used for verification consistency of constraints and a reasoned like the DROOLS Engine can be used for obtaining new assertions from the one which is existing. Fig. 1 represents the verification steps of the discussed method. Initially, the class diagram is transformed into the ontology (OWL-DL) and OCL constraints are transformed into the SWRL. After that, the correctness of the model against the constraints is verified and finally, feedback is returned to the user. The rest of the work in this section represents the conversion of UML Class/OCL model into the ontology and how the proposed strategy understands all the prerequisites referenced in the following segment. In the proposed technique UML classes are changed into metaphysics classes. Each occurrence of a class is considered as a one of a kind element in case of UML, i.e it supports UNA (Unique Name Assumption). Then again, in Ontology two distinct instances can be considered as an equivalent element. Be that as it may, by utilizing distinctive blend of metaphysics develops, the semantic of UNA can be acquired. For Example, in each class, an extra datatype

ISSN: 2233-7857 IJFGCN

property (ID) is joined as a key through HasKey build. Particulars of a class are represented as all different and Classes are declared mutually disjoint.

Declaration (Class (Class Name)) Has Key (Class Name (key Attribute))

Functional Data Property (Key Attribute) Inverse Functional Data Property (Key Attribute)

Identification of Requirements Inconsistencies resulting in the UML model

The process of requirement conflict identification can be performed by a variety of techniques ranging from formal languages to Petri Nets. The formal approaches pose difficulty in implementation of mathematical structures in integrating with the development process.

Ontology Based Technique builds a shared or common ontology that allows identification of conflicts arising from two or more requirements that confront leading to a halt. This feature of ontology is extremely helpful to identify inherent conflicts in the UML design model by converting it to its ontological equivalent.

The authors present a solution to identify inconsistencies and provide a selection for conflict resolution strategies which is nearly absent in the current research. The procedure is depicted in the figure below:

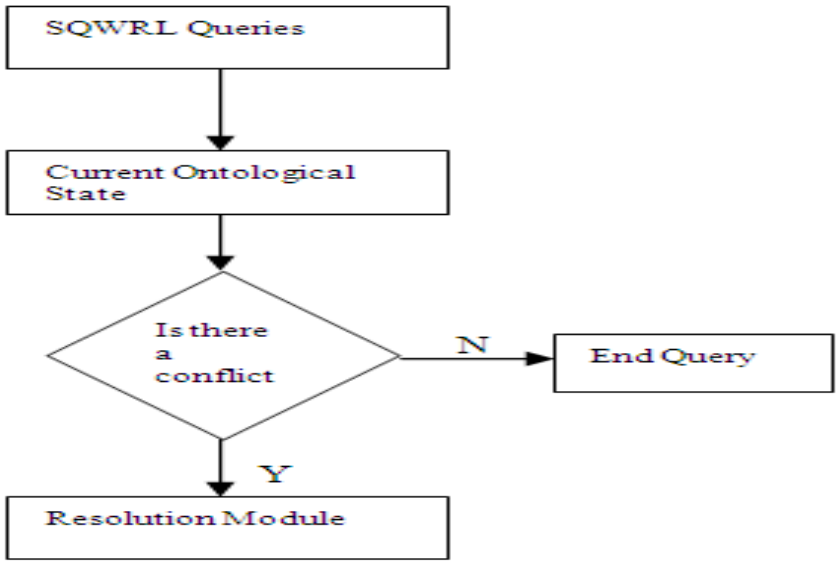

Figure1. Selection of Conflict resolution Strategies

The process uses a common ontology, named PRODUCT_ONT, for instance as a knowledgebase an E-Commerce Application. At definite points in time or when required, specific conflict detection SQWRL queries are run over the ontology to find any potential conflicts between UML Objects. These objects are actually used by the requirements for specificity. The terms associated with the ontology are as given:

ISSN: 2233-7857 IJFGCN

- Objects: Instances of UML Classes.
- Activities: The tasks that may be performed in one region
- Region: Area characterized as similar
- Variables: State like Product_ID, Product_Name, etc

The SQWRL queries are run inside Protégé 5.0.

Conflict recognition has been addressed through various approaches. Koegel states that contention location strategies and methods can be arranged into the accompanying approaches(Koegel, Herrmannsdoerfer, von Wesendonk, and Helming, 2010): State-based methodologies that stores a model's state, and infers dif-ferences by looking at two states (Conradi and Westfechtel, 1998).Change-based methodologies record the framework's progressions while they happen and store them, disposing of the requirement for differencing. Struggle discovery, as for a product situation, alludes to a framework's capacity to perceive a progressing or potential inconsitent framework state and recognize it individually through programming rationale or framework notices. When managing struggle recognition in HBAS, there are various unmistakable procedures and strategies to move toward the subject that normally differ with every framework worldview. For example, with regards to arrangement based frameworks, strategy cover is an ordinarily utilized identification instrument that checks whether another approach covers with some past principle. Rule based discovery is likewise utilized right now frameworks. Aim distinction is another instrument utilized basically in intrigue based frameworks that look for circumstances where clients' expectations are unique. Struggle circumstances can likewise be found through model querying, at the end of the day, querying an information model or metaphysics for conflicting framework

Any potential conflict arising by two or more requirements can be identified by one or both of the conditions met during communication as framed below:

Q1: If a variable, V is preferred by more than one object inside the same Region R.

Q2: If an activity, A1 has pre-condition as a variable V and at the same time V is a post-condition of another activity A2.

The SQWRL query for the above constraints can be written as:

Q1:

Object(?A1)^hasRegion(?A1,?R1)^ hasPreference(?A1, ?var) ^

Object(?A2)^hasRegion(?A2,?R1)^ hasPreference(?A2,?var)->sqwrl: select(?A1,?A2)

Q2.a:

Object(?A) ^hasRegion(?A, ?R1) ^ hasPreCondition(?A, ?pvar) -> sqwrl:select(?A)

b: Object(?A) ^hasRegion(?A, ?R1) ^ hasPostCondition(?A, ?pvar) -> sqwrl:select(?A)

Q1: Selects those objects that belong to the same region, R1 and have preference for the same variable var. This datum cannot be preferred by more than one Object at a time. The query Q1 returns the objects simultaneously accessing this datum. It provides the datum potentially vulnerable for UML objects conflict.

ISSN: 2233-7857 IJFGCN

Q2: Query 2a gives the number of objects that have Pre-Condition as variable 'pvar' depicting the break up DNA size and found in the same region R1. Query 2b gives the number of agents that have the Post-Condition as the same variable 'pvar'. If the count returned by both the queries is greater than or equal to 1, the object A can be identified as having potential conflict.

Ontology, named PRODUCT_ONT that is created through the UML transformation to equivalent ontology for objects association. It provides the necessary vocabulary, relationships, objects and properties of the Requirements' content. A snap of this ontology is given below:

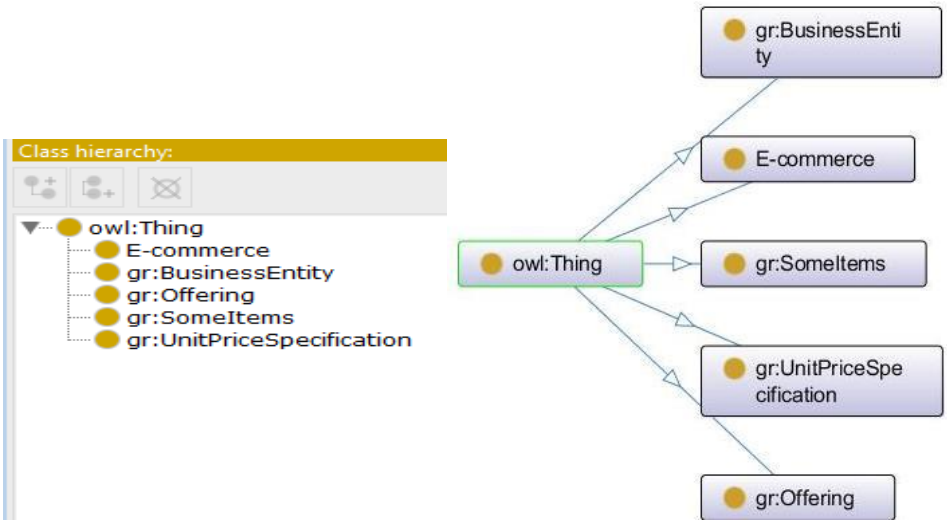

Figure 2. PRODUCT_ONT ontology for storing the details of a E-commerce Application built in Protégé 5.0

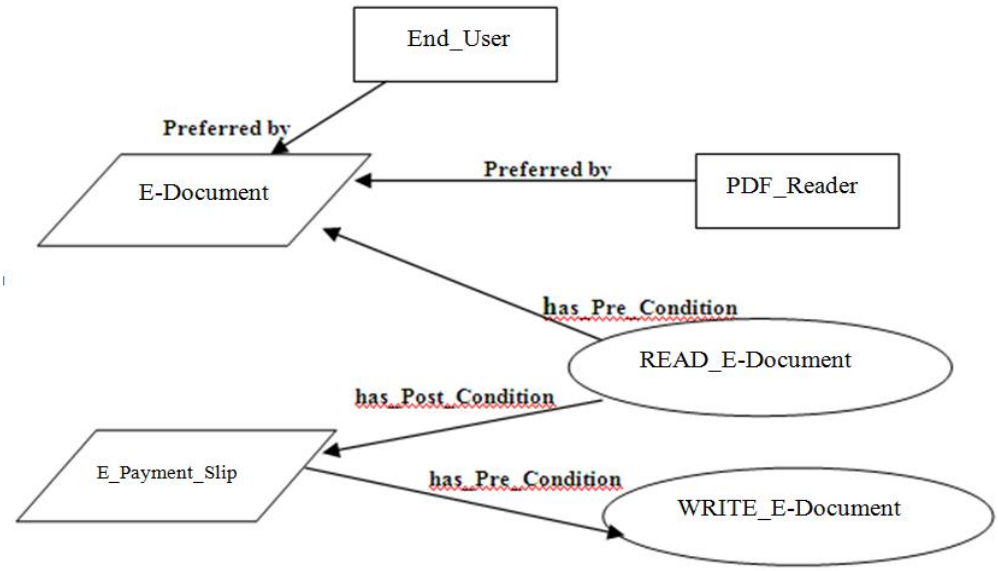

Figure3. A scenario for identification of Requirement Conflict in Ontology which cannot be modeled in UML.

ISSN: 2233-7857 IJFGCN

Requirement-3:

Mathematical calculations on integer and float numbers can be performed with the help of Constraints. Verification techniques should support the complex arithmetic calculation on both integer and float numbers. The third necessity for new UML/OCL model confirmation is the serious help of number juggling calculation. This prerequisite can be effectively acknowledged by the proposed strategy. Ontology underpins every single numeric data types, for example, number, float, decimal. SQWRL bolsters all standard arithmetic operations (+,- ,*,/, and so forth), and numeric functions (floor, ceil, supreme, min, max, and so forth). Requirements with huge arithmetic calculations can be effortlessly indicated through SQWRL. Table 3 shows the example of intensive arithmetic constraint and equivalent SPARQL query. In table 3 we see that the number of inventory items have been fixed by the invariant keyword 'inv' to be 500.

TABLE3. Conversion of constraint with Arithmetic calculation

OCL Constraint	SQWRL Query
context PRODUCT inv PRODUCTPatternLength: self.Count <=50	PRODUCT(?p) ^ hasProduct_InventoryCount(?p,?pic) ^ sqwrl.count(?pic<=500)
	PRODUCT(?p) ^ hasProduct_InventoryCount(?p,?pic) ^ sqwrl.count(?pic<=500)

Requirement-4:

Constraints can likewise perform string calculation and can utilize string functions. Verification strategy should bolster the string processing. The most significant prerequisite for the new UML Class/OCL model verification technique is backing of string on the grounds that current strategies once in a while bolster the limitations which have string activities. Be that as it may, Ontology bolsters string data types and SWRL has many predefined string functions for example substring, stringLength, lowercase, and so forth.

Requirement-5:

Verification method ought to have the option to gather results (new realities) from a lot of declared realities or adages. This is an extremely important and valuable requirement that is best supplied by an ontology. The author has used Protégé 5.2.0 with DROOLS reasoner that comes inbuilt. Its screenshot is given in the figure.

ISSN: 2233-7857 IJFGCN

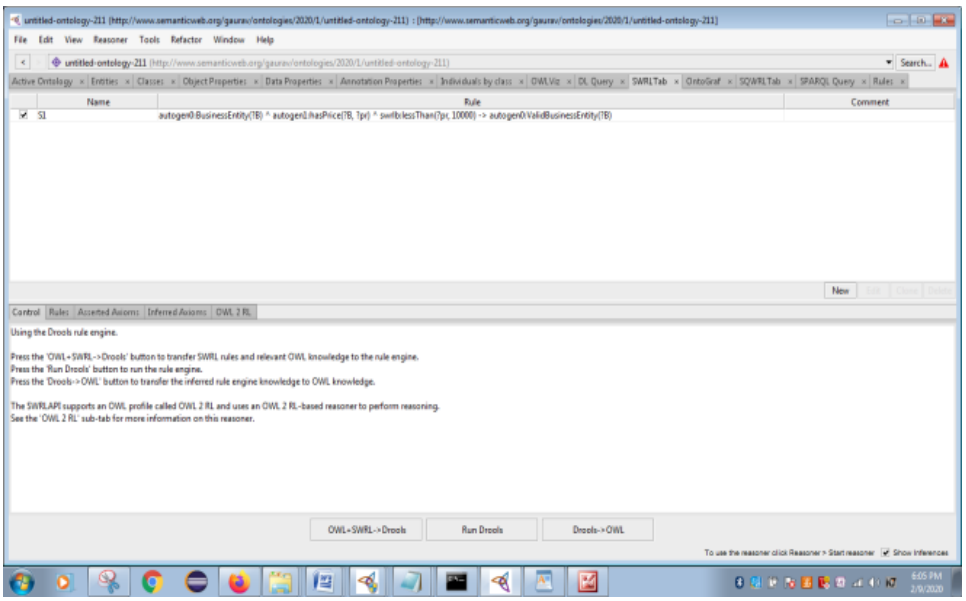

Figure 4. Protégé 5.2.0 with DROOLS reasoner

Requirement-6:

To cover all the aspect of any model, method for verification used must support a huge variety of instances together.Consequently, at any rate 10-30 examples of each class ought to be bolstered by a Verification method. Ontology is utilized for making formal models of real-world scenario and ontology reasoner can perform reasoning over the enormous models. Present day reasoning and a rule engine can process a huge number of ontological things inside a reasonable time.

4. Conclusion

UML Class/OCL model plays an important role showcase the static structure of any software system and is a significant piece of UML. They are the graphical tools, with the help of which we represent any real world scenario with a lot of ease and effectiveness. Various techniques have been used in past for performing model verification for UML Class/OCL models. This paper proposes another technique for check of UML Class/OCL model and points how various highlights of the model can be mapped using the concepts of Ontology with the help of SQWRL and SWRL Queries and Rules Inferencing individually. This work additionally displays how the proposed technique can handle all parts of UML Class/OCL model verification introduced in existing writing, for example, at the time when multiple constraints put together, verifying consistency, and integer calculation, processing string and acquiring new information. At last, executed the proposed strategy and built up a model device to give a proof of idea. The major limitations of current OCL are centered on verification of requirements. The requirements are intended to be without ambiguity, vagueness and obscurity. A method is provided to identify the inconsistencies. This can be achieved with semantic support such that the objects, properties, keywords and relationships including the vocabulary of the messages exchanged between UML objects. This becomes more important in the case if requirements conflict. The author leverages the strength of Ontology designed for

ISSN: 2233-7857 IJFGCN

this purpose in Protégé 5.2.0. In order to verify the requirements, policies are framed that provide the necessary rules. These organizational rules, designed in SWRL are executed on the DROOLS reasoner and rule engine.

References

[1] Baier, C., and Katoen, J.P., "Principles of Model Checking", The MIT Press Cambridge, Massachusetts, USA, 2008.
[2] Anastasakis, K., Bordbar, B., Georg, G., and Ray, I., "UML2Alloy: A Challenging Model Transformation", ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems, Volume 4735, pp. 436-450, Springer, USA, 2007.
[3] Traore, I., and Aredo, D.B., "Enhancing Structured Review with Model-Based Verification", IEEE Transactions on Software Engineering, Volume 30, No. 11, pp. 736 - 753,2004.
[4] Przigoda, N., Filho, J.G., Niemann, P., Wille, R., and Drechsler, R., "Frame Conditions in Symbolic Representations of UML/OCL Models", ACM/IEEE International Conference on Formal Methods and Models for System Design, pp. 65-70, Kanpur, 2016.
[5] Anastasakis, K., Bordbar, B., Georg, G., and Ray, I., "On Challenges of Model Transformation from UML to Alloy", Software & Systems Modeling, Volume 9, No. 1, pp. 69-86, Springer, 2010.
[6] Malgouyres, H., and Motet, G., "A UML Model Consistency Verification Approach Based on Metamodeling Formalization", ACM Symposium on Applied Computing, pp. 1804-1809, 2006.
[7] Cadoli, M., Calvanese, D., Giacomo, G.D., and Mancini, T., "FiniteSatisfiability of UML Class Diagram by Constraint Programming", International Workshop on Description Logics, Volume 104, BC, Canada, 2004.
[8] Shaikh, A., and Wiil, U.K., "A Feedback Technique for Unsatisfiable UMLOCL Class Diagrams", Software Practice and Experience, Volume 44, No. 11, pp. 1379-1393, Wiley, 2013.
[9] Shaikh, A., and Wiil, U.K., "Efficient Verification-Driven Slicing of UML/OCL Class Diagrams", International Journal of Advanced Computer Science and Applications, Volume 7, No. 5, pp. 530-547, UK, 2016.
[10] Encarnaci, M., Barrio-sol, M., Cuesta, C.E., and Fuente, P.D., "UML Automatic Verification Tool with Formal Methods", Volume, 127, pp. 3-16, 2005.
[11] Clark, T., and Evans, A., "Foundations of the Unified Modeling Language", 2nd BCS-FACS Conference on Northern Formal Methods, pp. 1-14, Springer, UK, 1997.
[12] Cali, A., Calvanese, D., Giacomo, G.D., and Lenzerini, M., "A Formal Framework for Reasoning on UML Class Diagram", 13th International Symposium on Foundations of Intelligent Systems, pp. 503-513, France, 2002.
[13] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., and Katz, Y., "Pellet: A Practical OWL-DL Reasoner", Journal of Web Semantics: Science, Services and Agents on the World Wide Web archive, Volume 5, No. 2, pp. 51-53, 2007.
[14] Mahmud, N., "Ontology-Based Analysis and Scalable Model Checking of Embedded Systems Models" M{ä}lardalen University, 2017.
[15] Nguyen, T.H., Grundy, J.C., and Almorsy, M., "OntologyBased Automated Support for Goal-Use Case Model Analysis", Software, Quality Journal, Volume 24, No. 3, pp. 635–673, 2016.
[16] Corea, C., and Delfmann, P., "Detecting Compliance with Business Rules in Ontology-Based Process Modeling", Proceedings of 13th Internationale Tagung Wirtschaftsinformatik, pp. 226-240, 2017.
[17] Liao, Y., Panetto, H., Simão, J.M., and Stadzisz, P.C., "Ontology-Based Model-Driven Patterns for Notification-Oriented Data-Intensive Enterprise Information Systems", Proceedings of 7th International Conference Information Social Technology, pp. 148-153, 2017.
[18] He, H., Wang, Z., Dong, Q., Zhang, W., and Zhu, W., "Ontology-Based Semantic Verification For UML Behavioral Models", International Journal of Software Engineering and Knowledge Engineering Volume 23, No. 2, pp. 117-145, World Scientific, 2013.
[19] Dilo, A., Zlatanova, S., and Oosterom P.V., "Modeling Emergency Response Processes: Comparative Study on OWL and UML", Joint ISCRAM-China and GI4DM Conference, 2008.

ISSN: 2233-7857 IJFGCN

[20] Bahaj, M., and Bakkas, J., "Automatic Conversion Method of Class Diagrams to Ontologies Maintaining Their Semantic Features", International Journal of Soft Computing and Engineering, Volume 2, No. 6, 2013.
[21] Belghiat, A., and Bourahla, M., "UML Class Diagrams to OWL Ontologies: A Graph Transformation Based Approach", International Journal of Computer Applications, Volume 41, No. 3, pp. 41-46, 2012.
[22] Parreiras, F.S., and Staab, S., "Using Ontologies with UML Class-Based Modeling: The TwoUse Approach", Data & Knowledge Engineering, Volume 69, No. 11, pp. 1194-1207, Elsevier, 2009.
[23] France, R., Evans, A., and Lano, K., "The UML as a Formal Modeling Notation",International Conference on UMLBeyond the Notation LNCS 1618, pp. 325-334, Berlin, Germany, 1998.
[24] Kim, S.K., and Carrington, D., "A Formal Mapping between UML Models and Object-Z Specifications", International Conference on Formal Specification and Development in Z and B, Volume 1878, pp. 2-21, UK, 2000.
[25] Truong, N.T., and Souquieres, J., "An Approach for the Verification of UML Models Using B", 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp. 195-202, Czech Republic, 2004.
[26] Cabot, J., and Teniente, E., "Incremental Integrity Checking of UMLOCL Conceptual Schemas", Journal of Systems and Software, Volume 82, No. 9, pp. 1459-1478, Elsevier, Spain, 2009.
[27] Maoz, S., Ringert, J.O., and Rumpe, B., "CD2Alloy Class Diagrams Analysis Using Alloy Revisited Model Driven Engineering Languages and Systems", Lecture Notes in Computer Science Volume 6981, pp. 592-607, Springer, New Zealand, 2011 .
[28] Shaikh, A., and Wiil, U.K., "Evaluation of Tools and Slicing Techniques for EfficientVerification of UMLOCL Class", Advances in Software Engineering Archive, pp. 1-18, Hindawi New York, USA, 2011.
[29] Gogolla, M., Buuttner, F., and Cabot, J., "Initiating a Benchmark for UML and OCL Analysis Tools", Tests and Proofs Lecture Notes in Computer Science, Volume 7942, pp. 115-132, Hungry, 2013.

Authors

Shikha Singh's, received B.E (CSE) degree from Guru Ghasi Das University, Bilaspur and M.Tech degree in IT from Karnataka University, India. She has started her teaching carrier in the year 2004 with BBD group in Lucknow. At present she is working as an Assistant Professor in the Department of Computer Science,(ASET) at Amity University Uttar Pradesh, Lucknow Campus . She has almost 14 years of teaching experience. She is currently pursuing Ph.D degree in the Department of Computer and Information Science, BBD University, Lucknow (U.P). His research interests include Software Engineering, Data Mining.

Dr. Manuj Darbari, is currently working as Professor and HOD, in Department of Information Technology at B.B.D.N.I.T.M (Babu Banarasi Das National Institute Of Technology And Management), Lucknow. He holds a teaching experience of more than 20 years. Prior to his current assignment he has taught for one year in M.N.R.E.C Allahabad as lecturer and 19 years in B.B.D.N.I.T.M. Lucknow in different positions. He has published more than fifty papers in referred international and national journals. He is selected for marquis who's who in science and engineering 2003-2007. His research interests include Software Engineering, Data Mining, Machine Learning , Artificial intelligence to name a few.

ISSN: 2233-7857 IJFGCN