A New PHP Discoverer for Modisco

Abdelali Elmounadi[1], Nawfal El Moukhi[2], Naoual Berbiche[3], Nacer Sefiani[4]
University Mohammed V, Rabat, Morocco[1, 3, 4]
University Ibn tofail in Kenitra, Kenitra, Morocco[2]

***Abstract*—MoDisco is an Eclipse Generative Modeling Technologies project (GMT Project) intended to make easier the design and building of model-based solutions that are dedicated to legacy systems Model-Driven Reverse Engineering (MDRE). It offers an open source, generic and extensible MDRE framework. Indeed, MDRE applies of Model-driven Engineering (MDE) principles to enhance traditional Reverse Engineering processes, and thus facilitate their understanding and manipulation. In the same context, the Architecture-Driven Modernization (ADM) is an OMG (Object Management Group) standard, which addresses the integration of MDA (Model-driven Architecture) and Reverse Engineering in the aim of understanding and evolving existing software assets. Thus, Modisco succeeded to stand out as the implementation reference in the MDRE and ADM field. Currently, Modisco handles only some technologies, such as Java and XML. Unfortunately, no adapted way to handle PHP (Hypertext Preprocessor) web-based projects by Modisco is available so far. This paper proposes a new model discovery tool intended for PHP language. This latter constitutes an extension for the Modisco framework that allows managing the applications assets written in PHP language. Thus, this work aims at enhancing the Modisco platform capabilities in managing more software development technologies.**

***Keywords*—MDRE; ADM; modisco; model discovery; PHP**

I. Introduction

Reverse Engineering still remains a challenging field in software engineering, notably because of the unceasing need to adapt to the continuous evolution of IT development. In fact, every organization needs to periodically reevaluate and evolve its company policies, because policies and rules must be aligned at all times, but unfortunately, this remains a challenging task [1]. In this context, Model Driven Reverse Engineering (MDRE) is a widely used approach that aims to enhance traditional Reverse Engineering processes [2]. It provides several technics based on the Model Driven Engineering (MDE) principles to allow modeling structures recovery from code-legacy, in order to facilitate its comprehension and manipulation. Among the various tools that have emerged for this purpose, MoDisco is an Eclipse GMT (Generative Modeling Technologies) project designed for the model discovery area. This tool is intended to make easier the design and building of model-based solutions dedicated to legacy systems reverse engineering [3]. However, MoDisco tool actually supports few technologies. For instance, it does not offer any possibility to handle PHP web-based applications despite the importance of this language in the web development area.

In this paper, the authors propose a new model discovery tool intended for PHP language as a PHP Discoverer integrated to the Modisco platform, in order to allow the model discovery of PHP-based web applications. The rest of this paper is organized as follow: Section 2 presents the research background. It presents all concepts related to MDRE and ADM with a presentation of the Modisco framework and its contribution in the model discovery area. Section 3 presents the adopted methodology in this work to achieve the contribution. Section 4 gives an experimentation case study to validate the congruency of the new model discovery tool. Finally, Section 5 presents the conclusion and the future works.

II. Research Background

A. Model-Driven Reverse Engineering

Generally, Reverse Engineering (RE) is about switching from the implementation heterogeneity technologies to the homogeneous world of models. It constitutes the process of comprehending software systems and producing models in a higher level of abstraction, suitable for documentation, maintenance, and reengineering. However, this process could suffer from two main disadvantages: for large-scale projects, it is difficult to predict time cost of the RE process. In addition, no standards are available to evaluate the quality of the obtained results [4]. Thus, MDRE is introduced to overcome these difficulties. This approach uses the modelling features and applies those features in the RE processes to overcome the problems cited above. In fact, with the current growing adoption of Model Driven Engineering (MDE) principles and techniques (where models are considered as first class entities in the whole development process) [5], several opportunities are presented for getting all of the benefits of MDE approach when designing new reverse engineering solutions.

MDRE is based on two systematic and consecutive phases as shown in Fig. 1, “Model Discovery” and “Model Understanding” [6]:

- Model discovery: This step consists in obtaining a model that represents a legacy system from its source code, data sets, documentation, etc. The obtained model conforms to a given metamodel that can be, according to the needs, technology-specific or more generic. Therefore, the model discovery is generally realized via components called “discoverers”. A discoverer can have various and varied natures depending on the type of system subject of reverse engineering. It can be either fully hardcoded or partially generated using model transformations combining the corresponding metamodels.
- Model Understanding: Most MDRE applications require the processing of the models discovered in the

Model discovery phase in order to obtain higher-level views of the legacy systems that facilitate their analysis, comprehension, and later reuse. Thus, this phase is called model understanding. Chains of model manipulation techniques are employed to query and transform the models obtained following the model discovery phase into more manageable representations, by omitting details that are not relevant for the MDRE scenarios.

B. *Architecture-Driven Modernization*

Architecture-Driven Modernization is the process of understanding and evolving existing software assets. According to [7], ADM is an OMG (Object Management Group) standard that addresses the integration of MDA and reverse engineering.

MDA encourages the separation of concerns, i.e. it preconizes the model transformations between different levels of abstraction, beginning with platform independent models (PIMs) which do not contain any specific information about the implementation platform, arriving to platform specific models (PSMs) that include specific information about implementation platforms. In fact, ADM is for MDRE what is MDA for MDE. It also preconizes the use of PIM, PSM and model transformations [8] concept to facilitate the systematic analysis of existing systems to gather their corresponding models (Fig. 2).

With the advent of ADM, OMG presented a new set of metamodel relatively to this context: Knowledge Discovery Metamodel (KDM) [9] and Software Metrics Metamodel (SMM) [10], and ASTM (Abstract Syntax Tree Metamodel) [11].

ASTM is a metamodel from the OMG that describes the set of elements used for composing abstract syntax trees. The purpose of ASTM is to provide a framework that allows common interchange of abstract syntax models of software based upon modeling specifications. ASTM serves as a universal high-fidelity gateway for modeling code at the most fundamental syntactic level. Thus, ASTM respects the scope of KDM and UML for modeling the semantics of higher-level software concepts and includes only the most basic semantics associated with code. The ASTM specification is organized into three levels of abstraction:

- GASTM: Generic Abstract Syntax Tree Metamodel is a generic set of language modeling elements common across numerous languages establishes a common core for language modeling, called the Generic Abstract Syntax Trees. In this specification, the GASTM model elements are expressed as UML class diagrams.
- SASTM: Language Specific Abstract Syntax Tree Metamodels constitute a set of metamodels for particular languages such as PHP, C++ or Java. These metamodels are derives from the GASTM along with modeling element extensions sufficient to capture the language. Fig. 3 illustrates the existing relationship between the GASTM level and the SASTM level.PASTM: Proprietary Abstract Syntax Tree Metamodels express AST representations for different languages modeled in formats that are not consistent with MOF (Meta-Object Facility), the GASTM, or SASTM. For such proprietary AST this specification defines the minimum conformance specifications needed to support model interchange.

C. *Modisco GMT Project*

MoDisco is an Eclipse Generative Modeling Tool (GMT), which provides an extensible and customizable MDRE framework to develop model-driven tools supporting different model driven reverse engineering scenarios such as legacy migration or modernization, quality assurance, re-documentation, etc. The main purpose of MoDisco is to offer an open source, generic and extensible MDRE framework (Fig. 4). Considering as inputs miscellaneous legacy artifacts (source code, databases, configuration files, documentation, etc.), MoDisco aims to providing the required functionalities for creating models and allowing their handling, analysis and computation. Afterwards, the framework targets the production of different types of artefact as outputs, depending on the selected MDRE objectives (source code, data, metrics, documentation, etc.).

Furthermore, MoDisco is an Eclipse-based project that provides and uses concrete implementations of three OMG standard meta-models: KDM, SMM and ASTM.

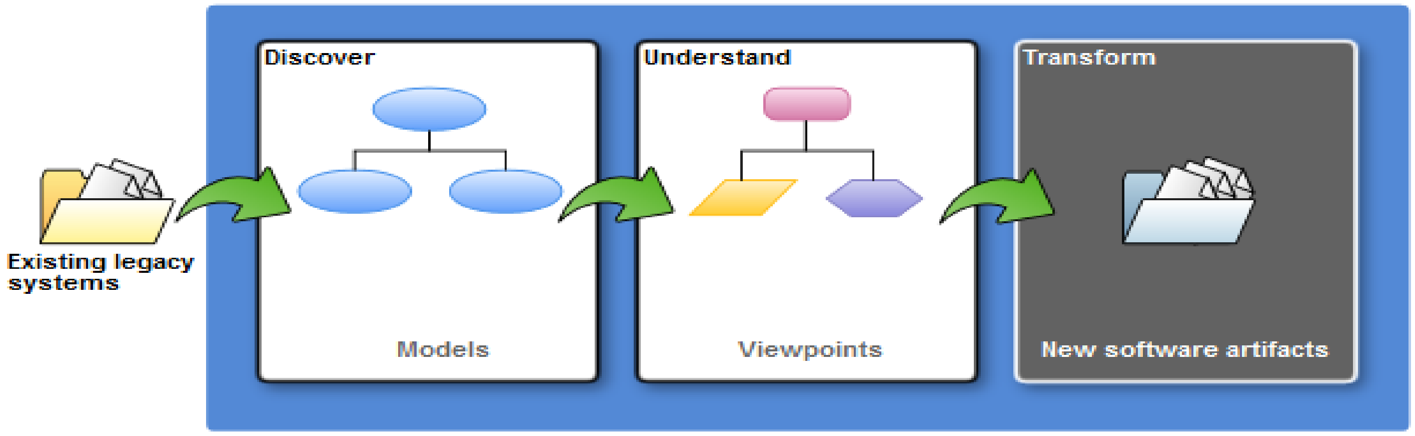


Fig. 1. MDRE Process.

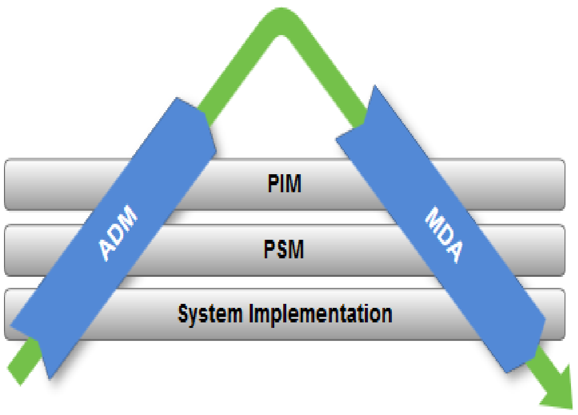


Fig. 2. Process for Evolving Existing Software Assets using ADM/MDA Approaches.

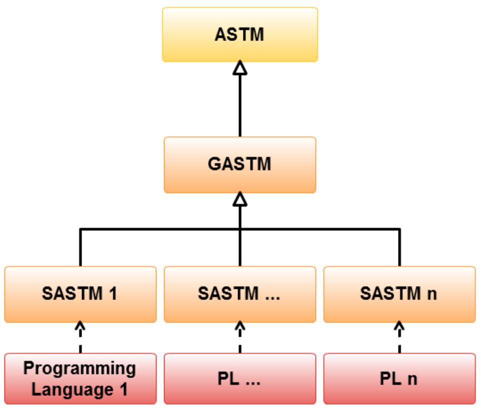


Fig. 3. SATSM - GASTM Relationship.

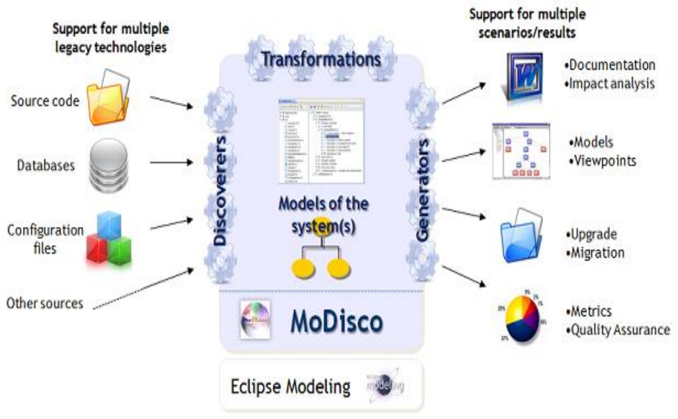


Fig. 4. The Modisco Framework.

Currently, Modisco offers extended technology specific support for XML model driven reverse engineering (intended for some JEE frameworks configuration files such as Struts) and Java language model driven reverse engineering (including a full Java language meta-model and a Java discoverer) only. Nevertheless, several other technologies are still not integrated in the Modisco project like PHP language. Therefore, the paper proposes a model discovery plugin as an extension for Modisco framework to allow supporting the model discovery of PHP web based legacy systems (Fig. 5).

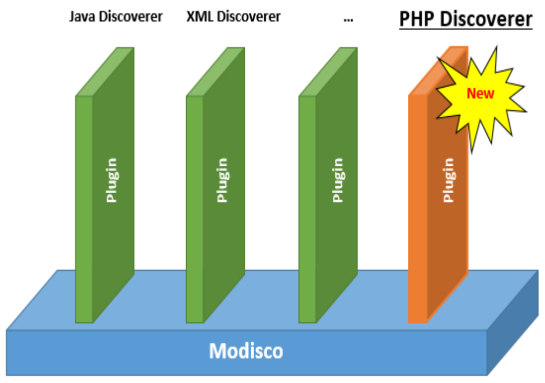


Fig. 5. Modisco Plugins Organization.

III. New PHP Discoverer

The main purpose of this work is to be able to apply model discovery process on existing PHP web-based application. To achieve this, the authors made a PHP metamodel and a dedicated discovery tool. Fig. 6 describes the employed model discovery process.

A previous work has covered the same issue related to the Java language [12]. As known, the Eclipse IDE constitutes an extensible development environment that supports a wide range of programming languages. This ability is provided to the Eclipse platform through artefacts called “Development Tools”. These development tools offer integrated development environments based on the Eclipse platform. Features include support for project creation, managed build for various toolchains, source navigation, various source knowledge tools, syntax coloration, source code refactoring, code generation and visual debugging tools for the given language. JDT [13] (Java Development Tools), PDT [14] (PHP Development Tools) and CDT [15] (C/C++ Development Tools) are some of the available development tools used with the Eclipse platform.

First, based on the Eclipse implementation of the PHP language through PDT (PHP Development Tools), the authors were able to establish a PHP metamodel by using EMF-Ecore [16]. Fig. 7 illustrates a part of the PHP metamodel hierarchy.

Then, the model discovery process is started by extracting the AST (Abstract Syntax Tree) from the source code provided as input. At this stage, the AST nodes are visited based on the visitor design pattern [17]. In fact, for each class that composes the PHP metamodel, the implemented visitor provides two main methods: visit and endVisit. The visit method is invoked once an instance of the concerned class is reached. Then, at the end of the element visit, the PHP node is mapped to a model discovery node with all its relative attributes.

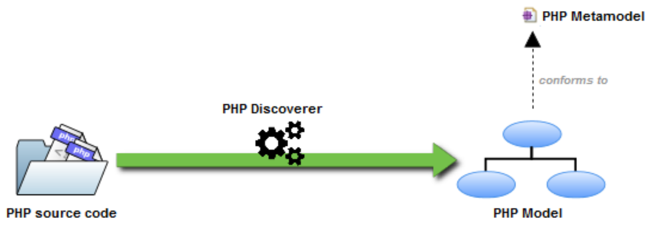
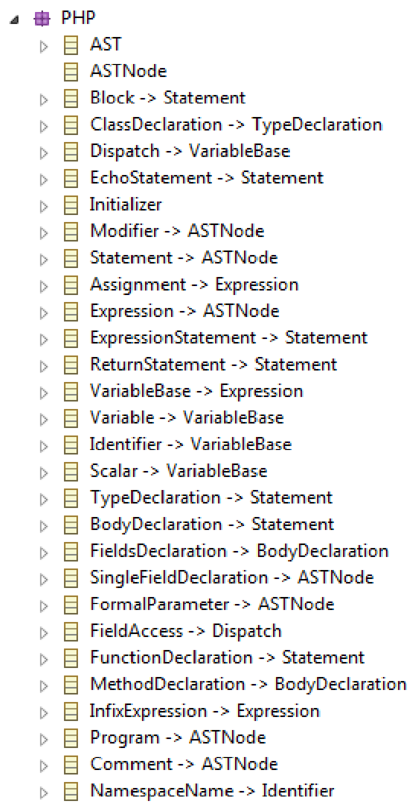


Fig. 6. The PHP Model Discovery Process.



```
PHP
  AST
  ASTNode
  Block -> Statement
  ClassDeclaration -> TypeDeclaration
  Dispatch -> VariableBase
  EchoStatement -> Statement
  Initializer
  Modifier -> ASTNode
  Statement -> ASTNode
  Assignment -> Expression
  Expression -> ASTNode
  ExpressionStatement -> Statement
  ReturnStatement -> Statement
  VariableBase -> Expression
  Variable -> VariableBase
  Identifier -> VariableBase
  Scalar -> VariableBase
  TypeDeclaration -> Statement
  BodyDeclaration -> Statement
  FieldsDeclaration -> BodyDeclaration
  SingleFieldDeclaration -> ASTNode
  FormalParameter -> ASTNode
  FieldAccess -> Dispatch
  FunctionDeclaration -> Statement
  MethodDeclaration -> BodyDeclaration
  InfixExpression -> Expression
  Program -> ASTNode
  Comment -> ASTNode
  NamespaceName -> Identifier
```

Fig. 7. SASTM of PHP Language.

As mentioned above, Modisco is an extensible tool, i.e. it offers an API for integrating new model discovery tools. Therefore, this API shows the relevant steps to declare a new discoverer. The framework defines a Java interface *“org.eclipse.modisco.infra.discovery.core.IDiscoverer<T>”* that every discoverer has to implement [18]:

```
public interface IDiscoverer<T> {
      boolean isApplicabeTo(T source);
      void discoverElement(T source,
   IProgressMonitor monitor) throws
   DiscoveryException;}
```

T represents the java type for the source of the discovery. The *isApplicableTo* method specifies if the source object could be handled by the discoverer. For example, for the end user, if the discoverer manages the selected source object, a discoverer menu will be available in the pop-up menu by clicking with the contextual button (Fig. 8). There are 3 types of source objects: *IProject* for projects, *IFolder* for folders, and *IFile* for files. In the current study, the discoverer is applied on a project of PHP Nature. The *discoverElement* method is a generic method for performing a model discovery from the source object. The service may throw some discovery exceptions (a class *DiscoveryException* instance).

Finally, the model serialization is performed after selecting the associated parameters. In this manner, the process provides an XML Metadata Interchange [19] (XMI) representation of the PHP discovered model from the source code project provided as input.

IV. Experimentation

In order to validate the current contribution, the new discoverer was tested on several PHP projects. The following example represents a simple PHP Math class contained in a PHP project, and that contains a static member with a function of adding two variables. A more complex example could have been presented, but the interest of this section is to show the enforceability of the method without occupying a large space in the article.

```
<?php
class Math {
 public static final $PI = 3.14159265359;
 public function add($a, $b) {
  return $a + $b;
 }
}
?>
```

By applying the model discovery process using the implemented PHP discoverer on the example shown above, the authors obtain the XMI serialization of the discovered model (corresponding to the PHP metamodel). Fig. 9 illustrates the obtained result from the Modisco model browser view.

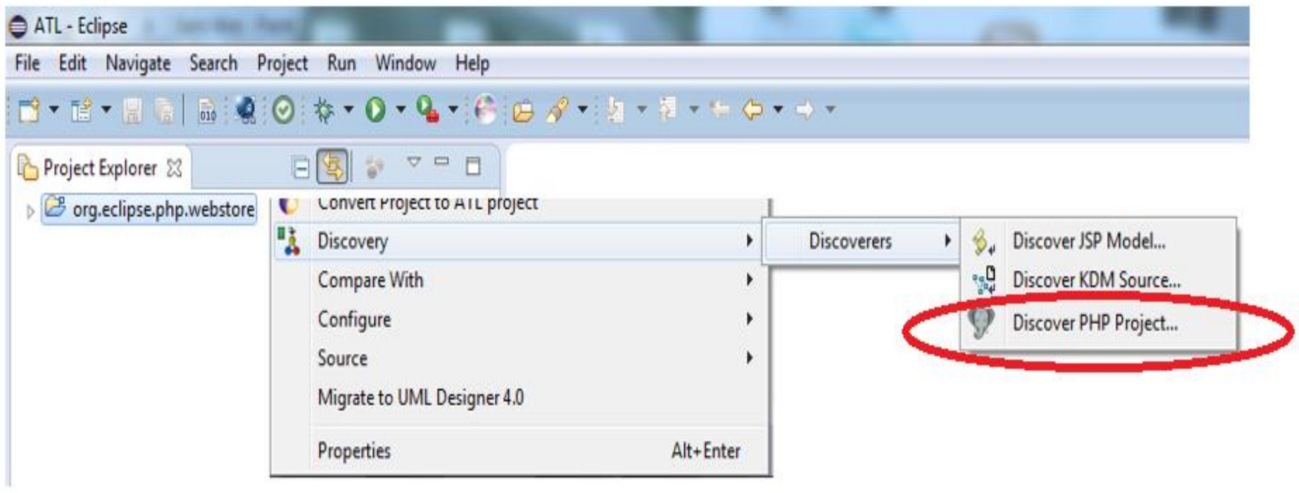
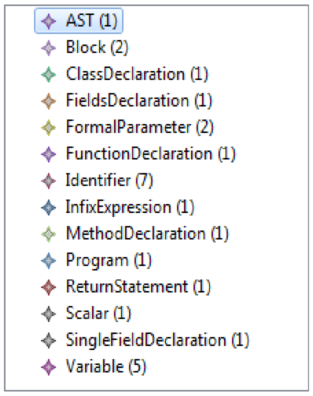


Fig. 8. The New PHP Discoverer in Action.



AST (1)
Block (2)
ClassDeclaration (1)
FieldsDeclaration (1)
FormalParameter (2)
FunctionDeclaration (1)
Identifier (7)
InfixExpression (1)
MethodDeclaration (1)
Program (1)
ReturnStatement (1)
Scalar (1)
SingleFieldDeclaration (1)
Variable (5)

Fig. 9. Modisco Model Browser view of the Obtained Result.

From the XMI source view, the representation of the obtained model is as follows:

```
<?xml version="1.0" encoding="ASCII"?>
<php:AST xmi:version="2.0"
  xmlns:xmi=http://www.omg.org/XMI
  xmlns:xsi="http://www.w3.org/2001/XMLS
  chema-instance"
  xmlns:php="http://eclipse.org/gmt/modi
  sco/php/incubation/beta">
<program>
<statement
  xsi:type="php:ClassDeclaration"
  modifier="none">
<identifier name="Math"/>
<body isCurly="true">
<statement
  xsi:type="php:FieldsDeclaration"
  modifier="public static">
<field
  xsi:type="php:SingleFieldDeclaration">
<variableName xsi:type="php:Variable"
  isDollared="true">
<name xsi:type="php:Identifier"
  name="PI"/>
</variableName>
<value xsi:type="php:Scalar"
  value="3.14159265359"/>
</field>
</statement>
<statement
  xsi:type="php:MethodDeclaration"
  modifier="public">
<function>
<identifier name="add"/>
<body isCurly="true">
<statement
  xsi:type="php:ReturnStatement">
<expression
  xsi:type="php:InfixExpression"
  operator="+">
<left xsi:type="php:Variable"
  isDollared="true">
<name xsi:type="php:Identifier"
  name="a"/>
</left>
<right xsi:type="php:Variable"
  isDollared="true">
<name xsi:type="php:Identifier"
  name="b"/>
</right>
</expression>
</statement>
</body>
<formalParameter>
<parameterName xsi:type="php:Variable"
  isDollared="true">
<name xsi:type="php:Identifier"
  name="a"/>
</parameterName>
</formalParameter>
<formalParameter>
<parameterName xsi:type="php:Variable"
  isDollared="true">
<name xsi:type="php:Identifier"
  name="b"/>
</parameterName>
</formalParameter>
</function>
</statement>
</body>
</statement>
</program>
</php:AST>
```

In this manner, the obtained XMI file can easily be used in M2M [20] model transformation processes, in a model-understanding context.

V. Conclusion and Future Works

This paper presented a new model discovery tool intended for PHP language. Based on the Eclipse platform, especially via PDT and EMF-Ecore, the authors were able to implement a PHP Ecore metamodel, which constitutes a building block of the model discovery of PHP legacy systems. In this manner, the authors were able to add value to the Modisco platform and meet a crucial need for the use of this framework. The authors were also able to answer a widely asked question in the online forums, mostly by engineering students, about the existence of a model discovery tool dedicated to the PHP language. In future works, the authors aim to integrate other programming languages using the same approach, to enhance the possibilities of model discovering existing systems in other languages and technologies.

References

[1] V. Cosentino, J. Cabot, P. Albert, P. Bauquel, and J. Perronnet, “A Model Driven Reverse Engineering Framework for Extracting Business Rules out of a Java Application,” in RuleML, Montpellier, France, 2012.

[2] A. Elmounadi, N. Berbiche, F. Guerouate, and N. Sefiani, “Smart Text to Model Transformation a Graph Based Approach to Cover Dynamic Analysis,” Int. Rev. Comput. Softw. IRECOS, vol. 11, no. 4, p. 344, Apr. 2016.

[3] H. Brunelière, J. Cabot, G. Dupé, and F. Madiot, “MoDisco: A model driven reverse engineering framework,” Inf. Softw. Technol., vol. 56, no. 8, pp. 1012–1032, 2014.

[4] S. Rugaber and K. Stirewalt, “Model-driven reverse engineering,” Softw. IEEE, vol. 21, no. 4, pp. 45–53, 2004.

[5] F. Tomassetti, M. Torchiano, A. Tiso, F. Ricca, and G. Reggio, “Maturity of software modelling and model driven engineering: A survey in the Italian industry,” 2012.

[6] M. Brambilla, J. Cabot, and M. Wimmer, Model-Driven Software Engineering in Practice, 1st ed. Morgan & Claypool Publishers, 2012.

[7] J.-N. Mazón and J. Trujillo, “A model driven modernization approach for automatically deriving multidimensional models in data warehouses,” in International Conference on Conceptual Modeling, 2007, pp. 56–71.

[8] Y. Rhazali, Y. Hadi, and A. Mouloudi, “A model transformation in MDA from CIM to PIM represented by web models through SoaML and IFML,” in 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), 2016, pp. 116–121.

[9] Object Management Group, “Knowledge Discovery Metamodel (KDM).” [Online]. Available: http://www.omg.org/technology/kdm/. [Accessed: 24-Apr-2018].

[10] Object Management Group, “About the Structured Metrics Metamodel Specification Version 1.1.1.” [Online]. Available: https://www.omg.org/spec/SMM/1.1.1/. [Accessed: 24-Apr-2018].

[11] Object Management Group, “About the Abstract Syntax Tree Metamodel Specification Version 1.0.” [Online]. Available: https://www.omg.org/spec/ASTM/1.0/. [Accessed: 24-Apr-2018].

[12] Eclipse Foundation, “https://wiki.eclipse.org/MoDisco/JavaAbstractSyntax,” Java Abstract Syntax Discovery Tool, 21-Jan-2018. [Online]. Available: https://wiki.eclipse.org/MoDisco/JavaAbstractSyntax.

[13] A. Elmounadi, N. Berbiche, F. Guerouate, and N. Sefiani, “Eclipse JDT-based method for dynamic analysis integration in Java code generation process,” J. Theor. Appl. Inf. Technol., vol. 95, no. 24, 2017.

[14] “PHP Development Tools,” PHP Development Tools. [Online]. Available: https://insight.io/github.com/eclipse/pdt/tree/master. [Accessed: 21-Oct-2017].

[15] Eclipse Foundation, “Eclipse CDT (C/C++ Development Tooling).” [Online]. Available: https://www.eclipse.org/cdt/. [Accessed: 21-Jan-2018].

[16] H. Kern and S. Kühne, “Model interchange between aris and eclipse emf,” in 7th OOPSLA Workshop on Domain-Specific Modeling at OOPSLA, 2007, vol. 2007.

[17] S. J. Metsker and W. C. Wake, Design patterns in java. Addison-Wesley Professional, 2006.

[18] Eclipse Foundation, “Discovery Manager Developer Documentation,” Eclipse documentation. [Online]. Available: https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.modisco.infrastructure.doc%2Fmediawiki%2Fdiscovery_manager%2Fplugin_dev.html. [Accessed: 22-Oct-2017].

[19] Object Management Group, “MOF 2 XMI Mapping, Version 2.4.” 2010.

[20] M. Rahmouni and S. Mbarki, “MDA-Based ATL Transformation To Generate MVC 2 Web Models,” Int. J. Comput. Sci. Inf. Technol., vol. 3, no. 4, pp. 57–70, Aug. 2011.