A TOOL FOR CHECKING CORRECTNESS OF DESIGN DIAGRAMS IN UML

Ivo Dlouhy
Master Degree Programme (4), FIT BUT
E-mail: xdlouh05@stud.fit.vutbr.cz

Supervised by: Marek Rychly
E-mail: rychly@fit.vutbr.cz

Abstract: This article describes a tool for checking correctness of design diagrams in UML. Correctness is checked by matching incorrectness and bad practice patterns on the UML model. Patterns are stored in a XML database file. UML Class Diagram is loaded from XML Metadata Interchange format or user interface. Patterns are detected using QVTr transformations from UML to result model. Command line and Visual Paradigm modelling software plugin interfaces use the tool library.

Keywords: uml correctness, design diagrams, visual paradigm, incorrectness patterns, uml, qvtr

1 INTRODUCTION

Software design and modeling plays a key role in a software life cycle. Recent modeling standards and languages are well defined, but they still need to be general and flexible enough to allow user to model every imaginable scenario. This ambiguity can cause incorrect use of the language. The errors made in design can heavily influence the result product and therefore it is desirable to eliminate them.

This article investigates the correctness of the UML modeling language, namely class design diagram denoting classes and their associations. There are several types of associations with different semantics and different rules to use. Most of these rules are defined by the Unified Modeling Language (UML) standard introduced by the Object Modelling Group (OMG). However there might be a chance for incorrect usage within the standard. Therefore this work is aimed at creating a tool to support software designers.

2 UML CORRECTNESS DESCRIPTION

There are a few approaches to UML correctness evaluation. UML modeling software often uses the integrated evaluation, but in fact it mostly performs just a basic check. None of the software alternatives apply advanced UML correctness evaluation to analyze the diagram for advanced incorrectness patterns. The term UML correctness is not precisely specified. Object Modelling Group (OMG) introduced the UML standard that defines the language and fundamental set of rules [1]. Other available sources usually publish additional rules in form of incorrectness patterns or anti-patterns [2]. These patterns refer to wrong UML usage or a bad practice that can lead to the incorrect final diagram.

Under the term 'incorrectness pattern' we understand a set of metadata describing pattern identification — a unique identifier to distinguish it from other patterns, a description of the general problem as well as detection pattern represented formally and a solution how to resolve the problem that can be also formally represented if available respectively.

The presented project investigates the usage of reasonable set of patterns to ensure UML document correctness with. The proceedings is as follows: either the pattern detected in the UML is considered correct or the user will be notified about the problem and provided with a guide on how to resolve it.

The outcome of this project is a tool for Checking Correctness of Design Diagrams in UML. The Core of the tool is the pattern matching component. A few approaches to pattern matching were considered, but the most suitable is the application of Queries/Views/Transformations (QVT) as referred in [3]. QVT is a standard for model transformation defined by the OMG covering 3 languages. The most useful of them is the QVT Relations language (called in short QVTr) that offers formal definition of model-to-model transformations. Transformation is expressed by a set of relations between the models. Each Relation can be defined by two or more domains or patterns. This work takes advantage of the analogy between incorrectness and QVTr patterns. Transformations from QVTr are applied to solve the pattern matching as published in [3, 4]. All available incorrectness patterns can be formally specified in QVTr and executed in the UML model. Afterwards the UML model is transformed to Result model that describes pattern matching results.

A demonstration of the pattern matching proces is shown in the figure 1. Model situation is following: a designer developes a complex UML class diagram with generalization hierarchy of Shapes. Afterwards he adds the AlertSign class thus creating a generalization cycle. Incorrectness pattern is matched on these classes and transformed to the Result model that contains the pattern occurence.

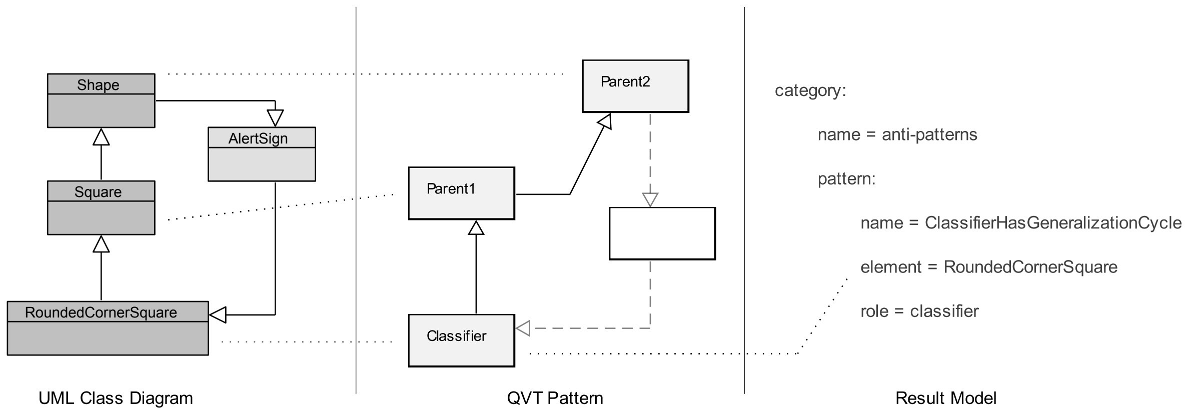

Figure 1: Demonstration of incorrectness pattern matching

3 REALIZATION

The UML correctness checking tool is build upon a library. The implemented Library incorporates the main functionality and allows for the tool to have more interfaces. Basic interface is formed by a command line that accepts input UML model in XMI format and produces results of the correctness check. The interface mentioned above is suitable for correctness checking automation as well as testing. In addition to that an advanced interface is introduced, that is represented by a plugin module to Visual Paradigm modelling software. The plugin enables the user to access the correctness checking functionality directly from Visual Paradigm user interface. The current diagram is checked for correctness and the results are presented in the output table that contains a description of the detected problem and an advice how to fix it.

A set of incorrectness patterns is stored in the pattern database. The database is constituted by a XML file that is distributed with the tool. It is also available online so that the tool can download recent updates of the pattern database. The pattern record consists of metadata and the pattern itself. The metadata contain name and ID that are used for pattern identification both in the transformation and in the result set. The pattern is formed by the QVTr transformation used during the detection and a textual description of the problem.

The library requires the UML class diagram represented in the library data model. It also needs a pattern database to operate. The output is a correctness check result set specified by the library data model. The interface for the library is well documented during the development in order to allow integration in any other application. Simplicity of this concept makes this 'UML pattern matcing platform' easily extensible.

In the command line application, the UML diagram stored in the XMI file is validated against the XML Schema Definition (XSD). The valid XMI input is parsed, converted to the data model and consequently passed to the library. The plugin interface in Visual Paradigm allows the tool to directly convert the diagram to the data model. After the model and the pattern database are loaded into the library, the QVTr transformations are run. The project utilizes Medini QVT engine to execute QVT transformations. Thus the input model is transformed into the result set which is converted to library result data model and returned to the user interface.

4 CONCLUSION

This article describes the tool for checking correctness of design diagrams in UML. It discusses a method of analyzing the UML diagram by a set of incorrectness and bad practice patterns. These patterns are formally specified using QVTr transformations and saved together with metadata. The transformations from database are applied on the input XMI file in the command line application or in the current diagram in the plugin for Visual Paradigm design software. The results of the correctness analysis including a hint on how to resolve the issue are presented to the user.

It has been found out that the correctness check using QVTr and incorrectness pattern matching can be used to support the software designer in the process of creating an UML class diagram. The tool helps to avoid critical errors in the model, therefore it can be also used for educational purposes. The pattern database is open and documented, thus enables the user to extend it with his own patterns and tailor the tool to his own needs. Additionaly the core components are enclosed in the library so that the tool can be easily adapted to a variety of other user interfaces or plugins and creates a platform for correctness check via pattern matching.

The next milestone of the project is to finish the implementation and thoroughly test the pattern detection. Another important target of the project is to build up sufficiently comprehensive database of incorrectness and bad practice patterns. The size of the pattern database will be the contribution of the project.

ACKNOWLEDGEMENT

This work was partially supported by the BUT FIT grant FIT-S-14-2299.

REFERENCES

[1] R. Miles, K. Hamilton. Learning UML 2.0. O'Reilly Media, 2006. ISBN 978-0-596-00982-3.

[2] M. Balaban, Maraee Azzam, Sturm Arnon. Management of Correctness Problems in UML Class Diagrams Towards a Pattern-Based Approach. International Journal of Information System Modeling and Design, 1(4), 2010.

[3] M. Elaasar, B. Lionel, Y. Labiche. Specification and Detection of Modeling Patterns: an Approach based on QVT. Technical report, 2010.

[4] M. Elaasar, B. Lionel, Y. Labiche. Domain-Specific Model Verification with QVT. In Modelling Foundations and Applications, vol. 6698 of Lecture Notes in Computer Science, Springer, 2011.