PRES LUNAM
Ecole Doctorale STIM
Sciences et Technologies de l'Information et Mathématiques

Spécialité : Informatique
Laboratoire : EMN-LINA
Equipe : AtlanMod

PRAGMATIC MODEL VERIFICATION

González, Carlos A.
Mél : carlos.gonzalez@mines-nantes.fr

Abstract: Model-driven Engineering (MDE) is a popular approach to the development of software which promotes the use of models as first-class citizens in the software development process. In a MDE-based software development process, software is developed by creating models to be successively transformed into another models and eventually into the software source code. However, the growing popularity of MDE has brought about an increase in the complexity of models and model transformations, thus risking both, the reliability of these software development processes and the soundness of the resulting software. Traditionally, ensuring software correctness and absence of errors has been addressed by means of software verification approaches, based on the utilization of formal analysis techniques, and software testing approaches. This work addresses the problem of ensuring quality and reliability in MDE-based software development processes by presenting two tools: a model verification tool to ensure model correctness and a model generation framework for testing model transformations. These two tools show how software verification and software testing techniques can be adapted and combined to preserve the quality and reliability of MDE-based software development processes.

Keywords: *MDE, Model Verification, Model Transformation, Testing, Testing of Model Transformations*

Collaborations : École des Mines de Nantes

1 Introduction

Model-driven Engineering (MDE) is an approach to the development of software in which models play a key role. In a MDE-based software development process, the software under development is not coded by hand, but by designing and creating a number of models that, by means of (semi)automatic model transformations, are successively transformed into another models and eventually into the code comprising the new software system. With the past of time, MDE has gained popularity, which has led to an increase in the complexity of models and model transformations. Therefore, the creation of these elements has turned into an error-prone task that risks both, the reliability of MDE-based software development processes and the soundness of the resulting software. For this reason, there is a great need of mechanisms to ensure quality and the absence of errors in models and model transformations.

Ensuring software correctness is not a new challenge, though. On the contrary, the research community has made significant efforts on this field throughout the years, efforts that have brought about two important trends to try to address the problem: software verification and software testing. Software verification comprises those approaches based on the use of formal analysis techniques to prove software correctness. Software testing, on the other hand, usually refers to those approaches that try to find errors in software by systematically running the software with a set of inputs for which the expected output is known, and then comparing the actual outcome with the expected one.

The objective of this work is to contribute new mechanisms and tools to ensure quality and absence of errors in models and model transformations. In this regard, two tools are presented: a model verification tool called EMFtoCSP, and a model generation framework for testing model transformations called ATLTest. EMFtoCSP is aimed at ensuring model correctness by means of the utilization of formal analysis techniques. ATLTest is aimed at generating an exhaustive set of models to be used as an input at the time of testing a model transformation. These two tools also demonstrate how verification and testing approaches can be combined to ensure the reliability of MDE-based software development processes.

The rest of the paper is organized as follows: Section 2 introduces the model verification tool EMFtoCSP. Section 3 introduces the test model generation tool ATLTest. Section 4 reviews the related work and, finally, Section 5 draws some conclusions and some future challenges that should be addressed.

2 Model Verification: EMFtoCSP

Formal verification of models, at least when it comes to static models like UML[1] (Unified Modeling Language) class diagrams, refers to the ability of the model under analysis to satisfy one or more correctness properties. These properties express certain characteristics that the model must feature in order to be considered correct. It is typical for this type of tools to translate the model along with the correctness properties to be checked into some kind of formalism, which is then exploited during the reasoning process, to determine whether the model satisfies the properties under scrutiny. This reasoning process is conducted with the help of tools specialized in reasoning over the chosen formalism.

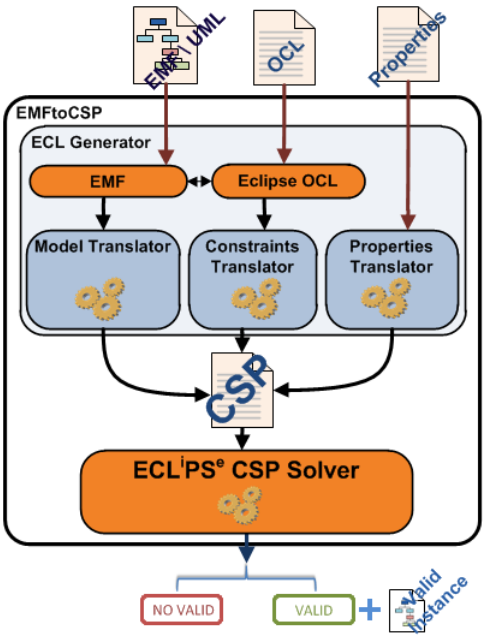

Figure 1: EMFtoCSP architecure

EMFtoCSP[2] is an Eclipse[3] integrated tool for the automatic verification of static models annotated with OCL[4] (Object Constraint Language) constraints. EMFtoCSP is an evolution of a previous tool called UMLtoCSP [1] devoted to the verification of UML class diagrams. In this sense, EMFtoCSP expands UMLtoCSP by also supporting EMF[5] models. EMF is the de facto standard modeling framework in the industry. In EMFtoCSP (Fig. 1), the input model along with its constraints and the correctness properties to be verified are translated into a constraint satisfaction problem (CSP), which it is then fed to a solver called ECL^{i}PS^e Constraint Programming System[6] for its resolution. Since the CSP is built such that it has a solution if and only if the model plus its constraints satisfy the correctness properties, if the solver finds a solution for the CSP, then it can be concluded that the model satisfies the selected correctness properties. In this case, the tool provides a valid instance of the input model to certify it. In the case the solver does not find a solution for the CSP then nothing can be concluded. This is because EMFtoCSP follows a bounded verification approach, consisting in limiting the search space to keep the reasoning process decidable. As of know, EMFtoCSP supports the verification of the following correctness properties: strong satisfiability, weak satisfiability, lack of constraint subsumptions and lack of constraint redundancies. For a more comprehensive description of the tool and the correctness properties supported, the reader is referred to the work the author has already published [2].

[1] http://www.omg.org/spec/UML/
[2] http://code.google.com/a/eclipselabs.org/p/emftocsp/
[3] http://www.eclipse.org/
[4] http://www.omg.org/spec/OCL/
[5] http://www.eclipse.org/modeling/emf/
[6] http://eclipseclp.org

3 Testing of Model Transformations: ATLTest

The methodology to test a model transformation is essentially the same as for program testing, that is, creating input test cases, running the software with the test cases, and finally, comparing the results yielded with the expected ones to determine whether errors came up or not. However, compared to program testing, model transformation testing must face an additional challenge: the complex nature of model transformation inputs and outputs. Models can be large structures and must conform to a meta-model (possibly extended with OCL well-formedness rules) thus making even harder the generation of test models and the analysis of the results.

It is generally accepted that the more input test models are created and the more time is spent exercising the model transformation, the higher is the probability of finding errors. However, since the number of input test models that can be created to test a model transformation can be potentially infinite, it is necessary to establish some strategy to carry out testing in an effective way. Two of the most prevalent strategies are black-box testing and white-box testing. The main difference betweeen them is that in black-box testing only model transformation specification is taken into account at the time of designing test models, whereas in white-box testing test models are created out of the analysis of model transformation internals.

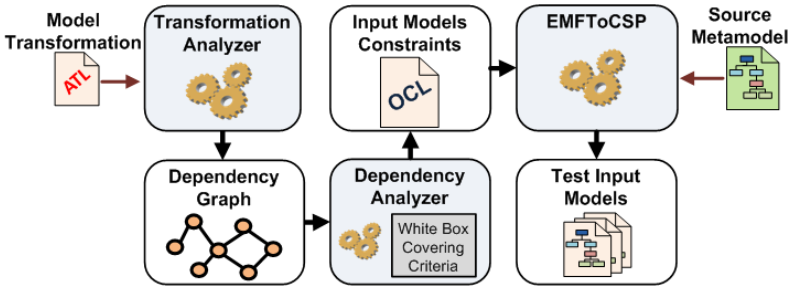

Figure 2: ATLTest architecure

ATLTest is a white-box test model generation framework for ATL model transformations. ATL[7] is one of the most popular transformation languages both in academia and industry. The generation of test models with ATLTest is a three-step process (Fig. 2). First, the ATL transformation is analyzed and a graph called "dependency graph" is produced. The dependency graph represents groups of interrelated conditions expressed in the OCL, that must be hold (totally or partially) by the test input models. Having created the dependency graph, the second step is to traverse it a number of times which, as for traditional testing approaches, is determined by some coverage criteria. Traversing the dependency graph implies setting truth values for the different conditions in the graph and, therefore, each traversal will yield a set of constraints that symbolizes a family of relevant test cases for the transformation. Finally, in the third stage, the test input models are created by computing models conforming to the source metamodel and satisfying the constraints for the test case by means of the tool EMFtoCSP presented before. For more information on ATLTest, the reader is referred to the work the author has already published [3].

4 Related Work

Formal verification of models is a complex problem, undecidable in general, specially when models are used in combination with languages like the OCL to represent certain aspects of the system in further detail. For this reason, some verification tools provide limited or no support to the OCL [4, 5]. Among the tools that, like EMFtoCSP, support the OCL completely, some of them are user-assisted, like [6] which requires user intervention to steer the reasoning process, but in general the majority of them are automatic, like [7], although at the expense of limiting the search space to keep the reasoning process decidable. In this last case, as it also happens with EMFtoCSP, the biggest trade-off is that these tools are not complete, that is results are only conclusive when a valid instance of the model is found.

The creation of an adequate set of test models is a key task at the time of testing a model transformation. Currently, the majority of test model generation approaches are of the black-box type that is, the generation of test models is based on the analysis of the model transformation specification. Some

[7] http://www.eclipse.org/atl/

representative examples of this trend are the works of Fleurey et al. [8] or Guerra [9]. On the contrary, there is little work when it comes to white-box test model generation approaches. To the best of the author's knowledge, before the publication of ATLTest, only two white-box approaches had been proposed [10, 11]. In these approaches, the model transformation definition is analyzed to detect the subset of the input metamodel (and possible relevant values for the metamodel attributes) that is accessed during the transformation and thus focus the generation of tests on that subset. In the case of ATLTest, the coverage of the input metamodel is derived from the test models generated when addressing the coverage of the model transformation internal structure. This analysis of the internal transformation structure also guarantees that the test models generated with ATLTest exercise all branches in the transformation, this way maximizing the effectiveness of the testing experience.

5 Conclusion

In this paper we have presented two tools that represent a significant step forward to improve the quality and reliability of MDE-based software development processes. The first one is EMFtoCSP, a tool for the verification of models aimed at ensuring model correctness. The second one is ATLTest, a test model generation framework for the generation of test models to be used when testing model transformations.

There are still challenges that must be addressed, though. Regarding model verification tools, they should provide a more meaningful response to unambiguosly help designers to determine what the conflicting elements of the model are and how they should be modified in order to make the model correct. Besides, since model creation is an iterative process, these tools should provide mechanisms to avoid complete model verification every time the model is refined. Regarding testing of model transformations, mixed strategies combining black-box and white-box approaches are usually encouraged in order to get a better testing experience. In this regard, grey-box testing is a promising testing paradigm that should be explored.

References

[1] Jordi Cabot, Robert Clarisó, and Daniel Riera. Umltocsp: a tool for the formal verification of uml/ocl models using constraint programming. In *ASE*, pages 547–548. ACM, 2007.

[2] Carlos A. González, Fabian Buttner, Robert Clarisó, and Jordi Cabot. Emftocsp: A tool for the lightweight verification of emf models. In *Software Engineering: Rigorous and Agile Approaches (FormSERA), 2012 Formal Methods in*, pages 44 –50, june 2012.

[3] Carlos A. González and Jordi Cabot. Atltest: A white-box test generation approach for atl transformations. In *MoDELS*, pages 449–464, 2012.

[4] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on uml class diagrams. *Artif. Intell.*, 168(1-2):70–118, 2005.

[5] Anna Queralt, Alessandro Artale, Diego Calvanese, and Ernest Teniente. Ocl-lite: Finite reasoning on uml/ocl conceptual schemas. *Data Knowl. Eng.*, 73:1–22, 2012.

[6] Achim D. Brucker and Burkhart Wolff. The HOL-OCL book. Technical Report 525, ETH Zurich, 2006.

[7] Kyriakos Anastasakis, Behzad Bordbar, Geri Georg, and Indrakshi Ray. Uml2alloy: A challenging model transformation. In *MoDELS*, pages 436–450, 2007.

[8] Franck Fleurey, Benoit Baudry, Pierre-Alain Muller, and Yves Le Traon. Qualifying input test data for model transformations. *Software and System Modeling*, 8(2):185–203, 2009.

[9] Esther Guerra. Specification-driven test generation for model transformations. In *ICMT*, pages 40–55, 2012.

[10] Franck Fleurey, Jim Steel, and Benoit Baudry. Validation in model-driven engineering: testing model transformations. In *Int. Workshop on Model, Design and Validation*, pages 29 – 40, nov. 2004.

[11] Jochen Malte Küster and Mohamed Abd-El-Razik. Validation of model transformations - first experiences using a white box approach. In *MoDELS Workshops*, pages 193–204, 2006.