International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:5, 2014

Some Considerations on UML Class Diagram Formalisation Approaches

Abdullah A. H. Alzahrani, Majd Zohri Yafi, Fawaz K. Alarfaj

***Abstract*—**Unified Modelling Language (UML) is a software modelling language that is widely used and accepted. One significant drawback, of which, is that the language lacks formality. This makes carrying out any type of rigorous analysis difficult process. Many researchers attempt to introduce their approaches to formalise UML diagrams. However, it is always hard to decide what language and/or approach to use. Therefore, in this paper, we highlight some of the advantages and disadvantages of number of those approaches. We also try to compare different counterpart approaches. In addition, we draw some guidelines to help in choosing the suitable approach. Special concern is given to the formalisation of the static aspects of UML shown is class diagrams.

***Keywords*—**UML formalisation, Object Constraints Language (OCL), Description Logic (DL), Z language.

I. Introduction

IN software engineering, the use of natural language and graphical notation (such as UML) for specification purposes could lead to incompleteness and lack of precision. Therefore, formalisation could help overcoming such issues. Furthermore, formalisation is an essential requirement for rigorous and automated analysis [1], [2].

UML accommodates a number of diagrams, for instance, Class, Sequence, and other diagrams which are used to represent structural and behavioural aspects of systems. UML suffers from informal representation [3]; therefore, many researchers have introduced their approaches [4]-[12] to formalize those diagrams.

UML diagrams are always present in all stages of software systems development process. Formalising those diagrams is needed for carrying out many rigorous analyses, for example, implementation verification, forward engineering, and design patterns detection. As a result, choosing an approach to formalise those diagram is a hard task. Therefore, this paper highlights some of formalisation approaches and offers some guidelines on choosing the suitable method.

The rest of this paper is organised as follows: In Section II, we define formalisation in general. Consequently, we introduce the main recognized methods for UML formalisation. In Section III, a comparison based on different criteria is curried out. Finally, we sum up the main finding in Section IV.

Abdullah A. H. Alzahrani, Majd Zohri Yafi, and Fawaz K. Alarfaj are PhD students at School of Computer Science and Electronic Engineering, University of Essex, United Kingdom (e-mail: aahalz@essex.ac.uk, mzohri @essex.ac.uk, falarf @essex.ac.uk).

II. UML Formalisation

Formalisation could be defined as any process of conveying ambiguous statements or notions into precise ones [13]. This is usually achieved using variety of mathematical and logical methods (i.e. first order logic FOL, higher order logic HOL, and temporal logic).

Because of the fact that UML lacks precision, this has led to ambiguity and incompleteness in its diagrams. Consequently, rigorous analysis cannot be carried out when such issues exist. This includes consistency verification, traceability, and formal reasoning [5], [14].

In order to overcome the above mentioned issues, researchers have introduced many methodologies to present UML in a formal shape. These methodologies fall into two categories [14]: a) transforming UML to formal models [15], [16], [17], and b) providing abstract syntax and formal semantics for UML diagrams [18]-[20]. However, there has to be a trade-off between these categories as each category emphasizes certain aspects on formalising UML. Many formal languages have been proposed in formalising UML such as Object Constraints Language (OCL), Z, Description Logic (DL), B, PVS, etc. Each of these languages has a number of properties which might not appear in others. As a result,

UML formalisation has been studied and carried out differently according to the researchers' reasons of choosing the formalisation approach. The following section casts light on some of the UML formalisation approaches which are based on different formal languages.

A. UML Formalisation Using OCL

OCL is a formal language used with UML in order to specify constraints and conditions on UML model. It plays an important role in improving precision of the specification of UML [21], [22]. Many researches [4], [11] have used OCL to express UML syntax and semantic.

For instance, in [4], the authors addressed the problem of consistency between the design presented in UML and the implementation. The authors highlight the issue of formalising UML composition relations in terms of lifetime and interoperability, and suggest using OCL to formalise UML composition property as well as other UML properties. They, in addition, claim that the existing modelling tools do deal with such problem. Furthermore, the authors propose an approach to overcome this issue; however, the approach is in its early stage and has not tool support.

The approach suggested in [11] is to transform UML class diagram to another UML class diagram employing only binary associations and OCL constraints. However, translating

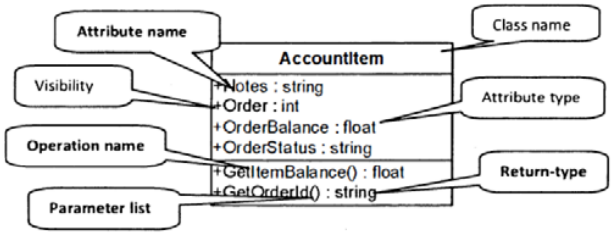
complex associations to binary ones has the issue of losing semantic information.

By the mean of formality, OCL was founded to overcome UML ambiguity. However, OCL itself suffers from a level of ambiguity [23]. As a result, it does not help with formal reasoning and formal proofs [5], [24]. Furthermore, it is difficult to be checked and detected so it raises issues in the development and the maintenance process of the software system. Moreover, OCL seems not to be sufficient when stating complex constraints [5].

B. UML Formalisation Using Description Logic

Description Logic (DL) is one of many approaches to knowledge representation. It is built on a mathematical foundation and supports formal reasoning. A number of description logic languages exist such as AL and ALEN [25]. Many researchers [5], [6], [9], [10], [12] have tried to formalise UML in DL.

Kadir et al. [5] have proposed an approach to formalise UML class diagram using DL. They have assessed their approach to UML formalisation as not satisfactory. This is due to two reasons: (1) many properties are not defined such as dependency, and (2) the formalisation is done manually and there is no tool to automate it. Furthermore, as it can be seen in Fig. 1, employing such approach in the process of the formalization of large-scale software can lead to an unreadable formal specification which consequently can be error prone. Moreover, it would need a solid mathematical foundation to be understood and verified.



$$AccountItem \sqsubseteq \forall P_{GetOrderId()}.string \sqcap (\leq 1 P_{GetOrderId()})$$

$$AccountItem \sqsubseteq \forall Notes.string \sqcap \exists Notes \sqcap (\leq 1 Notes)$$

Fig. 1 An example of formalising one attribute and one operation of a class in DL [5]

Zhihong et al. [9] considered the formalisation of UML class diagram in DL but from a different perspective. They considered the formalisation process itself. They first provided a brief comparison between DL languages in formalising UML. Then, they addressed some concerns when choosing a DL language for formalisation. They suggested some solutions to the problems which occur in formalising UML class diagram. They concluded that formalising UML in DL is a hard and difficult task.

Moreover, [10] have also contributed to the field of UML formalisation in DL. They have chosen DLR description logic language in order to formalise UML class diagram in terms of classes, associations, and constraints. They have shown how to map the constructs of a class diagram to the corresponding formalism in DL. They did not consider those aspects which relate to the implementation (source code) (public, protected and qualifiers for methods and attributes) when formalising the class diagrams. The approach was experimented in FACT [26]. Although, the work is promising, many properties need to be considered to mature this formal framework (such as modelling and reasoning on objects and links) as authors concluded.

Berardi et al. [12] carried out an experimental investigation on the use of the most dominant DL-based reasoning systems to reason about UML class diagrams. The authors illustrate their approach of formalising UML class diagram in DLR description logic language. The approach they used in formalising UML class diagram is similar to [10]; however, the difference is in the choice of the DL language. They reported detailed results about the most popular DL-based reasoning systems namely FACT [26] and RACER [27]. Briefly, the result of the experiment showed that the tested DL-based reasoning systems suffer from critical efficiency issues when dealing with knowledge bases.

DL is a formal approach which can be used to formalise UML diagrams. Its main features are soundness and completeness which are essential in rigorous reasoning. In addition, DL has a number of languages which vary in their features. However, DL cannot formally represent all UML properties such as dependency relation [5], [6], [8]. In addition, when formalising UML diagram in DL, the choice of the DL language needs to be considered carefully [9]. Furthermore, as DL's languages are similar, transferring between DL languages can happen easily which makes it difficult to say which DL language is used if not an explicit statement exists. In conclusion, the process of UML formalisation in DL is hard and needs skills (in DL) since it is still done manually.

C. UML Formalisation Using Z

Z is a specification language strongly typed in mathematics [7]. Since its introduction, it has been an interest for formalisation advocates. This resulted in introducing Object-Z which is an extension of Z. Object-Z was developed to improve Z in many aspects, mainly in structuring and object-oriented representations. This is to enhance effectiveness in specifying large and medium scale software systems. However, it is claimed that a specification in Z is also a specification in Object-Z [28]. UML formalisation using Z and Object-Z has been of interest to many researchers [7], [8]. The following is a review of some works done on this research area.

[7] has proposed a methodology to formalise different kinds of UML diagrams in Z language and represent the result visually using Entity-Relationship (ER) diagram. The authors have clarified their methodology on formalising UML class, Use-Case, and Sequence diagrams. However, the proposed approach has a number of drawbacks. First, the ER diagram can be large when representing industry-scale system. Second, the process of the transformation of Z specification into ER

diagram is not clear. This would introduce more ambiguity than UML diagrams. Finally, the proposed approach has not been implemented in a tool which automates formalisation into Z, ER representation, and consistency verification.

Another work has been carried out in [8] to formalise UML diagrams in Z language. Here, authors introduce their approach by formalising Use-Case diagram, Class diagram, and State Machine diagram, in Z specification language. They have, in addition, implemented a tool to support their approach. The tool checks diagrams and generates automatically the appropriate Z specification if there is no violation of constraints. Additionally, the tool could generate source code for class diagrams in C#, visual basic and JavaScript.

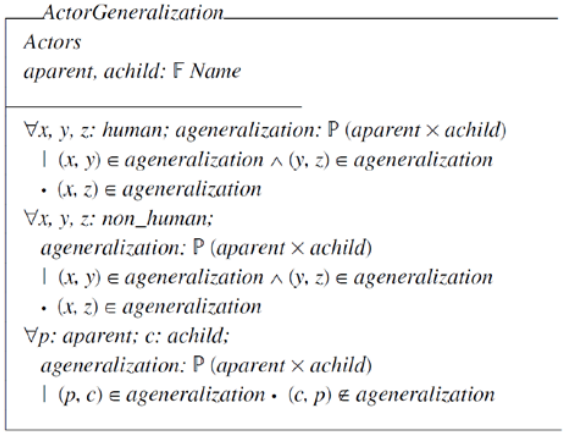


Fig. 2 An example of formalising one generalization relation between two classes in Z [8]

However, the approach uses abstraction concepts, of object-oriented programming, which are not sufficient in describing temporal relations [29]. Moreover, consistency verification is not considered in this approach. Fig. 2 demonstrates one generalization relation in Z. This approach has the limitation when dealing with large-scale software. The reason behind this is the overwhelming specification generated, which makes carrying out any formal reasoning a difficult process.

In conclusion, Z specification is a formal method which can help in formal reasoning and formal proofs. However, Z notations are not graphical and need a solid mathematics background to be formed and understood [7]. In addition, Z lacks of some notations such as Interface. It, also, lacks clarity when the specification is for large scale software systems [28]. Furthermore, a glance at state-of-the-art approaches reveals that there is no tool to support automating the formalisation of UML diagrams in Z and the detection of Z specification from source code.

III. Formalisation Comparison

In this section we compare between the three aforementioned approaches classified as the methodology used. Table I describes the main comparison findings. We used the following criteria: Firstly, we compare between the different approaches of being formal. The second criteria is information lose, which highlight the possibility of losing some information in the process of transferring the UML class diagram into a formal specification. Another criterion is abstraction support, which shows to which extent an approach is responsive to large-scale software representation. The complex constraints support criteria refers to whither an approach be used in formalizing complex constraints. Automation tool illustrate the availability of any tool which automate the formalisation processes using the current approach. Finally, "Math backg" describes the need for a solid mathematical background in the formalisation process or thereafter.

Formalisation approaches vary in order to fulfill the different needs. This makes each approach unique in the way it deals with UML diagrams. The difference does not make one approach outperform the others; however, it shows the main purpose behind the introduction of such an approach. Table I demonstrates the outcome results from comparing a number of UML class diagram formalisation approaches in OCL, DL, and Z.

IV. Conclusion

Formalization tackles ambiguity and incompleteness which exist in UML diagrams. In this paper we have reviewed a number of formalization approaches. The key finding is that UML formalisation approaches produce side-effects which can be divided into the following categories:

1. Information loss: this occurs when the chosen approach cannot formally represent a UML property (such as DL and dependency relation).
2. Manual formalisation: this is encountered in the absence of computer aided software.
3. Requires high mathematical foundation.
4. No tool to support automated consistency verification.

Finally, to the best of our knowledge, there is no such approach which can eliminate these side-effects. This makes those categories commendable guidelines when a new UML formalisation approach is to be introduced.

TABLE I
Methods Comparison Table

	Z	DL	OCL
Formal	Yes	Yes	No
Information lose	Yes	Yes	Yes
Abstraction support	No	No	No
Complex constraints support	Yes	Yes	No
Automation tool support	Yes	No	No
Math backg.	Yes	Yes	No

Acknowledgment

This research was funded in part by Umm Al-Qura University, Management Information Systems section at the University of Essex, and Imam Muhammad bin Saud University. We also would like to thank Dr. Amnon Eden for his outstanding feedback and help.

REFERENCES

[1] W. E. McUmber and B. H. Cheng, "A general framework for formalizing UML with formal languages," in Proceedings of the 23rd international conference on Software engineering, 2001, p. 433442.
[2] B. Potter, D. Till, and J. Sinclair, An introduction to formal specification and Z. Prentice Hall PTR, 1996.
[3] D. Pilone, UML 2.0 pocket reference. O'Reilly Media, 2006.
[4] H. Chavez and W. Shen, "Formalization of uml composition in ocl," in 2012 IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS), 2012, pp. 675–680.
[5] W. Kadir, W. M. Nasir, and R. Mohamad, "Formalization of uml class diagram using description logics," 2010.
[6] B. Zhou, J. Lu, Z. Wang, Y. Zhang, and Z. Miao, "Formalizing fuzzy uml class diagrams with fuzzy description logics," vol. 1, 2009, pp. 171–174.
[7] S. Sengupta and S. Bhattacharya, "Formalization of uml diagrams and their consistency verification: A z notation based approach," in Proceedings of the 1st India software engineering conference, ser. ISEC 08. New York, NY, USA: ACM, 2008, p. 151152.
[8] Mostafa, M. Ismail, H. El-Bolok, and E. Saad, "Toward a formalization of UML2.0 metamodel using z specifications," in Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007, vol. 1, 2007, pp. 694–701.
[9] Z. Zhihong and Z. Mingtian, "Some considerations in formalizing uml class diagrams with description logics," vol. 1, 2003, p. 111115.
[10] A. Cal, D. Calvanese, G. De Giacomo, and M. Lenzerini, "A formal framework for reasoning on uml class diagrams," in Foundations of Intelligent Systems. Springer, 2002, p. 503513.
[11] M. Gogolla and M. Richters, "Expressing uml class diagrams properties with ocl," ser. Lecture Notes in Computer Science, T. Clark and J. Warmer, Eds. Springer Berlin Heidelberg, 2002, pp. 85–114.
[12] D. Berardi, D. Calvanese, and G. De Giacomo, "Reasoning on uml class diagrams using description logic based systems," vol. 44, 2001.
[13] J. B. Wordsworth, Software development with Z: a practical approach to formal methods in software engineering. Addison-Wesley Longman Publishing Co., Inc., 1992.
[14] W. Yan and Y. Du, "Research on reverse engineering from formal models to UML models," 2010, pp. 406–411.
[15] Y. Ledru, "Using jaza to animate RoZ specifications of UML class diagrams," in Software Engineering Workshop, 2006. SEW '06. 30th Annual IEEE/NASA, 2006, pp. 253–262.
[16] M. Bittner and F. Kammuller, "Translating fusion/uml to object-z," 2003, pp. 49–50.
[17] E. Sekerinski and R. Zurob, "Translating statecharts to b," 2002, p. 128144.
[18] X. Li, Z. Liu, and H. Jifeng, "A formal semantics of uml sequence diagram," in Software Engineering Conference, 2004. Proceedings. 2004 Australian, 2004, pp. 168–177.
[19] M. Y. Ng and M. Butler, "Towards formalizing UML state diagrams in CSP," in First International Conference on Software Engineering and Formal Methods, 2003.Proceedings, 2003, pp. 138–147.
[20] S.-K. Kim and C. David, "Formalizing the uml class diagram using object-z." Springer, 1999, p. 8398.
[21] OMG, "Omg object constraint language ocl," Tech. Rep., 2012.
[22] I. Bajwa, B. Bordbar, and M. Lee, "Ocl constraints generation from natural language specification," 2010, pp. 204–13.
[23] S. Flake and W. Mueller, "An ocl extension for real-time constraints," in Object Modeling with the OCL. Springer, 2002, p. 150171.
[24] A. Evans, R. France, K. Lano, and B. Rumpe, "Developing the uml as a formal modelling notation," in Proc. UML98, LNCS, vol. 1618, 1998.
[25] F. Baader, The description logic handbook: theory, implementation, and applications. Cambridge: Cambridge University Press, 2003.
[26] I. Horrocks, U. Sattler, and S. Tobies, "Practical reasoning for expressive description logics," in Logic for Programming and Automated Reasoning, 1999, p. 161180.
[27] V. Haarslev and R. Mller, "Racer system description," in Automated Reasoning. Springer, 2001, p. 701705.
[28] G. Smith, The Object-Z specification language. Citeseer, 2000, vol. 101.
[29] D. Kakollu and B. Chaudhary, "A z-specification of classification and relationships between usecases," in Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08, 2008, pp. 779–784.