Architecture Driven Modernization: A Review on Reverse Engineering Techniques based on Models' Approach

MOHAMED KARIM KHACHOUCH[1], AYOUB KORCHI[1], YOUNES LAKHRISSI[2]
[1]SIGER Laboratory FST of Fez,
Sidi Mohamed Ben Abdellah University,
MOROCCO

[2]SIGER Laboratory ENSA of Fez,
Sidi Mohamed Ben Abdellah University,
MOROCCO

Abstract: - Software specifications represent one of the risks that can cause a project to fail if they tend to be modified during development. it is a problem that all companies with an information system or developing software can face regardless of the latter's size. Specification techniques have indeed evolved over the last few years to avoid this type of situation as much as possible. Nevertheless, one can never predict a client's evolutionary needs. To remedy this problem, there is a solution that we consider effective, which is reverse engineering. Reverse engineering is not a new term. Originally, reverse engineering meant analyzing hardware to improve it in the case of a proprietary product or to detect its strengths in the case of a competing product. By projecting these concepts onto the software, we conclude that the goal is to fully understand the system and its structure. And if the goal of reverse engineering on hardware is to duplicate the system, the goal on software is to understand its design for maintenance and support purposes.

Key-Words: - MDRE, model-driven reverse engineering, model-driven engineering, models, transformations, legacy system

Received: July 23, 2022. Revised: August 24, 2023. Accepted: September 25, 2023. Published: October 11, 2023.

1 Introduction

Recently, a shift from low abstraction development paradigms such as oriented object paradigm to high abstraction development paradigms such as MDE (model-driven development) is noticeable, [1]. MDE aims to organize all levels of abstraction and methodologies. It encourages developers to use models to describe both the problem and its solution at different levels of abstraction and provides a framework for methodologists to define what model to use at a given moment (i.e., at a given level of abstraction), and how to lower the level of abstraction by defining the relationship between the participating models, [2]. In the literature, many authors proposed some forward engineering approaches where specific models can be turned into source code. This adds more ease to the development process. But the problem to solve is that applications are often not developed from scratch and reverse engineering becomes a necessity to understand the software process by the mean of models with a high level of abstraction, easy to document, evolve and maintain. The use of MDE in reverse engineering is called MDRE (Model-Driven Reverse Engineering). Even if the approaches are numerous and differ between them, we distinguish two general stages in model-driven reverse engineering. The first one is the analysis of the legacy system and describes it as a model. The second one is the exploitation of that model to modernize the system or generate documentation. Nearly all approaches are based on source code as initial data or initial source of knowledge as it is called in the MDRE jargon. Model-to-model and text-to-model transformations can be automatic or semi-automatic in case a refinement is required.

This interest in model-driven reverse engineering encourages the OMG to set standardized metamodels that can help to modernize projects by creating the ADMTF (Architecture Driven Modernization Task Force).

In this paper, we analyze some approaches mentioned in the literature to put light on the current state of art of the model-driven reverse engineering, to collect ideas for developing our approach or choosing the best one and upgrading it.

The paper is organized as follows. In section II we present the model-driven reverse engineering approaches that we could describe completely and analyze them by certain criteria to conclude the analysis with the best approach between them before a general conclusion in Section III where we summarize our work and talk about our future works.

2 Model Driven Reverse Engineering

2.1 Introduction
In this section, we analyze the approaches we found in the literature relative to model-driven reverse engineering.

2.1 MDRE Approaches
In this section, we identified most of the model-driven reverse engineering approaches and analyze them.

These approaches can be characterized by some main concepts which are discovery, metamodeling, and transformation. Discovering mechanisms on hand help to find models automatically from the legacy system. Those models are called PSM in MDA jargon, [3]. Model transformation on the other hand helps to build other models with higher abstraction levels. Those models are called PIM in MDA jargon, [4]. By analyzing the literature, we may summarize the main steps of model-driven reverse engineering as follows:

- Discovery of legacy system and extraction of PSM basic models
- Knowledge of the necessary information using those basic models
- Calculation of views using the extracted information
- Result recuperation is described in derived models.

Therefore, we present in this section the MDRE approaches constituted by at least one PSM model, one PIM, and one
text-to-model and one model-to-model transformation.
For comparison purposes, we discuss each approach with the following elements:

- The models used: Each MDRE approach should follow platform-independent technologies (in this instance, metamodels and model-based components)
- The scope (generic/specific)
- Application fields of the approach
- Automation level (auto/semi)
- Type of analysis (static/dynamic)

In the following, we present the main metamodels used in MDRE then we present our selection of approaches. For each approach, we present a brief introduction to it, the models used in it, the transformation phases considering the automation level, and the tools used for it.

2.1.1 MDRE Normalized Models
To support MDRE, OMG defines a set of standardized models. The first one we're going to present is KDM.

Knowledge Discovery Metamodel aka KDM is by nature a Meta-Object Facility (MOF) model, [5]. It allows the definition of a set of concepts that will represent the foundations of a pattern language. Its main objective is to understand the legacy system in preparation for software modernization and provides the infrastructure to support domain-specific, application-specific, or implementation-specific knowledge definitions. The structure of KDM is as shown below in the figure:

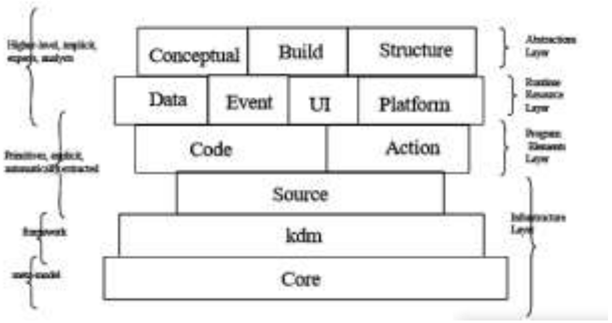

Fig. 1: KDM architecture

Figure 1 shows how KDM is arranged into a stack of packages where each one depends on one or more packages, and every other package relies firstly on the core package because that's where all the metamodel elements are defined, secondly on the kdm package because kdm model definition resides there.

Abstract Syntax Tree Metamodeling allows the ease of communication of system metadata between software development modernization tools, platforms, and metadata repositories in heterogeneous environments by describing the elements used to compose AST models, [6], [7]. An AST model is a model describing the structure of software statements to reflect the programming language grammar. We distinguish three domains of metamodels software artifacts:
Generic Abstract Syntax Tree Metamodel (GASTM): represents a generic set of language modeling elements common across numerous languages, [8].

Language Specific Abstract Syntax Tree Metamodels (SASTM): represents particular languages such as Ada, C, FORTRAN, and Java. [8] Proprietary Abstract Syntax Tree Metamodel (PASTM): expresses ASTs for languages such as Ada, C, COBOL, etc., modeled in formats inconsistent with MOF, the GSATM, or SASTM, [8]. The Abstract Syntax Tree Metamodel is presented in Figure 2.

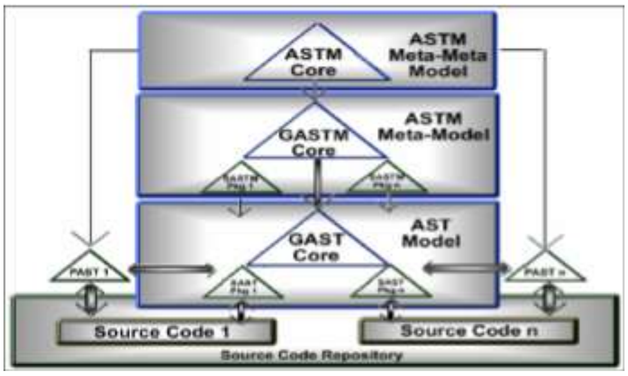

Fig. 2: Abstract Syntax Tree Metamodel

2.1.2 MDRE Literature Approaches Review

In the model-driven reverse engineering universe, we may classify an approach as whether it aims to reverse engineer the legacy system from a predetermined technology with a predefined scenario which is called in literature specific purpose solution, or the basis for any manipulation which is called a general-purpose solution.
In Table 1, we try to classify some literature approaches by those two classes:

Table 1. Classification of some MDRE approaches

Approach	Specific Reverse Engineering Solution	General Reverse Engineering Solution
Columbus, [9] [10].	✓	
JaMoPP, [11].	✓	
Spoon, [12].	✓	
ConQAT, [13].	✓	
GUPRO	✓	
SWAG Kit, [14].	✓	
CodeCity, [15].	✓	
CORUM, [16].		✓
Moose, [17].		✓
Rational Software Architect, [18].		✓
MagicDraw, [19].		✓

- In, [20]

MoDisco is an open-source Eclipse project where its authors tried to fulfill their vision of a good model-driven reverse engineering approach, [21]. A full MDRE approach must provide some characteristics such as being independent of any technology, but specific technology can be supported, can be extensible at the model or workflow level, can cover the whole system for the next steps of modernization, offer the possibility of reusing all resulted, internal and external components and automate the majority or even the whole process.
To reach those characteristics, MoDisco switches from step 1 from a heterogenous environment due to legacy systems to a homogenous environment of models without any information loss. Those resulting models are precise enough to start a MDRE scenario, but still at a very low level of abstraction. Then, the other steps of reverse engineering will only require those kinds of models. As a result, heterogeneity is reduced considerably and turned into a modeling problem.

For MoDisco, a simple equation represents the MDRE as follows: Model Driven Reverse Engineering = Model Discovery + Model Understanding, [22].

Model Discovery handles the representation of required information from the legacy system in the model without losing any required data. The advantage of MoDisco is that that phase is model-based 100%.

The model Understanding phase is in charge of exploiting the "discovered models" in the previous step to obtain by some model transformations the final result of the reversed system.
To resume this approach, we have established in Table 2:

Table 2. Summary of, [20] publication

MDRE models	MDRE steps	Tool Support
Extensible Metamodel provided based on OMG KDM	Model discovery Model Understanding	Offer pre-established specific metamodels for Java, JSP, XML

However, MoDisco has a limitation which is not supporting behavioral aspects from the legacy system, only structural aspects.

- In, [23]

The authors based their research on limitations of other existing tools such as MoDisco presented previously which are the absence of a framework that can handle both structural and behavioral models from legacy systems, [24], the absence of support to handle the extraction of behavioral aspects of legacy system, and the limitation for the extraction of structural aspects of legacy systems. Therefore, they came up with a tool called SRC2MOF (source-to-model framework), [23]. The latter is in charge of extracting structural aspects and representing them in the form of a class diagram, then the behavior aspects and representing them in the form of an activity diagram.

The architecture of SRC2MOF is composed of three main elements: A user interface allowing the user to input his source code. The IMD (Intermediate Model Discoverer) allows parsing Java classes into AST (abstract syntax tree) as a first generated model then generates an intermediate model in the form of an XML-like depiction of code source combining all stored MetaClasses. The model generator allows transforming the intermediate model into UML models, [23].

To resume this approach, we have established in Table 3:

Table 3. Summary of, [23] publication

MDRE models	MDRE steps	Tool Support
Java metamodel Domain Variable Model Business Rule Model	Model discovery Business rule identification Structural identification Business rules representation	ATL MoDisco

Even if SRC2MOF is fully automated, the user can refine the result after each processing of the framework if needed.

- In, [25]

This approach consists essentially of modernizing COBOL systems. The process of this approach can be resumed in identifying objects from legacy data by the following steps. First, the PSM metamodel is described which specifies the legacy system's data. Then, data are consumed from record files to create PSM models. After that, all those PSM models are merged into one Merge Model File Descriptor (MMFD) which is a model compliant with the previous PSM metamodel. That MMFD is then transformed into a PIM model, a domain class diagram to be precise. As a reminder, the domain class diagram is useful to show the different classes of a piece of software, their attributes, and methods. It provides an oriented object static view of the system. We note that the domain class diagram metamodel is predefined in this approach and respects the MOF specifications. The model-to-model transformations are performed using ATL which is a part of the OMG QVT requirements and based on OCL formalism. The final step of this approach is the domain class diagram refinement where the approach uses the legacy system data once again, [25].

To resume this approach, we have established in Table 4:

Table 4. Summary of, [25] publication

MDRE models	MDRE steps	Tool Support
Cobol Metamodel MMFD UML Domain class diagram	PSM metamodel Merge of PSM models Domain class diagram extraction and refinement	ATL

This approach unfortunately can only be used for COBOL systems for the moment.

- In, [26]

In the context of object-oriented programming, this approach proposes the extraction of use case diagrams, class diagrams, state diagrams, activity diagrams, and sequence diagrams from Java source code. The vision of the authors is to generate PSM and PIM models through static and dynamic analysis. PSM and PIM are expressed using UML diagrams and OCL, [27]. The process of this approach is as follows:

It first starts with the ISM (Implementation Specific Model) which is a result of the migration of the initial code into the oriented-object paradigm described in AST (Abstract Syntax Tree) according to a preestablished metamodel by performing a static analysis. Then a dynamic analysis is executed to complement the AST. Enriching the AST is essential in this approach to extract the PSM and then lift the abstraction of a latter to get the PIM in the form of UML diagrams, [28].

To resume this approach, we have established in Table 5:

Table 5. Summary of, [28] publication

MDRE models	MDRE steps	Tool Support
NEUREUS (for metamodels) UML OCL	Generation of AST Generation of PSM Generation of PIM	ATL

We note that this approach is generic even if the authors presented a solution for Java systems that isn't fully automated.

- In, [29]

This approach tried to focus on the quality of its results. Therefore, it's a semi-automatic approach, [29]. First, the legacy code must be parsed to generate the AST. The latter is the base of multiple model-to-model transformations that conform to an ANT metamodel to obtain the PIM. Those transformations are done by the mean of an engine called MIA (Model In Action). MIA is developed in three layers. The core engine is responsible for model transformations and code generation. The development environment is where the user interacts with the engine to design and implements model-to-model transformations and code generators. The user environment is represented in modules that can integrate IDEs easily. The resulting PIM which is in the form of an ANT model is transformed again into a PSM for the targeted platform. Finally, the code is generated based on the PSM model.
To resume this approach, we have established in Table 6:

Table 6. Summary of, [29] publication

MDRE models	MDRE steps	Tool Support
AST ANT UML	Generation of AST Transforming the AST into PSM Transforming the PSM into PIM	ATL

- In, [30]

This approach treats 3D X3D and JavaScript applications based on an MDD approach using SSIML (Scene Structure and Integration Modelling Language), a visual and UML-based language that provides support for the description of 3D user interface structures on an abstract design level. SSIML can also be based on DSL, [31]. SSIML is based on two different models that help to model a 3D scene on an abstract level. The first one is a scene model. It makes the modeling of 3D scene graphs possible. The second one is an interrelationship model. The latter, as its name suggests, helps to create associations between the elements of the scene model, [32]. This approach works as follows. First, by the mean of Xtext, ASTs are generated from JavaScript and X3D source code in the form of SSIML. Then, Those ASTs are transformed into an IM (intermediate model) by the mean of ETL (Epsilon Transformation Language). IM is a solution the authors come up with to solve the problem of non-simultaneous round-trip engineering for 3D development. As a reminder, round-trip engineering is an operating mode where reverse engineering and code generation are combined. After that, the IM is refined if modifications have been performed in the legacy code. Finally, the IM is converted to an SSIML abstract model by the mean of reflective API based on EMF.

To resume this approach, we have established in Table 7:

Table 7. Summary of, [31] publication

MDRE models	MDRE steps	Tool Support
SIMPLE IM X3D and JavaScript ASTs	Generation of ASTs from source code Generation of IM from ASTs Refinement of IM in the case of source code modifications Generation of SSIML abstract model from the IM	SSIML

We note that this approach has the advantage of being a general reverse engineering solution.

- In, [33]

This approach focuses on extracting business rules in legacy systems by using intermediate representation by the mean of KDM and raising the abstraction level of business rules, [34]. To proceed, this approach starts first with a preliminary study that consists of gathering needed data by reviewing the architecture of the legacy system and identifying the components of this architecture. The main objective of this preliminary study is to determine the strategy where the KDM is defined, [35]. After that comes the second step which is knowledge extraction. In this step, KDM models are generated representing the legacy system at multiple abstraction levels. Those levels are the infrastructure, runtime resources, program elements, and conceptual layers. The infrastructure layer is the model that represents the set of physical components of the legacy system such as containers, repositories, configuration files, etc. The program elements layer is the model that describes the legacy system's structure and behavior. Both the structure and behavior may be represented by AST models validated by an ASTM (Abstract Syntax Tree

Metamodel). The runtime resource layer is the set of models that represent data, UI, events, and platforms. Finally, there is the business logic abstraction step where the main objective is to separate the infrastructure model parts from the business logic implementation model parts within the KDM.

To resume this approach, we have established in Table 8:

Table 8. Summary of, [33] publication

MDRE models	MDRE steps	Tool Support
KDM Models	Extraction of the needed knowledge Generation of the KDM models Separation of the KDM models into infrastructure representation and business logic representation.	ATL KDM tools in Eclipse

We note that this approach is a generic semi-automatic one. In their future works, the authors are willing to represent the KDM model using UML to ease of use of their solution.

- In, [36]

This approach is an application of the MARBLE framework. MARBLE (Modernization Approach for Recovering Business Processes from LEgacy systems) is a framework that aims to generate business processes by the mean of model-driven reverse engineering from legacy systems. MARBLE is essentially based on KDM. MARBLE has four levels of abstraction, [37]. Level zero is what represents the legacy system. Level one is a set of PSM models, one for each artifact such as database, UI, source code, etc. Level two is one PIM model (KDM model) where all the PSM models of level one are merged according to the KDM metamodel to have an integrated view of all the level one PSM models. Level three describes, in the form of the KDM model, which is a CIM model at this level, the business processes of the legacy system, [38]. The transition from one level to another is resumed in Table 9:

Table 9. Transitions from one level to another

Transformation	Description
Level zero to level one	Static and dynamic analysis is used to generate the first PSM models according to pre-established metamodels depending on the system technologies such as Java metamodel, SQL metamodel, and so on.
Level one to level two	Model-to-model transformations are performed to generate the PIM model (KDM model) according to the KDM metamodel based on the level one PSM models. QVT is the mean to implement those transformations.
Level two to level three	The resulting BPMN model is generated by the mean of QVT which helps to implement the pattern-matching technique that aims to identify which element from the level two KDM should be built and its role in the business process. External actors such as business experts may help in this transformation to complete the lacking business knowledge.

To resume this approach, we have established in Table 10:

Table 10. Summary of, [36] publication

MDRE models	MDRE steps	Tool Support
KDM Models BPMN	Generation of PSM models after static and dynamic analysis Generation of PIM model using QVT Generation of BPMN using QVT in MARBLE and the experts' intervention if needed	MARBLE (available as a plugin in Eclipse) QVT

We note that this approach is a generic semi-automatic one.

- In, [39]

This approach aims to reengineer web applications based on RAD (Rapid Application Development) technology, [40]. To perform reverse engineering, the approach starts with the extraction of models from the legacy code by the mean of Gra2MoL, a DSL (Domain Specific Language) adapted for the extraction of the model from a legacy code according to grammar. After a Text-to-model transformation implemented in Gra2MoL, an AST is generated thus representing the source code with all the event handlers. After that, the AST is transformed to an intermediate model called RADBehaviour by the mean of a model-to-model transformation according to a RADBehaviour metamodel. Then, another model is generated based on RADBehaviour which is called EventConcerns. It aims to describe the legacy code in the form of a control flow graph, [41].

To resume this approach, we have established in Table 11:

Table 11. Summary of, [39] publication

MDRE models	MDRE steps	Tool Support
AST RADBehaviour EventConcerns	Generation of AST Transforming the AST into RADBehaviour Transforming RADBehiaviour into EventConcerns	RubyTL (All the model-to-model transformations are implemented using it)

- In, [42]

The objective of this approach is to migrate automatically the web application into CMS (Content Management System). For the moment, this approach only focuses on open-source CMS like WordPress, Drupal, and Joomla, [43]. To do that, the authors propose three steps for the approach: reverse engineering, restructuring, and forward engineering. Concerning the reverse engineering phase starts with the generation of AST models from the legacy code to have a description of the source code, according to the AST_PHP metamodel. Then, two KDM models are generated based on the previous AST models which are the code model and the inventory model. Finally, by the mean of model-to-model transformations, a CMS model is generated based on KDM model data according to the CMS Common Metamodel.

To resume this approach, we have established in Table 12:

Table 12. Summary of, [42] publication

MDRE models	MDRE steps	Tool Support
AST_PHP metamodel AST KDM CMS Common metamodel	Generation of AST Transforming the AST into PSM Transforming the PSM into PIM	Xtext EBNF (Extended Backus-Naur Form)

2.3 Analysis

As mentioned previously in the last section, we analyze each approach according to five elements. Those elements are the models used, the scope of the approach, the application fields of the approach, the automation level, and the analysis type of the approach whether it is static or dynamic. We have established in Table 13 to resume the whole. By going through the latter, we notice that four out of the ten approaches mentioned use KDM metamodels for different purposes. For instance, Normantas and Vasilecas use KDM for legacy code modeling, GUI modeling, and business processes. Two out of ten approaches, [28], [25], use UML, specifically either its default or personalized profiles. Three out of the ten approaches, [23], [29], [39], define new ad-hoc metamodels. One out of the ten approaches, [31], reuse, in the context of 3D Web systems, domain-specific models. Concerning tools, we distinguish two kinds of approaches, those who use new tools and those who use existing tools. Six out of ten approaches use new tools, [21], [29], [31], [36], [39], [42]. The most common new tool used by these approaches is MoDisco. The latter plays a major role in using KDM under the condition of tool support. For the rest of the mentioned approaches, they use available tools, such as ATL-based ones. Concerning the automation level, six of the ten approaches are automated. The goal behind this is to avoid as much as possible the human intervention to reduce errors. In our literature review, 60% of the approaches are fully automated which means that the model-driven reverse engineering tools are mature enough to support such an automation. Automation has other advantages like improving productivity and reducing costs. After that review, we can determine leads to choose the best approach. Therefore, which one is the best after analyzing a legacy code? It seems to us that there are four elements to take into consideration while choosing an approach. The first one is the application domain. From our analysis, we already distinguished between generic and specific approaches, and we can tell that the specific ones are more efficient in their respective domain. The second one is standards. We have already seen some approaches that use standardized models and others use newly defined models. Therefore, finding experts in standardized models is easier to recruit because of the availability of documentation. From what we have seen, specific approaches often rely on KDM models. The third one is model validation. Most of the approaches don't integrate such a phase in their process. But we think that this is an issue that has to be a threat because model-to-model transformations cannot be fully trusted and need a sort of validation. The fourth one is the tools. The best example in the mentioned approaches is MoDisco where its authors provide tool support so it can be reliable. Therefore, between all approaches that we mentioned, only one fulfills all the criteria which is the MoDisco approach.

Table 13. MDRE approaches analysis

MDRE approach	**Models used**	**Scope**	**Application fields of the approach**	**Automation level**	**Analysis type**
Bruneliere et al, [22]	Extensible Metamodel provided based on OMG KDM	Generic	Very various	Totally	Static and dynamic
Cosentino et al, [44]	Java metamodel Domain Variable Model Business Rule Model	Specific	Extraction of business processes of Java applications	Totally	Static
El Beggar et al, [25]	Cobol Metamodel MMFD UML Domain class diagram	Specific	COBOL applications	Totally	Static
Favre et al, [26]	NEUREUS (for metamodels) UML OCL	Generic	Object-oriented legacy systems	Partially	Static and dynamic
Fleurey et al, [29]	AST ANT UML	Generic	Large banking software	Totally	Static
Lenk et al, [31]	SSIML IM X3D and JavaScript ASTs	Specific	3D Web legacy systems	Partially	Static
Normantas et Vasilecas, [34]	KDM Models	Generic	Corporate software	Partially	Static
Perez Castillo et al, [36]	KDM Models BPMN	Generic	Extraction of business processes	Partially	Static and dynamic
Sanchez Ramon et al, [39]	AST RADBehaviour EventConcerns	Specific	Graphical user interfaces in RAD legacy systems	Totally	Static
Trias et al, [43]	AST_PHP metamodel AST KDM CMS Common metamodel	Specific	Web applications	Totally	Static

3 Conclusion

Due to the non-stop evolution of technology and software development, the number of applications to maintain has evolved too. To deal with this need for maintenance while respecting the cost and using new technologies, MDRE solutions whether they are fully or partially automated must be adopted. We note that adopting MDE techniques in reverse engineering is showing promising results.

This paper is a presentation of a sample of some model-driven reverse engineering approaches adopted by their authors which some of them applied in real-world projects. We analyzed them and helped the audience to choose between them by the mean of our proposed criteria. Even if this field is still young, where the first related work has been published in 2003-2005, numerous works followed which helps MDRE gain in maturity. Therefore, we may witness a huge development in the future.

For our future works, we intend to exploit what we have learned through our literature review to propose our approach. We will try to use as possible as we can standardized models so our final tool can be extended easily.

References:

[1] D. C. Schmidt, "Model-Driven Engineering," Computer-IEEE Computer Society, vol. 39, no. 2, p. 25, 2006.

[2] A. R. d. Silva, "Model-driven engineering: A survey supported by the unified," Computer Languages, Systems & Structures, vol. 43, pp. 139-155, 2015.

[3] A. W. Brown, "Model driven architecture: Principles and practice," Software and Systems Modeling, vol. 3, no. 4, pp. 314-327, 2004.

[4] J. V. Igor Sacevski, "Introduction to Model Driven Architecture (MDA)," Salzburg, 2007.

[5] OMG, "Knowledge Discovery Metamodel," 2016.

[6] OMG, "Abstract Syntax Tree Metamodel," 2011.

[7] C. Pereira, "Eclipse Community Forum," Eclipse, Décembre 2012. [Online]. Available: https://www.eclipse.org/forums/index.php/t/1070982/. [Accessed 21 08 2021].

[8] D. Owens and D. M. Anderson, "A Generic Framework for Automated Quality Assurance of Software Models - Implementation of an Abstract Syntax Tree," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 5, no. 1, pp. 32-38, 2014.

[9] Á. Beszédes, R. Ferenc and T. Gyimóthy, "Columbus: A Reverse Engineering Approach," In STEP, pp. 93-96, 2005.

[10] R. Ferenc, A. Beszedes, M. Tarkiainen and T. Gyimothy, "Columbus -- Reverse Engineering Tool and Schema for C++," International Conference on Software Maintenance, 2002. Proceedings, pp. 172-181, 2002.

[11] F. Heidenreich, J. Johannes, M. Seifert and C. Wende, "Closing the Gap between Modelling and Java," Software Language Engineering, pp. 374-383, 2009.

[12] R. Pawlak, M. Monperrus, N. Petitprez, C. Noguera and L. Seinturier, "Spoon: A Library for Implementing Analyses and Transformations of Java Source Code," Software Practice and Experience, vol. 46, pp. 1155-1179, 2016.

[13] F. Deissenboeck, L. Heinemann, B. Hummel and E. Juergens, "Flexible Architecture Conformance Assessment with ConQAT," ICSE '10: 32nd International Conference on Software Engineering, vol. 2, p. 247–250, 2010.

[14] B. Roy and T. C. N. Graham, "An Iterative Framework for Software Architecture Recovery: An Experience Report," Software Architecture, vol. 5292, pp. 210-224, 2008.

[15] R. Wettel, M. Lanza and R. Robbes, "Software systems as cities: a controlled experiment," 2011 33rd International Conference on Software Engineering (ICSE), pp. 551-560, 2011.

[16] S. Das and S. Shiva, "CoRuM: Collaborative Runtime Monitor Framework for Application Security," 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), pp. 201-206, 2011.

[17] C. J. Permann, D. R. Gaston, D. Andrš, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. M. Peterson, A. E. Slaughter, R. H. Stogner and R. C. Martineau, "MOOSE: Enabling massively parallel multiphysics simulation," SoftwareX, vol. 11, no. 100430, p. 101202, 2020.

[18] P. Swithinbank, M. Chessell, D. T. Gardner, C. Griffin, J. Man, H. Wylie and L. Yusuf, Patterns: Model-Driven Development Using IBM Rational Software Architect, IBM Redbooks, 2005.

[19] 3DS, "MAGICDRAW," Dassault Systèmes, [Online]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/. [Accessed 16 08 2023].

[20] H. Brunelière, J. Cabot, G. Dupé and F. Madiot, "MoDisco: A model driven reverse engineering framework," Information and Software Technology, vol. 56, no. 8, pp. 1012-1032, 2014.

[21] H. BRUNELIERE, "Generic Model-based Approaches for Software Reverse Engineering and Comprehension," Université de Nantes, Nantes, 2018.

[22] H. Bruneliere, J. Cabot, F. Jouault and F. Madiot, "MoDisco: A Generic And Extensible Framework For Model Driven Reverse Engineering," 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010), 2010.

[23] U. Sabir, F. Azam, S. U. Haq, M. W. Anwar, W. H. Butt and A. Amjad, "A Model Driven Reverse Engineering Framework for Generating High Level UML Models from Java Source Code," IEEE Access, vol. 7, pp. 158931-158950, 2019.

[24] L. Crowley, "Managing complexity of Digital Twin models _ Development of NewMODE as a network theory approach to model decomposition," Delft University of Technology, Delft, 2020.

[25] O. E. Beggar, B. Boussetta and T. Gadi, "Comparative Study between Clustering and Model Driven Reverse Engineering Approaches," Lecture Notes on Software Engineering, vol. 1, no. 2, pp. 135-140, 2013.

[26] L. Favre, "A Rigorous Framework for Model-Driven Development," Advanced Topics In Database Research, vol. 5, pp. 1-27, 2006.

[27] J.-M. Favre, "Foundations of model (driven)(reverse) engineering: Models," Proceedings of the International Seminar on Language Engineering for Model-Driven Software Development, Dagstuhl Seminar, 2004.

[28] L. Favre, L. Martinez and C. Pereira, "MDA-Based Reverse Engineering of Object Oriented Code," Enterprise, Business-Process and Information Systems Modeling, vol. 29, pp. 251-263, 2009.

[29] F. Fleurey, E. Breton, B. Baudry, A. Nicolas and J.-M. Jézéquel, "Model-Driven Engineering for Software Migration in a Large Industrial Context," Model Driven Engineering Languages and Systems, vol. 4735, pp. 482-497, 2007.

[30] B. JUNG, M. LENK and A. et VITZTHUM, "Model-driven multi-platform development of 3D applications with round-trip engineering," Software Engineering, 2013.

[31] M. Lenk, A. Vitzthum and B. Jung, "Model-driven iterative development of 3D web-applications using SSIML, X3D and JavaScript," In Proceedings of the 17th International Conference on 3D Web Technology (Web3D '12), p. 161–169, 2012.

[32] B. Jung, M. Lenk and A. Vitzthum, "Structured development of 3D applications: round-trip engineering in interdisciplinary teams," Comput Sci Res Dev, vol. 30, pp. 285-301, 2015.

[33] K. Normantas and O. Vasilecas, "Extracting term units and fact units from existing databases using the Knowledge Discovery Metamodel," Journal Of Information Science, vol. 40, no. 4, pp. 413-425, 2014.

[34] K. Normantas and O. Vasilecas, "Extracting Business Rules from Existing Enterprise Software System,"

Information and Software Technologies, vol. 319, pp. 482-496, 2012.

[35] K. Normantas, S. Sosunovas and O. Vasilecas, "An overview of the knowledge discovery meta-model," In Proceedings of the 13th International Conference on Computer Systems and Technologies (CompSysTech '12), pp. 52-57, 2012.

[36] R. Perez-Castillo, "MARBLE: Modernization approach for recovering business processes from legacy information systems," IEEE International Conference on Software Maintenance, pp. 671-676, 2012.

[37] R. Pérez-Castillo, M. Fernández-Ropero, I. G.-R. d. Guzmán and M. Piattini, "MARBLE. A business process archeology tool," 2011 27th IEEE International Conference on Software Maintenance (ICSM), pp. 578-581, 2011.

[38] R. Pérez-Castillo, I. G.-R. d. Guzmán and M. Piattini, "Business process archeology using MARBLE," Information and Software Technology, vol. 53, no. 10, pp. 1023-1044, 2011.

[39] Ó. S. Ramón, J. Sánchez Cuadrado and J. García Molina, "Model-driven reverse engineering of legacy graphical user interfaces," Automated Software Engineering, vol. 21, pp. 147-186, 2014.

[40] Ó. S. Ramón, J. S. Cuadrado and J. G. Molina, "Model-driven reverse engineering of legacy graphical user interfaces," Automated Software Engineering, vol. 21, pp. 147-186, 2014.

[41] Ó. S. Ramón, J. S. Cuadrado and J. G. Molina, "Reverse Engineering of Event Handlers of RAD-Based Applications," 18th Working Conference on Reverse Engineering, pp. 293-302, 2011.

[42] F. Trias, V. de Castro, M. L. Sanz and E. Marcos, "An ADM-based Method for migrating CMS-based Web applications," International Conference on Software Engineering and Knowledge Engineering, 2016.

[43] F. Trias, V. Castro, M. Lopez-Sanz and E. Marcos, "Migrating Traditional Web Applications to CMS-based Web Applications," Electronic Notes in Theoretical Computer Science, vol. 314, pp. 23-44, 2015.

[44] 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," International Workshop on Rules and Rule Markup Languages for the Semantic Web, pp. 17-31, 2012.

[45] J. H. C. Elliot J.Chikofsky, "Reverse Engineering and Design Recovery: A Taxonomy," IEEE Software, vol. 7, no. 1, pp. 13-17, 1990.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflict of interest to declare.