PalArch's Journal of Archaeology of Egypt / Egyptology

VANILLA Framework for Model Driven Re- Engineering of Declarative User Interface

Alok Aggarwal,[1] Smita Agarwal,[2] Narendra Kumar[3]

[1] School of Computer Science, University of Petroleum & Energy Studies, Dehradun, India, Email: alok.aggarwal@upes.ac.in
[2] Department of Computer Science, Mewar University, Chittorgarh (Raj), India, Email: smita.ag@gmail.com
[3] Icfai University, Jaipur, India, *drnk.cse@gmail.com*

Alok Aggarwal,[1] Smita Agarwal,[2] Narendra Kumar[3]: VANILLA Framework for Model Driven Re-Engineering of Declarative User Interface-- Palarch's Journal of Archaeology Of Egypt/Egyptology 17(9). ISSN 1567-214x.
***Keywords*—Model Driven Re-engineering, Model Transformation, Declarative User Interface, Models, Meta-Models.**

Abstract — A wide assortment of smart phone devices from different manufacturers that vary in platform, memory, processing capability screen size, screen resolution and compatibilities to different systems is available in today's market. To meet the ever growing demand of accessing websites and web application through the mobile devices, these web applications are being re-engineered to be acceptable in terms of cost and time. The usability of these web applications is highly dependent on the user interface. In this work, a platform independent Vanilla Framework is presented that uses model as the primary artifact. Model to model transformation is carried out from platform specific model to platform independent model. A detailed analysis of the existing metamodel-based transformation tools is done for the declarative model using an exhaustive criterion. The evaluation of the chosen tools is done which are open source and have download page available using search engine like Google Scholar and Github; like UML–RSDS, Tefkat, JTL, PTL etc. The analysis is performed over ten different parameters like language, model query, compatibility, cardinality etc. Based on the result of the evaluation and extensive investigation, a PIM for declarative user interface is proposed. Also the framework for model transformation using PIM model for declarative user interface has been put forward. This framework is then applied to five of the popular libraries such as SWING, HTML5, and more recent libraries of Android and Python-Tkinter. Results show that all selected tools produce domain specific target model which mostly transform PIM to PSM and none produces domain independent target model transforming a PSM into PIM.

Keywords—Model Driven Re-engineering, Model Transformation, Declarative User Interface, Models, Meta-Models

1. Introduction

Technology landscape is changing at a rapid pace with every gone year. About sixty percent of the world population accounting about 4.58 billion people are on-line today and the way how web is accessed all around the world is also changing moving more towards handheld devices from the desktop. Over the last one decade mobile traffic has gone up constantly however during 2018-2019 it was almost steady at 50 percent. With the emergence of 5G mobile traffic is expected to go even higher capered to what it was seen during the last half decade. Worldwide 52.2% of the web traffic came from mobile devices in 2019 compared to just 16.2% in 2013 from the mobile devices and 56.7% from the desktop devices in 2019 [1]. The smart phone devices have taken over desktops and laptops for accessing internet. The existing websites and web applications are being increasingly re-designed to improve upon their accessibility through these devices. Today the market has wide assortment of smart phone devices from different manufacturers that vary in platform, memory, processing capability screen size, screen resolution and compatibilities to different systems [2]-[3].

To meet the ever growing demand of accessing websites and web application through the mobile devices, these web applications are being re-engineered to be acceptable in terms of cost and time. The usability of these web applications is highly dependent on the user interface. To ensure excellent user experience, it should satisfy both in sensory and functional needs. The development of User Interface is highly dependent on their target platforms and usually don't take in consideration future portability. So when there is a need to port the application to another platform, the developer has to start from scratch.

The model and Meta-models of the system plays primary role in the re-engineering. The modeling and transformation techniques help to transform the source model to the target model. Both the source and the target models are platform specific models (PSM). The tools and techniques exist to reengineer the User Interface from PSM to PSM. This porting happens from a specific platform to another specific platform. If the source platform remains the same and the target platform differs, the entire exercise needs to be repeated again. This process is highly inflexible, constrained and very specific to particular couple (source and target platforms).

In this work, a platform independent Vanilla Framework is presented that uses model as the primary artifact. Model to model transformation is carried out from platform specific model to platform independent model. A detailed analysis of the existing metamodel-based transformation tools is done for the declarative model using an exhaustive criterion. The evaluation of the chosen tools is done which are open source and have download page available using search engine. The analysis is performed over ten different parameters. Based on the result of the evaluation and extensive investigation, a PIM for declarative user interface is proposed. Also the framework for model transformation using PIM model for declarative user interface has been put forward. This framework is then applied to five of the popular libraries such as SWING, HTML5, and more recent libraries of Android and Python-Tkinter.

Rest of the paper is organized as follows. Major earlier works done in the said domain are given in section 2. Proposed research methodology is given in section 3. Section 4 gives the declarative model driven re-engineering including proposed Vanilla transformation engine. Finally work is concluded with future work in section 5.

2. Related Work

During the last one and a half decade various modelling languages and techniques have been proposed for the design and development of software systems which are complex in nature. Major purpose of these initially developed modelling languages has been to better facilitate the system's coherent and common along with sharing so as to ease the communication amongst various stake holders of the system [4]. Later in the software engineering process models were considered as central theme of the system and not just the documentation artefacts. These advanced software engineering process models have been classified as model-driven engineering (MDE) like software factories [5], DSL engineering [6], MDA [7] or also as model-driven software development (MDSD) [8]-[9].

Existing reverse engineering approaches are quite flexible; be it the contemporary use of model analysis, model transformation, source code [10]-[11] or contemporary use of source code [12]-[14]. Platform Specific Models (PSM) are considered as initial models [15] which are abstracted and modified through model transformation. These models are termed as Platform Independent Models (PIM) [16]. The code of is analyzed by compilers and interpreters [17]: to calculate metrics [18], to find bugs [19], code clones [20] etc. There is program code where transformation was done on binary code [21] and for profiling [22], security [23], optimization [24] and refactoring [25]. The Model driven re-engineering process involves the source code for declarative user interface to obtain meta-model [26]. Then model transformation is applied to transform meta-model into platform specific models for a given platform (known as PSMs).

Modeling has been applied to reverse engineering by many researchers taking various contexts like the detection of design pattern [27]-[29], the reconstruction of software architecture [30], decomposition of design pattern into elemental structures [31]-[32] etc. For handling the business temporal rules, Arevalo et al. [33] proposed a model which addresses time-related issues. Business process model and natation metamodel have been extended.

Ovchinnikova et. al [34] proposed an approach for software development, TFM4MDA, based on model-driven engineering. Trias et al. [35]-[38] proposed various approaches for automating the migration of web applications to content management system based web application.

3. Research Methodology

Objective of the proposed work is to propose a frame work for re-engineering declarative user interface model transformation using PIM. Based on this objective, a conceptual framework is developed that formed the basis of the work done. A detailed analysis of the

existing metamodel-based transformation tools is done for the declarative model using an exhaustive criterion. The evaluation of the chosen tools, which are open source and have download page available using search engine like Google Scholar and Github, is done; like UML–RSDS, Tefkat, JTL, PTL etc. The analysis is performed over ten different parameters like language, model query, compatibility, cardinality etc. Based on the result of the evaluation and extensive investigation, a PIM for declarative user interface is proposed. Also the framework for model transformation using PIM model for declarative user interface has been put forward. This framework is then applied to five of the popular libraries such as SWING, HTML5, and more recent libraries of Android and Python-Tkinter.

4. Declarative Model Driven Re-engineering

In MDE model transformation plays a central theme for converting a source model to target model and visa-versa. Building a precise meta-model is a pre- requisite to model transformation where model can be a concrete or abstract model. The transformation can be bi-directional or uni-directional.

A detailed evaluation of the eleven chosen meta-model-based transformation tools is done for the declarative model using an exhaustive criterion. Gaps are identified for the declarative models in the existing meta-model-based transformation tools for the Model-to-Model transformation. Mostly those tools are identified which are open source and has download page available using search engine like Google Scholar and Github like UML-RSDS, Tefkat, JTL, PTL, ModTransf, Echo, QVTR-XSLT, ModelMorf, MediniQVT, PETE and TXL. The comparison criteria were Modeling Language, Meta-Modeling Language, Model Query, Compatibility with Standards, Model Transformation Language Syntax, Target Model, Cardinality, Type of Transformation, Directions, level of automation among many.

For the above work three languages namely Java, ASP and Turing have been selected, percentage of these languages are given in figure 1. Figure 2 and figure 3 gives the percentage of modelling languages and meta-modelling language respectively. Model Query Percentage in terms of Yes and No is given in figure 4. Percentage of compatibility with two standards is given in figure 5. Three types of target models have been considered; conservative, destructive and a combination of both, percentage of these target models are shown in figure 6. Three types of model transformation language syntax have been considered; textual, graphical and a combination of both, percentage of these is shown in figure 7. Two type of transformations have been considered; endogenous and exogenous, percentage of these is shown in figure 8. Four types of cardinality ratios have been considered; one-to-one, one-to-N, N-to-one and N-to-N, percentage of these four cardinality ratio is given in figure 9. Three types of direction of transformation have been considered; uni-directional, bi-directional and multi-directional, percentage of these three are shown in figure 10.

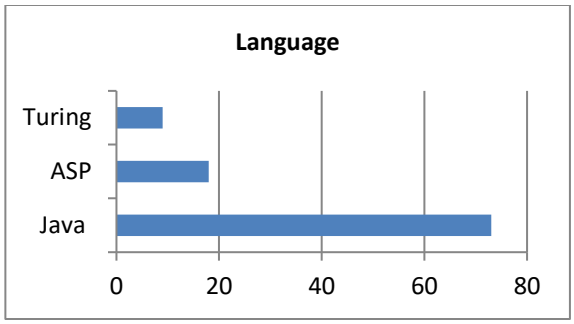

Fig. 1. Language Percentage

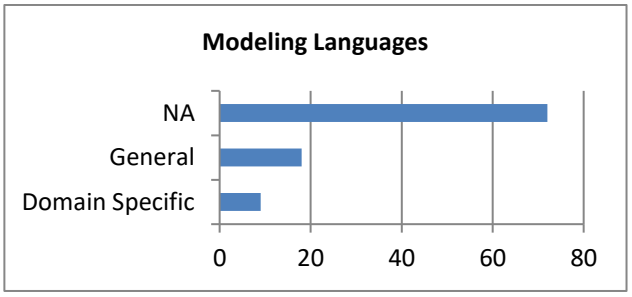

Fig. 2. Modeling Language Percentages

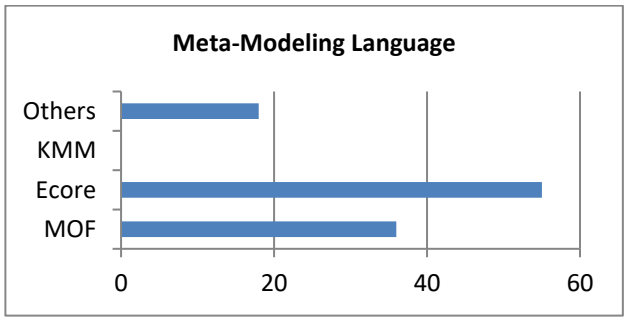

Fig. 3. Meta-Modeling Language Percentage

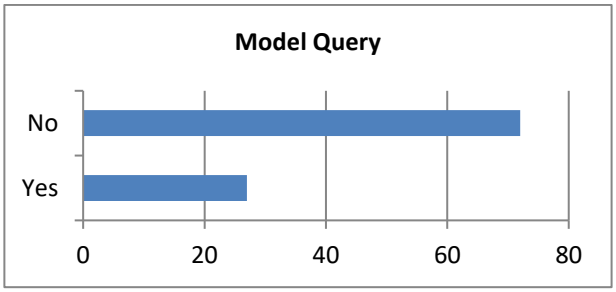

Fig. 4. Model Query Percentage

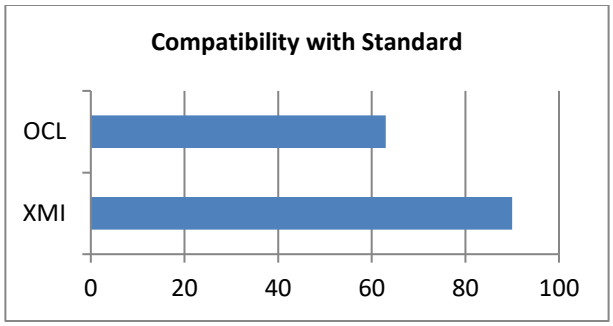

Fig. 5. Compatibility with Standards Percentage

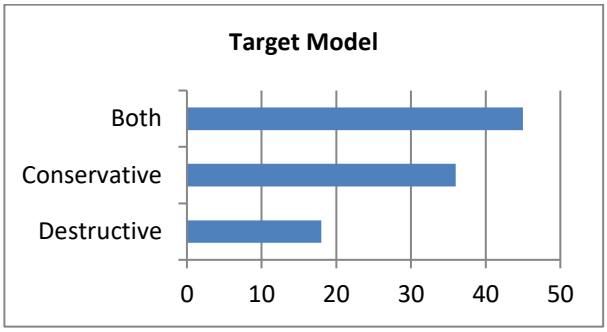

Fig. 6. Target Model Percentage

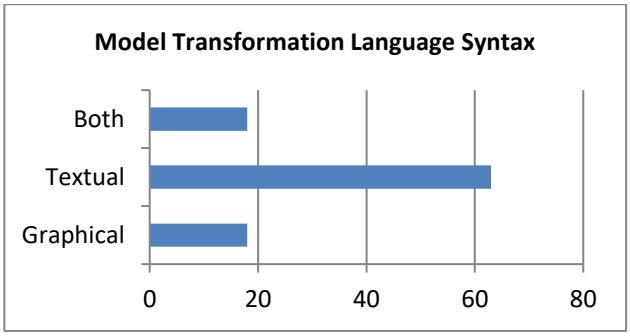

Fig. 7. Model Transformation Language Syntax Percentage

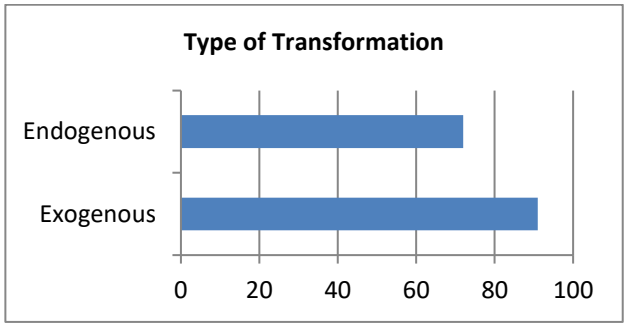

Fig. 8. Type of Transformation Percentage

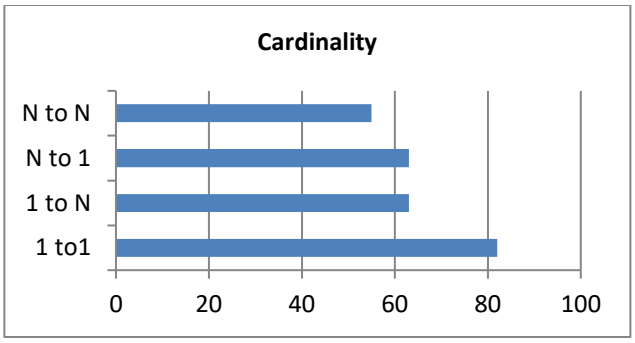

Fig. 9. Cardinality Percentage

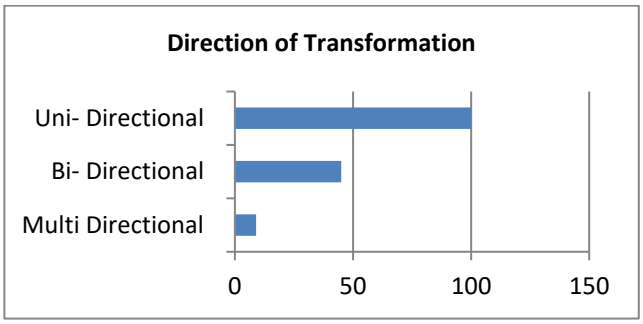

Fig. 10. Directions Percentage

The results show that all selected tools produce domain specific target model which mostly transform PSM to PSM and none produces domain independent target model transforming a PSM into PIM.

Vanilla Framework - Proposed Platform Independent Model

The proposed Vanilla Framework is a framework for Declarative User Interface with the backbone of Platform Independent Model. The source artifact user interface has been preserved. Different User Interface platforms differ in design and richness, the underlying functionality of the basic input and output element remain the same. The elements having similar functionality although have been given different class name in different market libraries. This transformation is then applied to some of the popular libraries such as SWING, HTML5, and more recent libraries of Android and Python-Tkinter.

In the Vanilla Model, 20 common elements of User Interface are identified with their characteristic structure like text box, check box, dialog box, radio button, button, audio. Model driven reverse engineering involves transformation of source model (PSM) from one platform specific meta-model to Vanilla Model (PIM). Model driven forward engineering involves transformation from Vanilla Model (PIM) to target model (PSM). The criteria for successful transformation should fulfill certain given below requirements.

i. The tool should be able to create, modify, retrieve and drop transformations.
ii. One can reutilize the transformation model defined for one transformation from source platform to target platform to other set of source and target platform.
iii. The transformation model must clearly define the termination condition.

iv. For each element in the source model, transformation must be complete.

VATE (Vanilla Transformation Engine) – A Transformation Tool based on Vanilla Framework

The proposed Vanilla transformation engine transforms from source PSM to abstract level PIM. For declarative user interface it then transforms to abstract level PIM to one or more target PSM. The proposed selected tool is XML due to its rich libraries. The bi-directionality is achieved by two ways. One by using the SQL queries for creating/ updating/ retrieving/deleting mappings and secondly by mapping the rules in the database table.

Different features of VATE are shown in table 1.

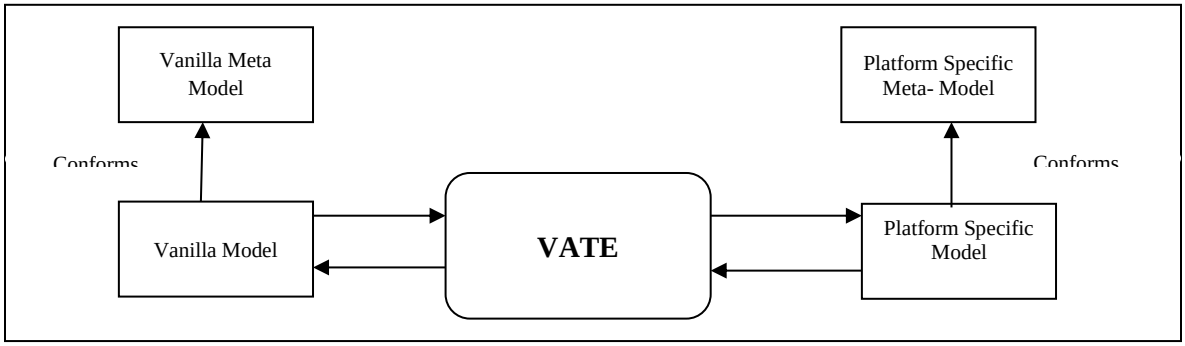

Fig. 11. Architectural Representation of VATE

Table 1
Features of VATE Tool

S. No.	Feature	Value
1.	Language	PHP & SQL
2.	Modeling Language	XML
3.	Compatibility with Standards	XML and supported technologies. Relational Database Schema
4.	Target Model	Constructive
5.	Cardinality	one-to-one (1-1), one-to-many (1-N)
6	Type of Transformation	Exogenous

7	Directions	Bi- Directional
8	Rules for Mapping	Relational Database Schema
9	CRUD	Supports CRUD
10	Level of Automation	Semi-Automatic

An XML based meta-model is defined containing model name, model elements and their properties. The relational database schema defines the mapping of elements from one model to another model. Design architecture of the proposed system is shown in figure 11.

One of the major feature of the proposed tool is the bi-directionality achieved through of relational database schema. This feature is not available in any other available tools. It consists of three major modules.

i. Meta-model
ii. Model
iii. Model Transformation Engine -VATE

VATE - The Model transformation engine is the core of this system which performs major functionalities. This module performs the transformation from PSM to PIM and visa-versa. Required inputs shall be taken from the user. From the relational database scheme proposed approach will read transformation rules one-by-one during transformation.

5. Conclusion

An extensive analysis of the existing meta-model based transformation tools is done for the declarative model using an exhaustive criterion. Results show that all selected tools produce domain specific target model which mostly transform PIM to PSM and none produces domain independent target model transforming a PSM into PIM. Proposed Vanilla transformation engine framework is a platform independent model based framework for declarative user interface based on XML technology and relational database schema. It is a bi-directional tool, i.e. it is able to transform the source model into the target model and target model back into the source model.

References

[1] https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics (accessed July 28, 2020)
[2] D. Zhang, Web content adaptation for mobile handheld devices, Communications of the ACM. 50 (2007) 75-79.
[3] S.Y. Hui, Challenges in the migration to 4G mobile systems, IEEE Communications magazine. 41 (2003) 54-59.
[4] J. Krogstie, Model-Based Development and Evolution of Information Systems – A Quality Approach, Springer, London 2012, https://doi.org/10.1007/978-1-4471-2936-3.
[5] J. Greenfield, K. Short, S. Cook, S. Kent, Software factories: assembling applications with patterns, models, frameworks, and tools, Wiley, 2004.

[6] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L.C. Kats, et al., DSL engineering: designing, implementing and using domain-specific languages, Createspace Independent Publishing Platform, 2013.

[7] D. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, Wiley, 2003.

[8] C. Atkinson, T. Kühne, Model-driven development: a metamodeling foundation, IEEE Software. 20 (2003) 36-41.

[9] B. Selic, Personal reflections on automation, programming culture, and model-based software engineering, Autom Softw Eng. 15 (2008) 379-91.

[10] O. Nierstrasz, S. Ducasse, and T. Gˇırba, The story of moose: An agile reengineering environment, in Proc. 10th Eur. Softw. Eng. Conf. Jointly. (2005) 1-10.

[11] M. Zanoni, F. Arcelli Fontana, F. Stella, On applying machine learning techniques for design pattern detection, J. Syst. Softw. 103 (2015) 102–117, http://dx.doi. org/10.1016/j.jss.2015.01.037.

[12] R. C. Holt, A. Winter, A. Schurr, GXL: Toward a standard exchange format, in Proc. 7th Work. Conf. Reverse Eng. (2000) 162-171.

[13] H. A. Müller, M. A. Orgun, S. R. Tilley, J. S. Uhl, A reverse engineering approach to subsystem structure identification, J. Softw. Maintenance, Res. Pract. 5 (1993) 181-204, http://dx.doi.org/10.1002/smr.4360050402.

[14] R. Kazman, L. O'Brien, and C. Verhoef, Architecture reconstruction guidelines, Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2002-TR-034. (2003), http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6255.

[15] H. Brunelière, J. Cabot, G. Dupé, F. Madiot, MoDisco: A model driven reverse engineering framework, Inf. Softw. Technol. 56 (2014) 1012–1032.

[16] O. Group, MDA Guide Revision 2.0, 2014, http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.

[17] T. J. Harmer, F. G. Wilkie, An extensible metrics extraction environment for object-oriented programming languages, Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation. (2002) 26-35.

[18] D. Hovemeyerand, W. Pugh, Finding bugs is easy, ACM SIGPLAN Notices, 39 (2004).

[19] T. Kamiya, S. Kusumoto, K. Inoue Ccfinder, A multi linguistic token based code clone detection system for large scale source code, Software Engineering, IEEE Transactions on. 28 (2002) 654-670.

[20] E. Bruneton, R. Lenglet, T. Coupaye, Asm: a code manipulation tool to implement adaptable systems, Adaptable and extensible component systems. 30 (2002).

[21] R. Shaham, E.K. Kolodner, M. Sagiv, Heap profiling for space - efficient Java, In ACM SIGPLAN Notices, 36 (2001) 104-113.

[22] Dahn, S. Mancoridis, Using program transformation to secure C programs against buffer overflows, In Proceedings of WCRE. 3 (2003) 323.

[23] D. B. Loveman, Program improvement by source-to-source transformation, Journal of the ACM (JACM). 24 (1977) 121-145.

[24] G. Kniesel, H. Koch, Static composition of re-factorings, Science of Computer Programming. 52 (2004) 9-51.

[25] Smita Agarwal, Alok Agarwal, Model driven reverse engineering of user interface - A comparative study of static and dynamic model generation tools, International Conference on Parallel, Distributed and Grid Computing. (2014) 268-273.
[26] A. Kleppe, J. Warmer, W. Bast, MDA Explained, The Model - Driven Architecture: Practice and Promise. Addison Welsey, 2003.
[27] F. Arcelli Fontana, S. Masiero, C. Raibulet, Elemental design patterns recognition in java, in Proc. 13th Int. Workshop Softw. Technol. Eng. Pract. (STEP), Budapest, Hungary. (2005) 196-205.
[28] F. Arcelli Fontana, S. Maggioni, C. Raibulet, Design patterns: A survey on their micro-structures, J. Softw., Evol. Process. 25 (2013) 27-52, http://dx.doi.org/10.1002/smr.547.
[29] F. Arcelli Fontana, F. Perin, C. Raibulet, S. Ravani, JADEPT: Dynamic analysis for behavioral design pattern detection, in Proc. 4th Int. Conf. Eval. Novel Approaches Softw. Eng. (ENASE), Milan, Italy. (2009) 95-106.
[30] F. Arcelli Fontana, R. Roveda, M. Zanoni, C. Raibulet, R. Capilla, An experience report on detecting and repairing software architecture erosion, in Proc. 13th Work. IEEE/IFIP Conf. Softw. Arch. (WICSA), Venice, Italy. (2016) 21-30.
[31] F. Arcelli Fontana, S. Maggioni, C. Raibulet, Understanding the relevance of micro-structures for design patterns detection, J. Syst. Softw. 84 (2011) 2334-2347, http://dx.doi.org/10.1016/j.jss.2011.07.006.
[32] F. Arcelli Fontana, S. Masiero, C. Raibulet, F. Tisato, A comparison of reverse engineering tools based on design pattern decomposition, in Proc. 16th Austral. Softw. Eng. Conf. (ASWEC), Brisbane, Australia. (2005) 262-269, http://dx.doi.org/10.1109/ASWEC.2005.5.
[33] C. Arevalo, M.J. E. Cuaresma, I. M. Ramos, M. Domínguez-Muñoz, A metamodel to integrate business processes time perspective in BPMN 2.0, Inf. Softw. Technol. 77 (2016) 17–33.
[34] V. Ovchinnikova, E. Asnina, The algorithm of transformation from UML sequence diagrams to the topological functioning model, in Proc. 10th Int. Conf. Eval. Novel Approaches Softw. Eng. (ENASE), Barcelona, Spain. (2015) 377-384.
[35] F. Trias, V. Castro, M. López-Sanz, E. Marcos, Reverse engineering applied to CMS-based Web applications coded in PHP: A proposal of migration, in Proc. 8th Int. Conf. ENASE, Angers, France. (2013) 241-256.
[36] F. Trias, V. de Castro, M. López-Sanz, E. Marcos, An ADM-based method for migrating CMS-based Web applications: Extracting ASTM models from php code, in Proc. 1st Int. Workshop Softw. Evol. Modernization (SEM), Angers, France. (2013) 85-92.
[37] F. Trias, V. de Castro, M. López-Sanz, E. Marcos, An ADM-based method for migrating CMS-based Web applications, in Proc. 25th Int. Conf. Softw. Eng. Knowl. Eng. (SEKE), Boston, MA, USA. (2013) 256-261.
[38] F. Trias, V. de Castro, M. López-Sanz, E. Marcos, Migrating traditional Web applications to CMS-based Web applications, Electron. Notes Theor. Comput. Sci. 314 (2015) 23-44.