OCL Constraints Automatic Generation for UML Class Diagram

Li Tan, Zongyuan Yang and Jinkui Xie
Department of Computer Science and Technology
East China Normal University
Shanghai, China
darkwhite29@gmail.com, {yzyuan, jkxie}@cs.ecnu.edu.cn

Abstract—**As a standard modeling language of software architecture design, UML lacks formal semantics on account of its informal graphical notation. To further provide refined description of UML, OCL is primarily and widely employed. Generally, OCL constraints are written manually, which may cause incorrectness and extra overhead. Therefore, generating OCL constraints template for UML models is a superior solution. The OCL constraints template automatically generated can be used as a reference for software designers. First of all, the significance of automatic generation of OCL constraints was emphasized, and then the application domain of OCL was shown, followed by a lexical analysis of how to extract the target objects in UML models where OCL constraints were needed to build and an algorithm of extraction. Eventually, this extraction algorithm was implemented by Perl. In our way, the overall quality and efficiency of software design is enhanced and thus contributions are made for the automation of Software Engineering.**

Keywords-UML class diagram; XMI; OCL; automatic generation; lexical analysis

I. INTRODUCTION

From the point of view of the software life cycle in Software Engineering, software architecture is the core of the structure and behavior of software. Thus software architecture design is bound to be the core of the software design, and the basis for the subsequent code development as well. The significance of software architecture design is self-evident. As software architecture design itself is a kind of modeling activity, one problem is raised: how to model precisely? That is, the model should be built without ambiguity and lost information. UML (Unified Modeling Language), the standard modeling language of software architecture design, is a kind of graphic notation which hardly has formal semantics, resulting lack of preciseness. Therefore, some formal methods are employed to describe UML mathematically and evaluate it an equivalent formal semantics in formal languages. Among them, OCL (Object Constraint Language) is one major method. As a standard sub-language of UML [1], OCL can make up for what UML cannot present and also provide UML a precise description in semantics. Nevertheless, in general, OCL constraints need to write manually, so the correctness and overhead on personnel are necessary to be considered. Thereby automatically generating OCL constraints template for UML models is badly required. The output may be referred by software designers. Eventually, the overall efficiency of Software Engineering could be improved.

The rest of this paper is organized into four sections. Firstly, OCL formalism and verification in previous work of others will be simply reviewed in Section 2. In Section 3, brief introduction to XMI for UML description and OCL will be given. Then, our approach of lexical analysis based OCL constraints automatic generation is elaborated in Section 4. Finally, the conclusion and future work are put forwarded in Section 5.

II. RELATED WORK

OCL has been widely studied since it was proposed. Some of them are indeed good jobs and the core concepts of them are accepted in a large scale. OCL formalism is one initial research objective on OCL. Reference [2] points out the semantics of OCL constraints is in general not precisely defined and firstly presents a formal semantics for OCL which facilitates verification, validation and simulation of models. Furthermore, OCL is extended with temporal operators to formulate temporal constraints [3]. On the other hand, some work is fulfilled in verifying OCL via size analysis and model checking [4][5]. Since identifying design errors before the implementation stage is cost effective, an automatic verification theory and relevant tool are set up to check the correctness of OCL specifications [4]. Moreover, properties of UML class diagram with its OCL constraints, such as class liveliness and redundancy of integrity constraints are checked by translating both the class diagram and the OCL constraints into a logic representation, and then verifying whether these properties hold [5].

Among the work above, there are few ideas on OCL constraints automatic generation. Therefore, this paper is to work at generating OCL constraints by algorithm we proposed and thus some effective hints of building OCL constraints can be provided for software designers.

III. BACKGROUD KNOWLEDGE

A. UML in an XMI Way

In order to generating OCL constraints for UML, analysis of UML models is necessary. UML itself, as a graphic way, can hardly facilitate the process of analysis, while, UML expressed in a plain text way is easily to parse. Therefore, we need to consider how to transform UML into a format easy to analyze and extract information needed.

In this era of daily changing Internet and knowledge explosion, XMI (XML Metadata Interchange) [6], which attempts to describe system model in the syntax of XML (eXtensible Markup Language) [7], was introduced to express

This work is supported by grants from Natural Science Fund of Shanghai Municipality under Grant No. 09ZR1409500 and Key Project in Basic Research of Science and Technology Commission of Shanghai Municipality under Grant No. 09JC1405000.

and communicate the design of software efficiently on the Internet. From the initial idea of the founder and introducer, XMI is a framework for defining, interchanging, manipulating and integrating XML data and objects, typically used as interchange format for UML models. With the popularity of XML proposed in 1996, XMI which was put forward in 1999 has been also accredited and set up as an international standard [8]. Nowadays, it is getting popular to describe UML, the graphic notation, via XMI, a kind of semi-structured text, and communicate UML through the Web. Hence the drawbacks of UML on data sharing and platform independence are overcome.

Fortunately, there are lots of UML modeling tools which provides easy and automatic transformation from UML diagrams into XMI. Thus, as a unified internal data expression, XMI makes the system design become independent from modeling tools. When XMI is employed, the function of data model saving and loading could be given in a higher abstract level, which is exactly the essence of MDA (Model Driven Architecture) [9].

B. OCL

Based on First Order Logic and Set Theory in Mathematical Logic, OCL is a formal specification language which is used to provide extra precise description and limitation for UML. It firstly appeared in the chapter 7 of the UML specification version 1.3 issued in March, 2000. Afterwards, OCL specification is separate from UML specification and current version is 2.2 [10].

By means of adding OCL constraints to UML models to precisely describe the elements of UML like classes, interfaces, attributes and operations, it is very convenient to show some information hard to express in graphic UML models such as invariables and integrity and consistency constraints among elements. Besides, the ambiguity in UML semantics can be eliminated efficiently. In addition, as a formal specification language, OCL has no side effects. That is, adding OCL constraints into UML models will not change elements as well as relationship among elements in the model. In a word, building appropriate OCL constraints into UML models will definitely do good to the optimization of software design.

The general format of OCL constraints is as follows:

- Start with a context identifier “context”, followed by the name of element which OCL constrains.
- Under the line of “context”, invariables would be given, if necessary.
- Under the line of “context”, pre-conditions and/or post-conditions would be given, if necessary.

The elements constrained by OCL must satisfy all the conditions of invariables, pre-conditions and/or post-conditions when situation changes, such as operation executed and object founded. Thus, the attributes and/or operations of elements are well constrained by OCL constraints. According to the definition above, a simple piece of OCL constraints statements fragment are given as follows:

```
context Faculty
inv: facultyID >= 0 and facultyID <= 999999
inv: courseNum.oclIsTypeOf(int)
context Faculty::deleteID()
post: facultyID = null
```

IV. Lexical Analysis Based OCL Constraints Automatic Generation

After showing evidence of significance and feasibility of OCL constraints automatic generation, this section is to make it real. The main idea of this section is using lexical analysis to find the element of UML where OCL constraints need to build, and then an OCL constraints target extraction algorithm is given, so finally an OCL constraints template for UML models can be generated automatically.

Our OCL constraints template is considered as a good reference for software designers. On one hand, this template could remind software designers of building OCL constraints for some important elements, in case that they neglect. On the other hand, this template could help software designers improve the OCL constraints they have written by themselves to achieve a more complete OCL constraints of higher quality.

A. The Application Domain of OCL Constraints

According to the introduction of OCL in Section 3, it is necessary to consider the application domain of OCL constraint in this paper. Generally speaking, the application targets of OCL are classes, interfaces, attributes and operations. These elements are widely used in UML class diagram. As is known to all, UML class diagram is the core structure of whole system and the foundation of other UML diagrams. It is reasonable and necessary to provide UML class diagram some OCL constraints where needed. For the simplicity, in this paper we only consider generating OCL constraints template for UML class diagram.

The elements of UML class diagram for which we generate OCL constraints include: attributes and operations. As for domain-related OCL constraints, that is, OCL constraints which need to understand the semantics of UML class diagram before building them, are not discussed here.

B. Extracting OCL Constraints Targets

In the process of generating OCL constraints automatically, the key step is to find targets, that is, the elements for which need to build OCL constraints. The essence of solution of this problem is refined as how to locate the OCL constraints targets. We use the lexical analysis technology of compiling. To sum up, the OCL constraints targets are chosen by means of lexically analyzing the XMI file which contains all information of UML models when traversing the DOM (Document Object Model) [11] of XMI file, and then some corresponding OCL constraints are added for the chosen targets. Eventually, a whole OCL constraints template for one UML class diagram is generated after traversing.

1) Definition of OCL Constraints Targets

Based on the discussion above, several characteristics of targets for which need to build OCL constraints are shown as follows:

- The Identifier Attributes of a Class

In general, as identifiers, there are two basic attributes defined in all the classes: “name” and “ID” (attributes name may be different slightly but are based on the two words). Thus, attributes “name” and “ID” (or in the similar name) can be considered as targets for which need to build OCL constraints.

- The “New” and “Delete” Operations of the Identifier Attributes of a Class

In UML class diagram, if there are “name” and “ID”, two basic attributes defined as identifiers in one class. The “New” and “Delete” operations are indispensable (“Modify” operation could be implemented by “New” first and then “Delete”). Thus, operations “New” and “Delete” (or in the similar name) of attributes as identifiers can be considered as targets for which need to build OCL constraints.

- The “Add” and “Remove” Operations of a Class as a Set/Container

As for a class itself as a set or container, like a “transcript” class, its object, that is, a piece of transcript, can own lots of entries. Each of the entries is a score of one subject tested. It is evident that the “Add” and “Remove” operations of entries are indispensable as well (“Edit” operation could be implemented by “Add” first and then “Remove”). Thus, operations “Add” and “Remove” (or in the similar name) of a class as a set or container can be considered as targets for which need to build OCL constraints.

2) Algorithm of Extracting OCL Constraints Targets

Based on the previous definition of OCL constraints targets, an algorithm of key steps for extracting these OCL constraints targets is summarized below.

Algorithm 1: *Extracting OCL Constraints Targets*

```
Procedure abstract($O_i$)
 1: for (each attribute in $CD_j$) do
 2:   if (attribute.id match $pattern_{a1}$) then
 3:     addOCL(idOCL);
 4:   end if
 5:   if (attribute.name match $pattern_{a2}$) then
 6:     addOCL(nameOCL);
 7:   end if
 8: end for
 9: for (each operation in CDj) do
10:   if (operation.name match $pattern_{o1}$) then
11:     addOCL(newOprOCL);
12:   end if
13:   if (operation.name match $pattern_{o2}$) then
14:     addOCL(deleteOprOCL);
15:   end if
16:   if (operation.name match $pattern_{o3}$) then
17:     addOCL(addEntryOprOCL);
18:   end if
19:   if (operation.name match $pattern_{o4}$) then
20:     addOCL(removeEntryOprOCL);
21:   end if
22: end for
```

This algorithm is implemented by Perl, a script language with powerful and efficient capability of text parsing and operating [12]. Due to page limitation, code will not be listed here.

C. Case Study

A specific case in an academic system is given below to demonstrate the effect of the algorithm of extracting OCL constraints targets. First, a UML class diagram is designed as follows:

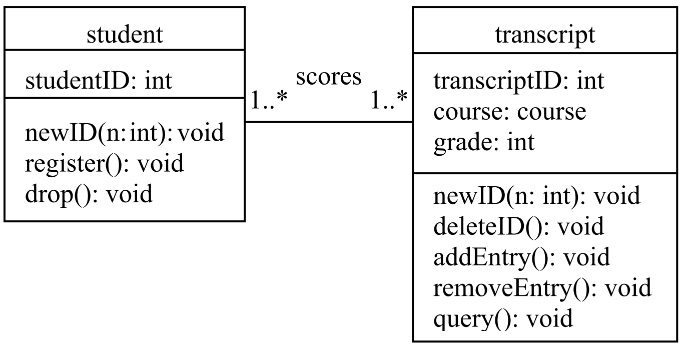

Figure 1. UML class diagram of an academic system.

With regard to this UML class diagram, we generate an OCL constraints template automatically based on the algorithm of extracting OCL constraints targets proposed above. A piece of OCL constraints statements sample generated is shown below.

```
context student
inv: studentID >= 0 and studentID <=
99999999
context student::newID(n:int)
pre: n >= 0 and n <= 99999999
post: studentID = n
context student::deleteID()
post: studentID = null
context transcript
inv: transcriptID >= 0 and transcriptID <=
99999999
context transcript::newID(n:int)
pre: n >= 0 and n <= 99999999
post: transcriptID = n
context transcript::deleteID()
post: transcriptID = null
context transcript::addEntry(e:string)
pre: scores->excludes(e)
post: scores = scores@pre->including(e)
context transcript::removeEntry(e:string)
pre: scores->includes(e)
post: scores = scores@pre->excluding(e)
```

In it, 3 OCL constraints are generated for the class “student”. Firstly, the attribute “studentID” is restricted as an integer between 0 and 99999999 by defining an invariable regarding “studentID”; Secondly, the pre-conditions and post-conditions of operation “newID” are generated to define a range/effect of “newID”; Similarly, the post-conditions of operation “deleteID” are given as well.

Besides, 5 OCL constraints are generated for the class “transcript”. Firstly, the attribute “transcriptID” is restricted as an integer between 0 and 99999999 by defining an invariable regarding “transcriptID”; Secondly, the pre-conditions and post-conditions of operation “newID” are generated to define a range/effect of “newID”; Similarly, the post-conditions of operation “deleteID” are given as well; Finally, since “transcript” is set/container class, the pre-conditions and post-conditions of operation “addEntry” and “removeEntry” are also taken into consideration, respectively.

V. Conclusion and Future Work

OCL is a primary formal specification language for UML precise description and restriction. With the help of OCL, some information unable to express in UML are shown successfully, such as invariables and constraints among UML elements. OCL is already considered as the standard sub-language of UML for some auxiliary work in UML which is indispensable for completeness and preciseness of modeling. Nevertheless, OCL is generally written manually, resulting errors and extra overhead. Therefore, automatically generating OCL constraints for UML models as a template for software designers’ information should be a superior solution. Nowadays, this kind of work has not done very much. Theoretically, OCL constraints automatic generation is an excellent way to enhance the overall efficiency of software design. Thus, the process of Software Engineering is optimized as well.

This paper proposes a novel algorithm of OCL constraints automatic generation typically for UML class diagram. We define the features of several kinds of OCL constraints targets to provide the pathway of how to locate them, and then use lexical analysis technology to extract OCL constraints targets from XMI file which contains all information in a UML class diagram. In this way, corresponding OCL constraints are added for these targets extracted. Finally, a case of academic system proves the feasibility of our method in practice.

There is still much work for further research of OCL constraints automatic generation. First, the application domain should be extended. More OCL constraints targets in UML class diagram and other UML diagrams need to consider, like guard in UML state diagram. Secondly, the algorithm of extracting OCL constraints targets should be optimized on the aspect of efficiency, or the framework of algorithm can just be changed for a more efficient way. As for tool implementation, a prototype system for this paper has been built in a Web-based and user-friendly way.

References

[1] Object Management Group (OMG), UML Specification, Version 1.3, 2000.

[2] M. Richters and M. Gogolla, “On Formalizing the UML Object Constraint Language OCL,” the 17th International Conference on Conceptual Modeling (ER’98), Springer LNCS, vol. 1507, pp. 449–464, 1998.

[3] P. Ziemann and M. Gogolla, “OCL Extended with Temporal Logic,” Springer LNCS, vol. 2890, pp. 351–357, 2003.

[4] F. Yu, T. Bultan, and E. Peterson, “Automated Size Analysis for OCL,” the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'07), Dubrovnik, Croatia, pp. 331–340, 2007.

[5] A. Queralt and E. Teniente, “Reasoning on UML Class Diagrams with OCL Constraints,” the 25th International Conference on Conceptual Modeling (ER’06), Springer LNCS, vol. 4205, pp. 497–512, 2006.

[6] Object Management Group (OMG), MOF 2.0/XML Metadata Interchange Mapping Specification, Version 2.1.1, 2007.

[7] World Wide Web Consortium (W3C), Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008.

[8] International Organization for Standardization (ISO), XML Metadata Interchange Specification, Version 2.0.1, 2005.

[9] J. D. Poole, “Model-Driven Architecture: Vision, Standards And Emerging Technologies,” Position Paper, European Conference on Object-Oriented Programming Workshop on Metamodeling and Adaptive Object Models, 2001.

[10] Object Management Group (OMG), Object Constraint Language Specification, Version 2.2, 2010.

[11] L. H. Philippe, “The W3C Document Object Model (DOM),” World Wide Web Consortium (W3C), http://www.w3.org/2002/07/26-dom-article, 2002.

[12] P. M. Nugues, An Introduction to Language Processing with Perl and Prolog, Springer-Verlag, 2006.