Journal of Computer Science 10 (8): 1423-1427, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.1423.1427 Published Online 10 (8) 2014 (http://www.thescipub.com/jcs.toc)

APPLYING BLACK-BOX TESTING TO MODEL TRANSFORMATIONS IN THE MODEL DRIVEN ARCHITECTURE CONTEXT

[1]Luciane Telinski Wiedermann Agner, [1]Inali Wisniewski Soares, [2]Jean Marcelo Simao and [2]Paulo Cézar Stadzisz

[1]Department of Computer Science, UNICENTRO-Mid-West State University, Guarapuava, Brazil
[2]CPGEI, UTFPR-Federal University of Technology Parana, Curitiba, Brazil

Received 2014-02-10; Revised 2014-02-18; Accepted 2014-03-20

ABSTRACT

Testing model transformations has played a leading role with the dissemination of MDA in software development processes. Software testing based on black-box testing, together with the "category partitioning" method, can be efficiently used in order to conduct the verification of model transformations. This study employs software testing techniques to an ATL model transformation in the MDA context and points out their benefits. The black-box testing method was adapted to the MT-PROAPES model transformation based on profiles and platform models. The platform models define the range of input models of the MT-PROAPES and are used for the creation of the test cases. The test cases were selected so as to meet certain requirements and increase the ability to detect errors in the model transformation. This approach makes the test process more agile and does not require any abstraction of behavioral properties of the transformations. The field of transformation testing and verification still faces significant challenges and requires a lot of research. Although having some limitations, black-box testing conforms to various situations, besides allowing its integration with other test strategies.

Keywords: Model Transformation, Transformations Testing, Functional Testing, Category Partitioning

1. INTRODUCTION

The increasing complexity of software development processes has encouraged the creation of the Model Driven Architecture (MDA). MDA is a software development approach that aims to transform platform independent into platform dependent models and eventually into executable source codes (Qiu and Zangh, 2012). The primary benefits claimed by model driven approaches consist of portability, productivity and reusability (Agner *et al*., 2013).

The MDA development process invloves the specification of a Platform Independent Model (PIM) and the Model Transformation (MT) of a PIM into a Platform Specific Model (PSM) based on a specific platform (Loniewski *et al*., 2011). The PIM constitutes a model represented at a high abstraction level and defines the software functionalities, architecture and behavior. PIMs can survive the advancements in technology and software architectures. In its turn, the PSM is also a software model containing details of the target platform, such as a specific operating system or software libraries.

Model Transformation (MT) is an essential issue in MDA. Transformation between models can be defined as the translation of a model from a higher abstraction level to a lower abstraction level, based on a group of clearly defined rules (Dube and Dixit, 2012). Several Model Transformation Languages (MTLs) were proposed so as to define and execute Model-To-Model

Corresponding Author: Luciane Telinski Wiedermann Agner, Department of Computer Science, UNICENTRO-Mid-West State University, Guarapuava, Brazil

(M2M) transformations. Atlas Transformation Language (ATL) is one of the most prominent among these languages, widely recognized as a solution for the development of model transformations (Tolosa *et al*., 2011; Troya and Vallecillo, 2011).

In the software development cycle, once a model transformation is specified and implemented by an MTL, the verification process consists in determining whether the implementation of this transformation meets its specifications (Lamari, 2007). Model transformations are defined by means of transformation programs or modules and, therefore, are considered as software. In this way, well-established techniques in the field of software engineering can be used to test and verify these model transformations.

This study illustrates how the functional test, also known as black-box testing, can be applied to the verification of MTs. Next, the test is employed in the verification of the model transformation MT-PROAPES (Soares *et al*., 2012), defined in ATL and based on a platform profile. The results are discussed in the endof this article.

2. MATERIALS AND METHODS

The development of model transformations is not a trivial task and thus is liable to errors. The creation of a model transformation is not complete before tests are done and the fulfillment of the implementation specifications is verified (Bauer and Küster, 2011). In this sense, verification increases reliability and usability of the transformations. When a transformation does not work properly, it may insert errors in the generated software model or code. Testing and verifying the model transformation is thus necessary in order to detect and correct errors as soon as possible.

Traditionally, testing consists in executing a program in previously selected representative situations by defining adequate test cases. Software testing is part of the software development process (Kaur and Singh, 2012). It is used to reveal system errors, to assure the system conforms to its specifications and to verify whether the system behaves in the intended manner.

"Software testing" techniques are widely employed for verifying and testing model transformations (Wang *et al*., 2008; Lamari, 2007). Such techniques consist in verifying whether a transformation works properly for a particular group of test cases composed of some input models, without trying, however, to validate it for all the input combinations.

2.1. Black-Box Test

Model transformation tests can be carried out through the functional technique, as known as "black-box" testing, aiming at verifying the functional requirements of the Software being tested. Under functional tests, the generation of test cases is independent of the model transformation implementation, once it is based on the transformation specification. "Black-box" tests are widely employed in the verification of transformations in the MDA context (Guerra, 2012; Fiorentini *et al*., 2010; Sen *et al*., 2009; Fleurey *et al*., 2009; Wang *et al*., 2008; Lamari, 2007).

The conduction of functional tests in model transformations must follow the steps below:

- Generation of test cases based on the model transformation specification and the specified test criteria. In the group of test cases generated, every test case must define an input model as well as the expected results, i.e., the target model expected as result of the model transformation execution
- Tests must be executed in accordance with the group of test cases selected
- Results obtained in the execution must be analyzed, that is, the result expected for each test case must be compared to the result obtained
- Changes in the transformation programs (modules) must be made so as to address the errors found during the execution of step 3

The testing activity must define a group of test cases in which the probability to find an unrevealed error is high. The creation stage of the group of test cases can be founded on the "category partitioning" method (Chimisliu and Wotawa, 2012). According to this method, the input data partitioning is based on the assumption that the software to be tested behaves in the same way for a data group forming a partition class. The idea consists in dividing an input domain into some groups and choosing a specific test model from each group. In this way, data are partitioned regarding some criteria that shall affect the data input characteristics likely to have an impact on the software behavior under test. The "category partitioning" method is adopted in several researches that perform model transformation tests within the MDA approach (Fleurey *et al*., 2009; Pons and Garcia, 2008; Lamari, 2007).

The generation of significant models approach (Fleurey *et al*., 2009), based on the topology of the domain to be partitioned, can also be used as a complement to the "category partitioning" method. This approach focuses on the program elements that must be exercised and the coverage measures the level at which the elements of an implementation are executed during the tests. Based on the category partitioning and generation of significant models approaches, the coverage criteria can be defined and used to select the test cases.

2.2. MT-PROAPES Model Transformation

The MT-PROAPES defines a PIM-to-PSM model transformation. Both the PIM and the PM apply elements of the PROAPES profile (Soares *et al*., 2012). The PROAPES defines a group of stereotypes in order to abstractly describe the services provided by a platform of embedded systems, based on specific processors required for their execution. The MT-PROAPES aims to provide independence between the transformation rules and the platform features by means of the PROAPES profile.

The MT-PROAPES transformation was implemented in ATL, a hybrid transformation language based on rules and designed to express model transformations as required by model-driven approaches (Troya and Vallecillo, 2011). ATL is one of the most widely used model transformation languages (Tolosa *et al*., 2011). Besides being defined by an ATL module, the MT-PROAPES transformation was structured by means of transformation rules. The ATL module implements specific rules for each type of stereotype applied to a message of the PIM model (Soares *et al.*, 2012). In so doing, considering that the PROAPES profile defines several stereotypes related to hardware and software applications, the MT-PROAPES transformation is composed of rules oriented to the transformation of the source model elements related to each one of these stereotypes. The MT-PROAPES module is composed of specific rules for creating the new PSM elements that contain platform-specific features.

3. RESULTS

The “black-box” test together with the “category partitioning’ method were employed in the transformation of MT-PROAPES models. The MT-PROAPES makes use of the ATL language and is based on a platform profile that contains several stereotypes (organized as a domain model). In this case, a group of appropriate test cases for the developed transformations must fulfill at least the following coverage requisites:

- Rule coverage: Each transformation rule defined must be exercised by at least one test cases. Given that the ATL is a transformation language based on rules, the rule-based coverage is appropriate in this context
- Linkage component coverage: Each linkage component defined by a stereotype must be instanciated in at least one test model. Therefore, the idea resides in separating the components defined in these profiles in specific ranges in order to assure such special values to be used in the test

Given the fact that, in the context of this research, platform profiles define the range of the input models of the transformations, for generating test cases it is adequate to define criteria based on profiles and use them for the partition of the input range (Wang *et al*., 2008). In so doing, the selection of a group of relevant test data is intended. The selection of test cases, therefore, involved the selection of valid Input Models (PIMs) from a group of input models. Such test models are valid once they belong to the transformation input domain (in conformity with the input meta-model) (Sen *et al*., 2009).

The “generation of significant models” approach (Fleurey *et al*., 2009) was used together with the “category partitioning” method. In this way, the performed partitioning was based on the topology defined through the stereotypes established in the linkage profiles and used to annotate the PIM model.

Tests of ATL-based model transformations were conducted so that the groups of selected test cases were considered. To do so, it is important to point out the execution of an ATL transformation, involving the following steps: (i) Verification of semantic errors related to the ATL meta-model and the source and target meta-models; (ii) compilation; and (iii) execution of the transformation (Brunelière *et al*., 2010). That is to say, in the first two steps the verification of certain types of errors is already performed and the execution is interrupted, if needed. The syntactic correctness of the model transformation is assured by the ATL language compilation process. Thus, the ATL compiler verifies whether the program/model developed is syntactically correct in relation to this language.

Syntax errors found in this stage were solved during the conduction of the referring tests. Other cases in which errors occurred were those in which the properties (tagged-values) related to the stereotypes had not been correctly completed by the designer. It is yet important to point out that such error is not reflected on the target model generated by the transformation, given that the transformation execution process is interrupted by the ATL as the error is detected. In order to try to minimize this problem, constraints Object Constraint Language (OCL) can be defined, through OCL rules, thus making it possible to validate the PIM model in relation to the OCL constraints. In this way, it is possible to anticipate the problem, ensuring that the PIM is defined in accordance with the requisites required by the transformation.

4. DISCUSSION

The state-of-the-art status of model transformation tests is presented in (Selim *et al*., 2012). The "black-box"software testing technique was adapted to the MT-PROAPES transformation. This technique enabled testing the execution of the implemented transformation, so the obtained result could be evaluated and compared to the expected result. The generation of test models by means of black-box technique has become more popular since they do not need to deal with the transformation language or technology (González and Cabot, 2012).

Although it is not possible to prove the transformation correctness fully, by means of functional tests, this approach is very useful in identifying errors in a feasible way and enables verifying the transformations developed without needing to abstract any of the structural or behavioral properties of the transformations (Gogolla and Vallecillo, 2011). This makes the test execution process of the MTs more agile and less complex to be performed.

Compared to software testing, model transformation testing bears an additional challenge: The complex nature of model transformation (González and Cabot, 2012). Generally, models are formed by elements of different kind and must conform to a meta-model, what hampers the generation of test case models and the analysis of the final result obtained.

Another important strategy that can be employed in MT testing regards the use of OCL constraints. A method that applies contract-based OCL to specify and validate a model transformation is presented in (Cariou *et al*., 2011). This method focuses on contracts written in OCL to validate a transformation result with respect to the transformation specification. Another method to derive OCL into variants from declarative model transformations in order to enable their verification and analysis is proposed in (Cabot *et al*., 2010). It is important to highlight that the verification through OCL constraints is more difficult to be performed, though its results are more complete.

5. CONCLUSION

The correctness of transformations is essential to the success of MDA. This study reviewed the employment of the functional testing technique as well as its stages in model transformation testing. The functional testing technique wasadapted to the MT-PROAPES transformation based on its main features (platform profile and domain model). The employment feasibility of the technique in a model transformation, defined through a rule-based language, was also presented. This technique allowed testing the execution of the MT-PROAPES transformation as well as verifying whether the obtained result met the expected result.

More advanced techniques can be selected to perform the verification of model transformations, e.g., the use of OCL constraints. In this way, black-box testing can be integrated with these techniques for testing and validating MTs. In addition, it is important to point out that the field related to model transformation testing presents new challenges and still demands researches.

6. REFERENCES

Agner, L.T.W., I.W. Soares, P.C. Stadzisz and J.M. Simão, 2013. A Brazilian survey on UML and model-driven practices for embedded software development. J. Syst. Software, 86: 997-1005. DOI:10.1016/j.jss.2012.11.023

Bauer, E. and J.M. Küster, 2011. Combining specification-based and code-based coverage for model transformation chains. Proceedings of the 4th International Conference Model Transformations on Theory and Practice, Jun. 27-28, LNCS, Springer, pp: 78-92. DOI: 10.1007/978-3-642-21732-6-6

Brunelière, H., J. Cabot, F. Jouault, M. Tisi and J. Bézivin *et al*., 2010. Industrialization of research tools: The ATL case. Proceedings of the 3rd International Workshop on Academic Software Development Tools and Techniques, Jan. 5-5, Antwerp, Belgique.

Cabot, J., R. Clarisó, E. Guerra and J. de Lara, 2010. Verification and validation of declarative model-to-model transformations through invariants. J. Syst. Software, 83: 283-302. DOI: 10.1016/j.jss.2009.08.012.

Cariou, E., C. Ballagny, A. Feugas and F. Barbier, 2011. Contracts for model execution verification. Proceedings of the 7th European Conference on Modelling Foundations and Applications, Jun. 6-9, LNCS, Springer, pp: 3-18. DOI: 10.1007/978-3-642-21470-7-2

Chimisliu, V. and F. Wotawa, 2012. Category partition method and satisfy ability modulo theories for test case generation. Proceedings of the 7th International Workshop on Automation of Software Test, Jun. 2-3, IEEE Xplore Press, Zurich, pp: 64-70. DOI: 10.1109/IWAST.2012.6228992

Dube, M.R. and S.K. Dixit, 2012. Modeling theories and model transformation scenario for complex system development. Int. J. Comput. Appli., 38: 11-18. DOI: 10.5120/4698-6847

Fiorentini, C., A. Momigliano, M. Ornaghi and I. Poernomo, 2010. A constructive approach to testing model transformations. Proceedings of the 3rd International Conference on Theory and Practice of Model Transformations, Jun. 28-Jul. 2, LNCS, Springer, pp: 77-92. DOI: 10.1007/978-3-642-13688-7-6

Fleurey, F., B. Baudry, P.A. Muller and Y.L. Traon, 2009. Qualifying input test data for model transformations. Software Syst. Model., 8: 185-203. DOI: 10.1007/s10270-007-0074-8.

Gogolla, M. and A. Vallecillo, 2011. Tractable model transformation testing. Modell. Foundat. Applic. LNCS, 6698: 221-235. DOI: 10.1007/978-3-642-21470-7_16

González, C.A. and J. Cabot, 2012. ATLTest: A white-Box test generation approach for atl transformations. Proceedings of the 15th International Conference Model on Driven Engineering Languages and Systems, Sept. 30-Oct. 5, LNCS, Springer, pp: 449-464. DOI: 10.1007/978-3-642-33666-9-29

Guerra, E., 2012. Specification-driven test generation for model transformations. Proceedings of the 5th International Conference Theory and Practice of Model Transformations, (ICMT’12). LNCS, Springer, pp: 40-55. DOI: 10.1007/978-3-642-30476-7_3

Kaur, P. and H. Singh, 2012. Configuration management issues in software process management. Am. J. Eng. Applied Sci., 5: 261-265. DOI: 10.3844/ajeassp.2012.261.265

Lamari, M., 2007. Towards an automated test generation for the verification of model transformations. Proceedings of the ACM Symposium on Applied Computing, (SAC’07), pp: 998-1005. DOI: 10.1145/1244002.1244220

Loniewski, G., A. Armesto and E. Insfran, 2011. An architecture-oriented model-driven requirements engineering approach. Proceedings of the Model-Driven Requirements Engineering Workshop, Aug. 29-29, IEEE Xpoler Press, Trento, Italy, pp: 31-38. DOI: 10.1109/MoDRE.2011.6045364

Pons, C. and D. Garcia, 2008. A lightweight approach for the semantic validation of model refinements. Elect. Notes Theor. Comput. Sci., 220: 43-61. DOI: 10.1016/j.entcs.2008.11.005

Qiu, W. and L. Zhang, 2012. Research on real-time software development approach. J. Software, 7: 1593-1600. DOI: 10.4304/jsw.7.7.1593-1600

Selim, G.M.K., J.R. Cordy and J. Dingel, 2012. Model transformation testing: The state of the art. Proceedings of the 1st Workshop on the Analysis of Model Transformations, (AMT’ 12), New York, pp: 21-26. DOI:10.1145/2432497.2432502

Sen, S., B. Baudry and J.M. Mottu, 2009. Automatic model generation strategies for model transformation testing. Proceedings of the 2nd International Conference on Theory and Practice of Model Transformations, Jun. 29-30, LNCS, Springer, pp: 148-164. DOI: 10.1007/978-3-642-02408-5-11

Soares, I.W., L.T.W. Agner, P.C. Stadzisz and J.M. Simão. 2012. Modeling of embedded software on MDA platform models. J. Comput. Sci. Technol., 12: 133-139.

Tolosa, J.B., O. Sanjuán-Martínez, V. García-Díaz, B.C.P. G-Bustelo and J.M.C. Lovelle, 2011. Towards the systematic measurement of ATL transformation models. Software Pract. Exp., 41: 789-815. DOI: 10.1002/spe.1033

Troya, J. and A. Vallecillo, 2011. A rewriting logic semantics for ATL. J. Object Technol., 10: 1-29. DOI: 10.5381/jot.2011.10.1.a5

Wang, J., S.K. Kim and D. Carrington, 2008. Automatic generation of test models for model transformations. Proceedings of the 19th Conference on Australian Software Engineering, Mar. 26-28, IEEE Xplore Press, Perth, WA., pp: 432-440. DOI: 10.1109/ASWEC.2008.4483232