Applying Model-Driven Web Engineering to the Testing Phase of the ADAGIO Project

L. Morales(✉), S. Moreno-Leonardo, M. A. Olivero, A. Jiménez-Ramírez, and M. Mejías

Computer Languages and Systems Department, University of Seville, Avenida Reina Mercedes, s/n, 41010 Seville, Spain
{leticia.morales, sara.moreno, miguel.olivero, andres.jimenez}@iwt2.org, risoto@us.es

Abstract. The Model-Driven Engineering (MDE) has been used in recent years to promote better results in the development of Web Applications, in the field that has been called Model-Driven Web Engineering (MDWE). One of the advantages of applying MDWE is that it offers a solution to reduce the cost of the tests without affecting their quality execution. This paper presents the application of a MDWE methodology (Navigational Development Techniques, NDT) that provides support for all the phases of the lifecycle of a software project development proposing transformations between these phases, to manage the test phase of a real-world case study named ADAGIO. This project, among other goals, proposes the development of a web application whose main objective is to offer researchers the possibility of integrating and consolidating heterogeneous data sources, showing a unified vision of them, allowing to simplify the search task in different repositories as well as the relationship between the sources found.

Keywords: Model-Driven Web Engineering · NDT · Early Testing
Web application

1 Introduction

Model-Driven Engineering (MDE) offers the advantage to support automation, as the models can be automatically transformed from the early stages of development to the final ones. Therefore, MDE enables automating tasks involved in software development, such us testing tasks [1]. The process of software development involves a series of activities in which the chances of a human error is high (mistakes can happen in the beginning of the process, in which the objectives can be inadequately specified, as well as during later steps) [2]. In this context, trying to find errors in the earliest stages of development, would greatly help to reduce the costs of the total computation of the project, showing errors before they appear.

Model-Driven Testing (MDT) [3], increases the level of automation, automating not only the execution of system tests, but also their design. System tests are generated automatically from a software product model. This results in a repeatable and rational

basis for testing the product, ensuring the coverage of all its behaviors, and allowing the tests to be directly linked to the requirements.

In the Web Engineering and Early Testing IWT2 research group, where this research has been carried out, we have spent years working on the Model-Driven Web Engineering (MDWE) [4] paradigm. From this research, the Navigational Development Tools (NDT) [5] methodology was born, including a very important pillar based in MDT. This paper presents a real-world case study of validation about the application of the automatic system tests generation that NDT proposes in a concrete scenario: the ADAGIO Project.

The remainder of this paper is organized as follows: Sect. 2 presents some related work based on Model-Driven proposals that aim to automate the generation of system tests, Sect. 3 describes the ADAGIO project is being developed based on two main pillars: MDE and NDT. Section 3 presents the ADAGIO project. Section 4 describes how the system tests generation proposed by NDT methodology has been applied into the ADAGIO project. And finally, Sect. 5 summarizes a set of conclusions and future works.

2 Related Work

There are a great variety of proposals that are based in methods for generating system tests. Next, some related work is presented.

Anand et al. [6] proposed the following classification for MDT for modeling notations: scenario-oriented, state-oriented and process oriented notations. The first one, directly describe the sequences of input and output between the System-Under-Test (SUT) and its environment. Usually, they are based on sequence charts, activity charts or use case diagrams although textual variations have also been proposed. This technique is generally simpler than other ones because a scenario is very similar to a test case. However, the system tests generation are not final system tests because it is needed to perform a test selection and to define the input parameters. The second one describes the SUT by its reaction on an input or output in a given state. As result, the models state is evolved and an output maybe produced. This technique may be expressed as diagrammatic or textual form. The last modeling notation describes the SUT in a procedural style, where inputs and outputs are received and sent as messages on communication channels. The approach that has been used in this paper is focused in scenario-oriented notations. In this sense, some related work to generate automatic system tests using this technique are presented.

Nogueira et al. [7] offer a strategy for the automatic generation of system tests of parameterized use cases templates. This approach considers a natural language representation that mixes control and state representation, which can be used to select particular scenarios during test generation. Kumar et al. [8] propose a model to generate an activity flow table (AFT) from an activity diagram converting it to activity flow graph (AFG). By using activity coverage criterion, authors generate the test paths through traversing the AFG. Finally, the system tests from these paths are generated. Olajubu et al. [9] presented a Model to Text (M2T) transformation language for test case generation from requirement specifications expressed in a Domain Specific Language (DSL).

Gutiérrez et al. [10] introduce a systematic process based on the Model-Driven paradigm to automate the generation of functional system tests from functional requirements, defining a set of metamodels and transformations and a DSL. Marín et al. [11] show a model-based testing technique that automatically generates abstract system tests from conceptual models used in Model-Driven Development (MDD) and a model-based testing technique that automatically generates concrete system tests in Java and C#, reducing the testing effort in MDD projects environments. Elalloui et al. [12], propose an improved version of a previous work where they implemented an algorithm that took as input user stories, and automatically generate UML sequence diagrams. In this improved work, authors automatically generate the system tests applying Model to Model (M2M) and M2T transformations through the use of AndroMDA, an open source MDA Framework. Usaola et al. [13] show a method to describe generic test scenarios by means of regular expressions, whose symbols point to a System Under Test (SUT) operation.

3 Background: The ADAGIO Project

The Analytics Data Aggregated Geolocation Open (ADAGIO) project, is funded by the center for industrial technological development (Centro para el Desarrollo Tecnológico Industrial, CDTI), public business entity of the Ministry of Economy, Industry and Competitiveness of Spain. This initiative is launched by the consortium of the IWT2 research group of the University of Seville and Servinform S.A. company.

The vision of the ADAGIO project is to develop of a web application that combines strategies of Big Data and Machine Learning in areas of treatment of geotagged data from diverse and heterogeneous data sources, to generate knowledge, starting from Open Data information already available. The generic goals of this project are:

- Facilitate research and data processing.
- Obtain geotagged populations that fit predefined criteria.
- To enable the performance of statistical analyzes and correlations of these populations.

ADAGIO functional model (Fig. 1) proposes 4 main modules: (i) download and catalog of data sources, (ii) the entity reconciliation process, (iii) administration and management of the data sources, and (iv), exploitation of the information through a web application.

This paper focuses on the fourth module. The main goal of this module is based the creation of queries by users of the system, in a language as natural and high level as possible, applying supervised Machine Learning and Text Mining techniques, with the aim of extracting hidden knowledge from the data sources previously processed in the other modules. The functionalities that this module will offer are: user authentication, data exploitation, automation of the import into the system of the data sources that have been identified at the beginning of the project and finally, the data sources will be available for their exploitation and accessible for future queries working as a unique data source thanks to the automation of the import process.

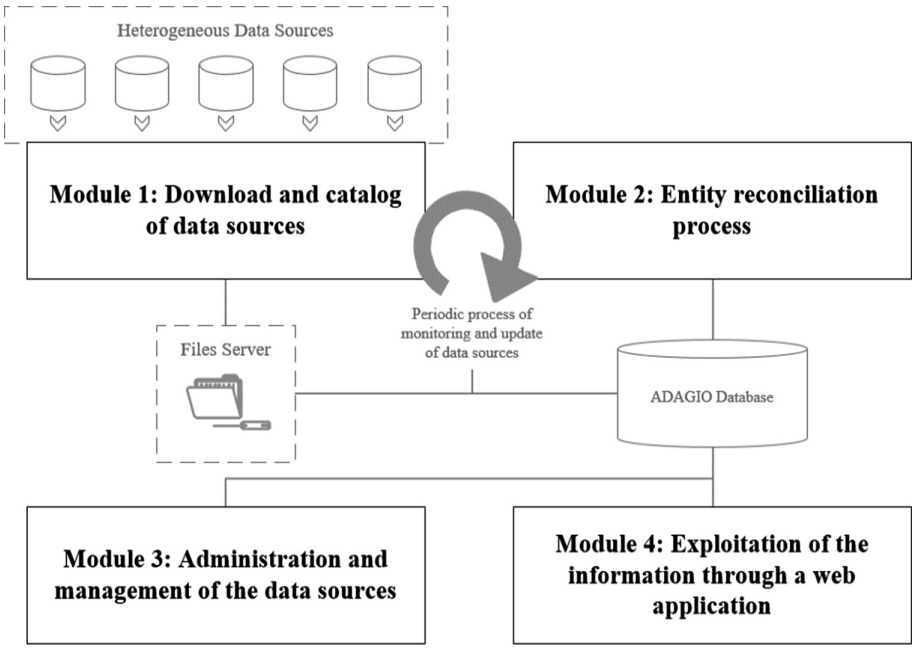

Fig. 1. ADAGIO functional model.

4 Model-Driven Web Engineering in the ADAGIO Project

Navigational Development Techniques (NDT) is a methodological approach oriented to Web Engineering [5]. NDT defines a set of metamodels for every phase of the life cycle of software development: the feasibility study phase, the requirements phase, the analysis phase, the design phase, the implementation phase, the testing phase, and finally, the maintenance phase.

Nowadays, NDT is a complete framework named "NDTQ-Framework" that provides support for all the phases of the lifecycle of a software project development and proposes automatic transformations between these phases. NDTQ-Framework proposes six groups of processes that have been defined on the basis of the life cycle of NDT although its terminology has referenced the standards ISO 12207 [14]. These processes are: development, software maintenance, testing, software quality, management and security.

NDT-Suite [15] is a set of tools that allow a software engineer to apply the NDTQ-Framework in a business environment. NDT-Suite is composed of 5 main tools: NDT-Profile, NDT-Quality, NDT-Driver, NDT-Prototypes, NDT-Glossary. The three more important are:

- NDT-Profile: this tool defines all the profiles that have been created to instantiate the metamodel proposed by NDT in the Enterprise Architect (EA) case tool. Profiling these metamodels, it is possible to use each NDT artifact easily.

- NDT-Quality: receives as input a NDT-Profile project and the main goal of the tool is to verify that the traceability between artifacts and NDT rules are met.
- NDT-Driver: implements a set of automatic procedures that allows transformations between the NDT-defined models using the NDT-Profile, thus generating analysis models of the requirements, analysis design models and test models of the requirements.

In the case of the ADAGIO project, the transformation that has been carried out using NDT-Driver covered the phases: requirements to analysis and requirements to testing. From the last one, it is possible obtain the set of system tests of the ADAGIO platform. The execution of these tests will allow the tester to obtain the test coverage level of the platform.

The process for generating the system tests using NDT-Driver must follow these steps:

- Define the requirements with NDT-Profile and execute NDT-Quality to check that the traceability and NDT standards are fulfilled in a specific project. This process decreases the development time since it allows applying NDT transformations from the requirements to the analysis, analysis to design, and requirements to testing phases of the lifecycle of a software development project.
- Specifically, for each functional requirement, the tool creates a new artifact called "System Test" with information relevant to that test.
- Subsequently, the different tests that have been generated from the definition of the functional requirements are associated to each artifact. The functional requirements are based on activity diagrams, so the NDT-Driver generates a test for each possible path that exists in the diagram between the initial and the end activity.

Left side of Fig. 2, shows how the functional requirement "RF-01. System Login" of the ADAGIO platform has been defined through the use of the EA tool. It is possible to see that it is highlighted in red color, one of the possible paths of the activity diagram considering it as one of the tests generated for the ADAGIO platform. Right side of Fig. 2 shows one of the specific tests generated for this functional requirement.

Once all the system tests considered relevant have been obtained, it is possible to generate a Test Plan through the use of the NDT-Report tool. This tool defines a template for the description of each of the tests automatically generated. This template includes the definition of the system tests in two different ways: a table that describes the name, version, date of creation, authors, relationships, description and pre and post conditions of the test, and the other one, where the activity diagram of the test is defined, which shows all the activities that the tester must perform to carry out the test satisfactorily.

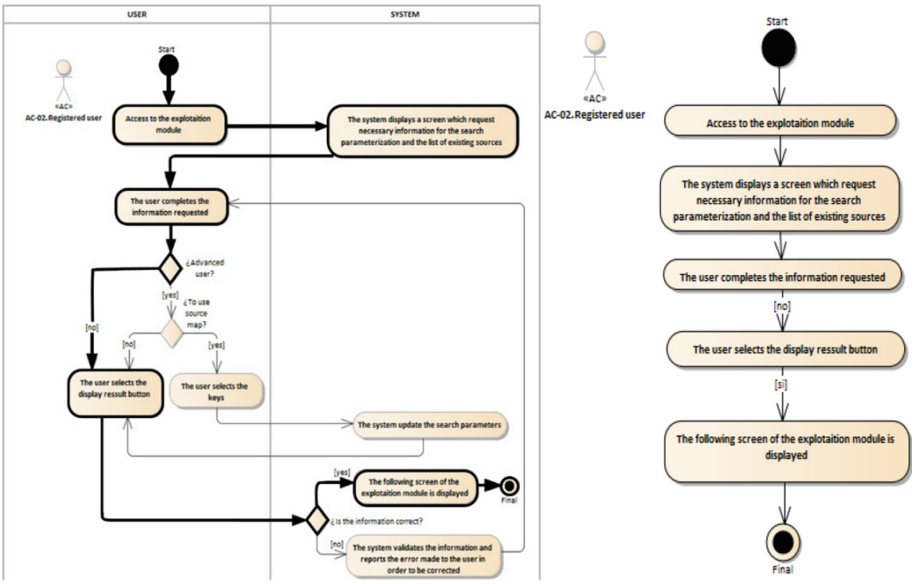

Fig. 2. Use case definition with EA and system test generated by NDT.

4.1 Results

As mentioned before, NDT methodology has been used, first, for defining the requirements and analysis phase of ADAGIO project, and second, for generating the system tests of that project.

After performing the requirement and analysis phases of the ADAGIO project, a total of 25 functional requirements (FR) were defined. The definition of the Test Plan of the web platform was defined in two ways: (i) by Servinform S.A. company and (ii), by IWT2 research group using NDT.

As possible to note in Table 1, where results of the number of system tests generated by each group are showed, the total number of system tests generated by IWT2 (136) are quite higher than the ones generated by Servinform S.A. (76), increasing considerably the system testing coverage level.

In addition, with the aim of rising the automation level of the Test Plan execution, it was used Selenium tool [16]. With this tool, authors achieved to reduce notably the cost of executing the Test Plan for each iteration.

5 Conclusion and Future Work

This paper has presented how a Model-Driven Web Engineering methodology (NDT) has been applied to a real world project called ADAGIO.

NDT has been used to define the requirement and analysis phase of the ADAGIO project, and, in addition, for generating and managing the test phase of this project. The definition of the requirements and analysis phase ended with a total of 25 functional

Table 1. Definition of test plan by Servinform (S) and IWT2 research group (I)

FR	System tests		FR	System tests		FR	System tests		FR	System tests	
	S	I		S	I		S	I		S	I
MADMON											
FR-01	4	7	FR-02	6	9	FR-03	1	3	FR-04	4	7
FR-05	6	9	FR-06	1	3	FR-07	4	7	FR-08	6	9
FR-09	1	3	FR-10	4	7	FR-11	6	9	FR-12	1	3
FR-13	4	7	FR-14	6	9	FR-15	1	3	FR-16	1	2
MEXPL											
FR-01	2	5	FR-02	3	7						
MGEN											
FR-01	1	2	FR-02	1	2						
MGEST											
FR-01	1	2	FR-02	1	2	FR-03	4	7	FR-04	6	9
FR-05	1	3									
TOTAL	**20**	**38**		**22**	**39**		**16**	**31**		**18**	**28**

requirements. From these functional requirements, both, Servinform company and IWT2 research group, developed a Test Plan for the testing phase of the project. As mentioned before, NDT takes the activity diagram that defines each functional requirement to generate the different system tests. In this sense, the coverage level of the Test Plan will be the 100%.

Table 1 of Sect. 4, shows that Servinform company defined a total of 76 system tests and IWT2, a total of 136, increasing a bit more than the 44% of the ones created by Servinform. Thus, it is possible to conclude that the use of NDT for this project has served for improving the quality of the final product.

Finally, future works encompasses different tasks such as: (i) improve NDT-Driver to reduce the computational cost of generating tests from complex functional requirements and (ii), improve NDT-Driver to define a process that let automatically generate selenium tests based on the requirements specification allowing to reduce the cost of reviewing and instantiating the test plan.

Acknowledgment. This research has been supported by the POLOLAS project (TIN2016-76956-C3-2-R) of the Spanish Ministry of Economy and Competitiveness, the VPPI of the University of Seville and the ADAGIO Project (P106-16/E09).

References

1. Garcia-Garcia, J.A., Enriquez, J.G., Garcia-Borgonon, L., Arevalo, C., Morillo, E.: A MDE-based framework to improve the process management: the EMPOWER project. In: Proceedings of the 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017 (2017)
2. Enríquez, J.G., García-García, J.A., Domínguez-Mayo, F.J., Escalona, M.J.: ALAMEDA ecosystem: centering efforts in software testing development. In: Quality Control and Assurance-An Ancient Greek Term Re-Mastered, vol. 1, pp. 155–172 (2017)
3. Dai, Z.R.: Model-driven testing with UML 2.0. In: Computer Science at Kent, pp. 179–187 (2004)
4. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice (2012)
5. Escalona, M.J., Aragón, G.: NDT. A model-driven approach for web requirements. IEEE Trans. Softw. Eng. **34**, 377–394 (2008)
6. Saswat, A., et al.: An orchestrated survey of methodologies for automated software test case generation. J. Syst. Softw. **86**, 1978–2001 (2013)
7. Nogueira, S., Sampaio, A., Mota, A.: Test generation from state based use case models. Form. Asp. Comput. **26**, 441–490 (2014)
8. Jena, A.K., Swain, S.K., Mohapatra, D.P.: A novel approach for test case generation from UML activity diagram. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques, pp. 621–629 (2014)
9. Olajubu, O., Ajit, S., Johnson, M., Turner, S., Thomson, S., Edwards, M.: Automated test case generation from domain specific models of high-level requirements. In: Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems, RACS, pp. 505–508 (2015)
10. Gutiérrez, J.J., Escalona, M.J., Mejías, M.: A model-driven approach for functional test case generation. J. Syst. Softw. **109**, 214–228 (2015)
11. Marín, B., Gallardo, C., Quiroga, D., Giachetti, G., Serral, E.: Testing of model-driven development applications. Softw. Qual. J., 1–29 (2016)
12. Elallaoui, M., Nafil, K., Touahni, R.: Automatic generation of TestNG tests cases from UML sequence diagrams in Scrum process. In: IEEE International Colloquium on Information Science and Technology (CiSt), pp. 65–70 (2017)
13. Usaola, M.P., Romero, F.R., Aranda, R.R.-B., Rodriguez, I.G.: Test case generation with regular expressions and combinatorial techniques. In: 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops, pp. 189–198 (2017)
14. Jones, A.: ISO 12207 Software life cycle processes? Fit for purpose? Softw. Qual. J. **5**, 243–253 (1996)
15. García-García, J.A., Escalona, M.J., Domínguez-Mayo, F.J., Salido, A.: NDT-Suite: a methodological tool solution in the model-driven engineering paradigm. J. Softw. Eng. Appl. **7**, 206–217 (2014)
16. Selenium: Selenium website documentation. RA-MA Ed. Accessed Feb 2018