Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6, 1530-1544
2025
Publisher: Learning Gate
DOI: 10.55214/25768484.v9i6.8185
© 2025 by the authors; licensee Learning Gate

WA2MA: A model-driven approach for reengineering web applications into mobile applications

Nadri Khiati[1*], Djelloul Bouchiha[2], Yahia Atig[3], Sofiane Boukli Hacene[4]
[1,4]EEDIS Laboratory, University of Sidi Bel Abbes, Algeria; nadri.khiati@univ-saida.dz (N.K.).
[2,3]Department of Computer Science, University Centre of Naama, Algeria.

Abstract: With the increasing prevalence of mobile computing, organizations must adapt legacy web applications to pervasive information systems, ensuring seamless accessibility across diverse devices. This paper introduces WA2MA, a model-driven reengineering approach for systematically transforming web applications into mobile applications. The approach follows a structured three-phase process based on Model-Driven Architecture (MDA) principles: Reverse Engineering, where Platform-Independent Models (PIMs) are extracted to represent the structural and behavioral aspects of the existing web application; Transformation, where these models are refined into Platform-Specific Models (PSMs) using a tailored Unified Modeling Language (UML) profile to address mobile platform constraints; and Forward Engineering, where automated model-to-text transformations generate a significant portion of the mobile application's source code. The Acceleo technology is leveraged to facilitate this automation through predefined transformation rules and code generation templates. A case study demonstrates the effectiveness of the WA2MA approach, highlighting a substantial reduction in manual effort and improved consistency in the generated code. This research contributes to the advancement of model-driven engineering (MDE) by providing a structured and scalable methodology for the migration of web applications to mobile platforms.

Keywords: *Code generation, Mobile applications, Model-Driven architecture MDA, Pattern, Pervasive information systems (PIS), Profile, Software reengineering, Unified modeling language (UML), Web applications.*

1. Introduction

The rapid advancements of wireless and mobile technologies has introduced a new era defined by the emergence of pervasive information systems (PIS) [1]. These systems, powered by ubiquitous computing, have revolutionized how we access and interact with information. In the modern interconnected world, information is no longer confined to specific locations or devices; it is accessible from a variety of devices, at any time, and from virtually any place.

This paradigm shift presents a critical challenge: how to effectively adapt and reengineer legacy information systems (ISs), particularly web applications, to meet the requirements of pervasive information systems, with a strong emphasis on mobile applications. This complex transformation process, commonly referred to as re-engineering [2] is the focal point of our research.

In this paper, we propose a systematic approach for reengineering legacy web applications into pervasive information systems, with a focus on mobile platforms, leveraging the principles of Model-Driven Architecture (MDA) a methodology that emphasizes the use of models as the primary artifacts throughout the software development lifecycle. The process is structured around three key phases: Reverse-engineering, Transformation, and Forward-engineering.

- Reverse-engineering: This phase involves the extraction of High-level PIMs from the web application, providing a comprehensive overview of its structure and functionality. These models serve as the foundation for the subsequent phases of transformation and forward-

History: Received: 3 April 2025; Revised: 6 June 2025; Accepted: 9 June 2025; Published: 19 June 2025
* Correspondence: nadri.khiati@univ-saida.dz

engineering, The MDA principles ensures that these models remain independent of the target platform, allowing for more flexible and scalable transformations.

- Transformation: During this phase, we obtain PSMs models by refining PIMs using a proposed UML profile, adapting them to meet the specific needs of mobile platforms. UML, as a standardized visual language for modeling software systems, plays a crucial role here. By extending UML through specialized profiles for mobile applications, we can introduce platform-specific features and constraints.
- Forward-engineering: In the forward-engineering phase, we use Acceleo[1] an MDA based technology to automate the generation of mobile application source code from the UML models. Acceleo supports Model-to-Text generation by utilizing a repository of mapping rules and patterns for technologies such as PHP and Android Java. This automation ensures consistency, reduces manual intervention, and accelerates the development process.

The main contributions of this research include:

1) A comprehensive survey of existing work in web reverse-engineering and mobile application generation.
2) The development of the WA2MA approach, based on MDA principles, for reengineering web applications into mobile applications. This approach integrates reverse-engineering, transformation, and forward-engineering phases, leveraging our custom-developed JAAB tool to automate code generation.
3) A case study demonstrating the practical application of the proposed approach, illustrating its effectiveness in reducing manual effort and improving the efficiency of the reengineering process.

The remainder of this paper is structured as follows: Section 2 reviews related work in the domain. Section 3 introduces the WA2MA framework, detailing the tools and methodologies involved. Section 4 presents a case study that showcases the application of the approach. Finally, Section 5 concludes the paper and outlines directions for future research.

2. Related Work

Over the last decade, several proposals have addressed the challenge of reengineering web applications into various other technologies. Most of these solutions address only parts of the reengineering process (e.g., reverse-engineering). Below, we highlight the contributions made in this domain, particularly those leveraging model-driven techniques.

The field of web reverse-engineering has seen the development of various tools and methodologies, each tailored to address the inherent complexities of web applications. In this context, Bruneliere, et al. [3]. introduced MoDisco, an open-source Eclipse project designed for model-driven reverse-engineering of legacy systems. This project supports processes such as migration, refactoring, and quality assurance, playing a crucial role in transforming legacy systems into modern architectures [3]. Similarly, Reis and da Silva [4] proposed XIS-Reverse, a model-driven reverse-engineering approach that extracts XIS models from legacy relational database schemas and user preferences, facilitating the transformation of legacy information systems [4].

In the broader domain of Model-Driven Engineering (MDE) and Model-Driven Development (MDD), several contributions have been made towards the modeling and automatic generation of mobile applications. For instance, Benouda, et al. [5] presented an MDA approach for mobile application development that incorporates UML-based modeling and automatic code generation, aiming to accelerate the mobile application development process by generating platform-independent models [5]. Lachgar et al. expanded on this by adopting an MDA approach for designing mobile application UIs. They utilized Domain-Specific Languages (DSLs) to create Platform-Independent Models (PIMs)

https://eclipse.dev/acceleo/overview.html[1] in a textual format, applying transformations through Xtext and Xtend 2 to generate platform-specific GUIs [6].

A slightly different approach was taken by Koji, et al. [7] who proposed a UML-based MDD method that separates the design of smartphone applications into structural and UI views. This approach enhances flexibility in application design across multiple mobile operating systems by using a Smartphone Feature Specific Model and a GUI builder independent of the operating system [8]. In the same vein, Ribeiro and Silva introduced XIS-Mobile, a DSL that defines a UML profile generating PIMs, which can be transformed into platform-specific code for multiple mobile platforms, addressing platform fragmentation [9].

Expanding the scope of cross-platform GUI generation, Channonthawat and Limpiyakorn [10] presented a model-driven approach that generates Android GUI code from Windows navigation diagrams. This approach converts input diagrams into XML for UI layouts and Java for UI actions, demonstrating the flexibility of MDA in transforming one platform's models into another's code [10]. Vaupel et al. proposed an MDD approach for modeling mobile applications, supported by an Eclipse-based tool environment that included data, behavior, and UI models, alongside code generators for Android and iOS [11].

Further advancing the MDD approach, Rieger and Kuchen developed the MAML framework, a graphical DSL for modeling business apps. MAML models are translated into md^2 models, which are subsequently transformed into platform-specific code for Android (Java) and iOS (Swift), facilitating seamless model-to-code generation [12]. Similarly, Rehman et al. made significant contributions to cross-platform GUI modeling through MDSE, utilizing UML profiles to generate platform-specific code using Xamarin for both Android and iOS platforms [13].

Few approaches offer a comprehensive solution that covers the entire reengineering process. Bouougada, et al. [14] introduced a model-driven methodology for transforming web applications into linked data, focusing on enhancing the interconnectivity and semantic structure of web content [14]. In contrast, Mehra et al. developed KIRKE, a system that does not directly generate mobile application code but instead creates a wrapper that enables web application functionalities to be accessed as web services within mobile applications [15]. Moreover, Fernandes et al. explore how Progressive web Applications (PWAs) embody the principles of Pervasive Information Systems (PIS), offering flexible, adaptable applications that function seamlessly across various operating systems and devices [16]. Additionally, in the area of cybersecurity, Bisht proposed a solution aimed at reengineering the security of web applications in cloud environments, employing machine learning algorithms to improve security measures [17].

3. Proposed Approach

The Model-Driven Architecture (MDA) paradigm, introduced by the Object Management Group (OMG), provides the theoretical foundation for the proposed approach. MDA emphasizes the use of models as primary artifacts throughout the software development lifecycle, advocating for the separation of system functionality from underlying technological platforms [18].

By abstracting system functionalities into Platform-Independent Models (PIMs), MDA promotes flexibility and facilitates the reuse of core logic across diverse technological environments. These PIMs are subsequently transformed into Platform-Specific Models (PSMs) that incorporate the necessary details for deployment on targeted platforms. As illustrated in Figure 1, the automated model transformation process, governed by predefined rules and mapping specifications, ensures consistency, reduces manual effort, and accelerates the generation of platform-specific code [18].

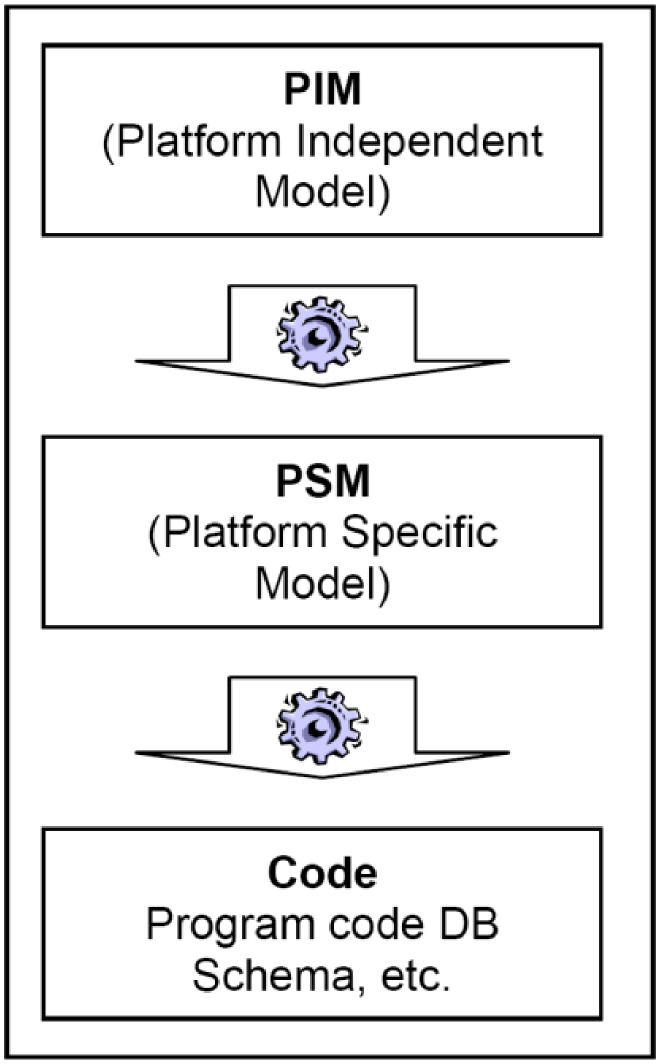
PIM
(Platform Independent Model)
PSM
(Platform Specific Model)
Code
Program code DB Schema, etc.

Figure 1.
MDA Process flow and transformations [18].

Grounded in the principles of Model-Driven Architecture (MDA), the WA2MA (Web Application to Mobile Application) approach provides a systematic solution for reengineering legacy web applications into mobile applications. The approach is particularly suited for business-oriented, data-driven applications. It operates by analyzing the source code of conventional web applications (e.g., HTML, PHP, JSP) and systematically generating the corresponding mobile application source code.

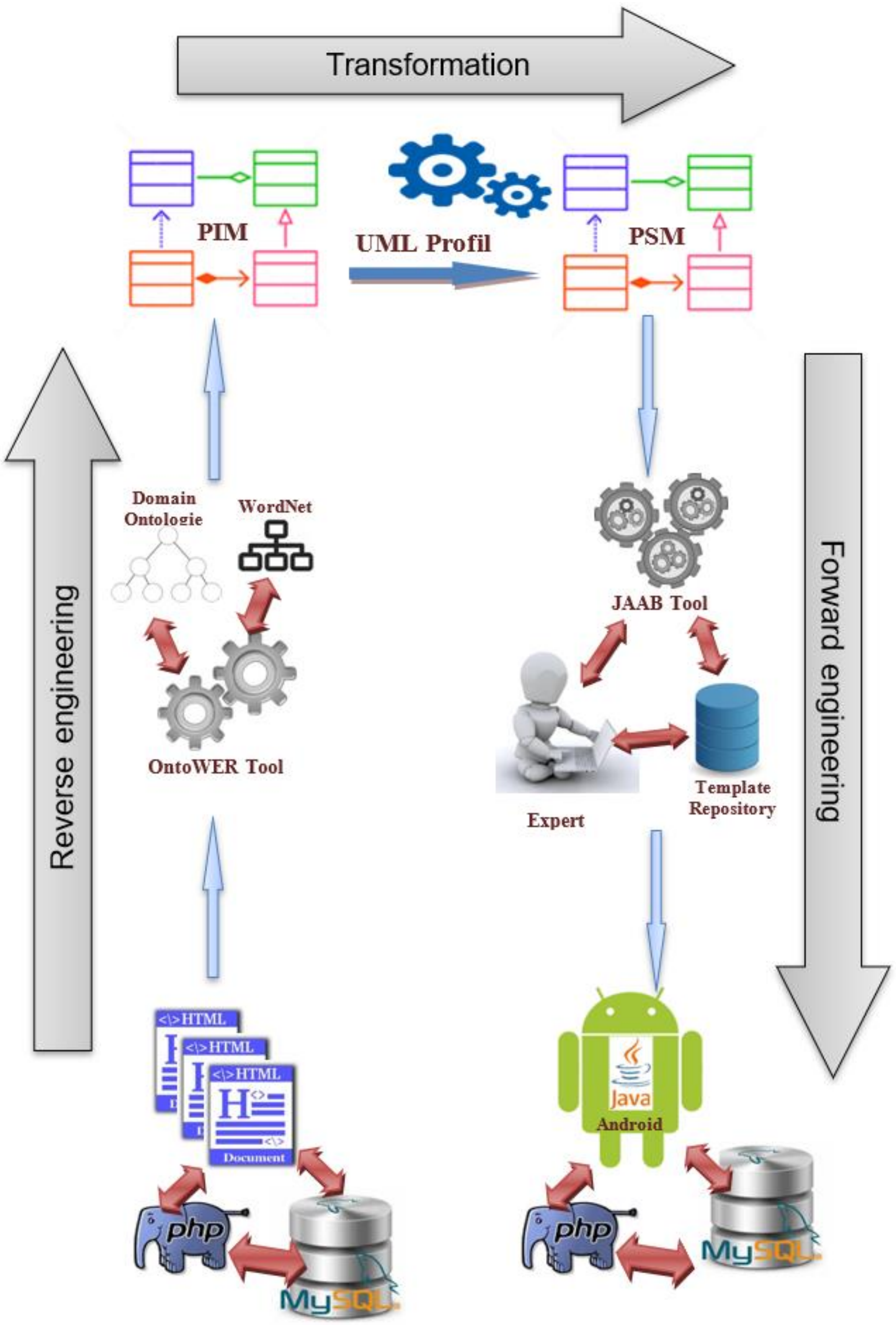

Figure 2.
High-level overview of the WA2MA framework.

As depicted in Figure 2, WA2MA is structured into three core phases: reverse engineering, transformation, and forward engineering. Each of these phases is described in the following subsections.

3.1. Reverse-Engineering

In this phase, Platform-Independent Models (PIMs) are derived from the source code of dynamic web applications (such as HTML, PHP, and JSP). These conceptual models, expressed as UML diagrams, provide an abstract representation of the system's structure and functionality, independent of any specific implementation platform. To facilitate this process, the OntoWeR (Ontology-based Web Reverse-Engineering) tool we proposed in previous work, facilitates this process by utilizing OWL-DL domain ontologies to systematically extract UML class diagrams from the web application's source code [19].

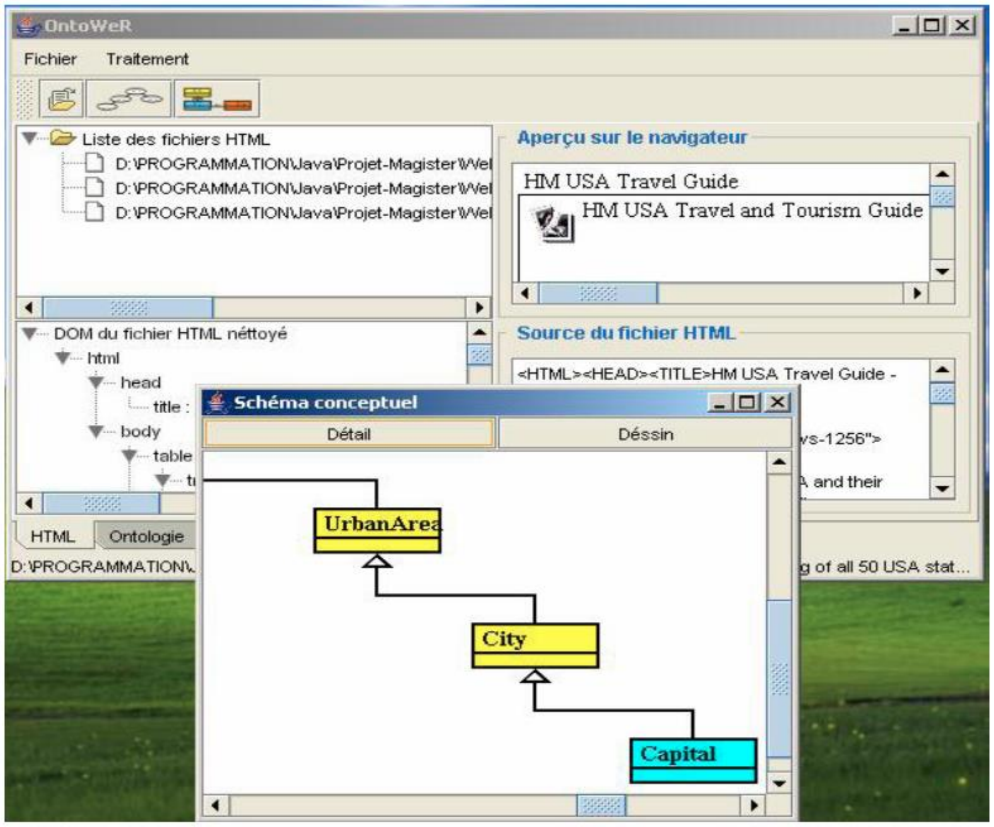

Figure 3.
Snapshot of the OntoWeR tool [19].

OntoWeR employs descriptive logic reasoning to automatically generate diagrams that capture both the static and dynamic aspects of the web application. Its implementation is based on the Protégé-OWL API, which enhances the precision and efficiency of the reverse engineering process.

3.2. Transformation

The transformation phase focuses on refining the Platform-Independent Model (PIM), previously derived during reverse engineering, into a Platform-Specific Model (PSM) tailored for mobile application development. Specifically, this step enhances the UML class diagram by applying a dedicated UML profile targeting the Android platform.

A UML profile serves as a mechanism for customizing and extending the standard UML metamodel to address platform-specific requirements. This specialization is achieved through three key extensibility constructs: stereotypes, tagged values, and constraints. Stereotypes introduce new semantics by extending existing UML metaclasses, tagged values enrich model elements with additional metadata, and constraints define rules to ensure model consistency and correctness according to the target platform's specifications.

In this context, the applied UML profile systematically annotates the conceptual models (PIMs) with Android-specific features, facilitating their transformation into detailed PSMs that guide the subsequent code generation process. Figure 4 illustrates the stereotypes defined within our UML profile, highlighting how they bridge the gap between the abstract PIM and the Android-specific implementation model.

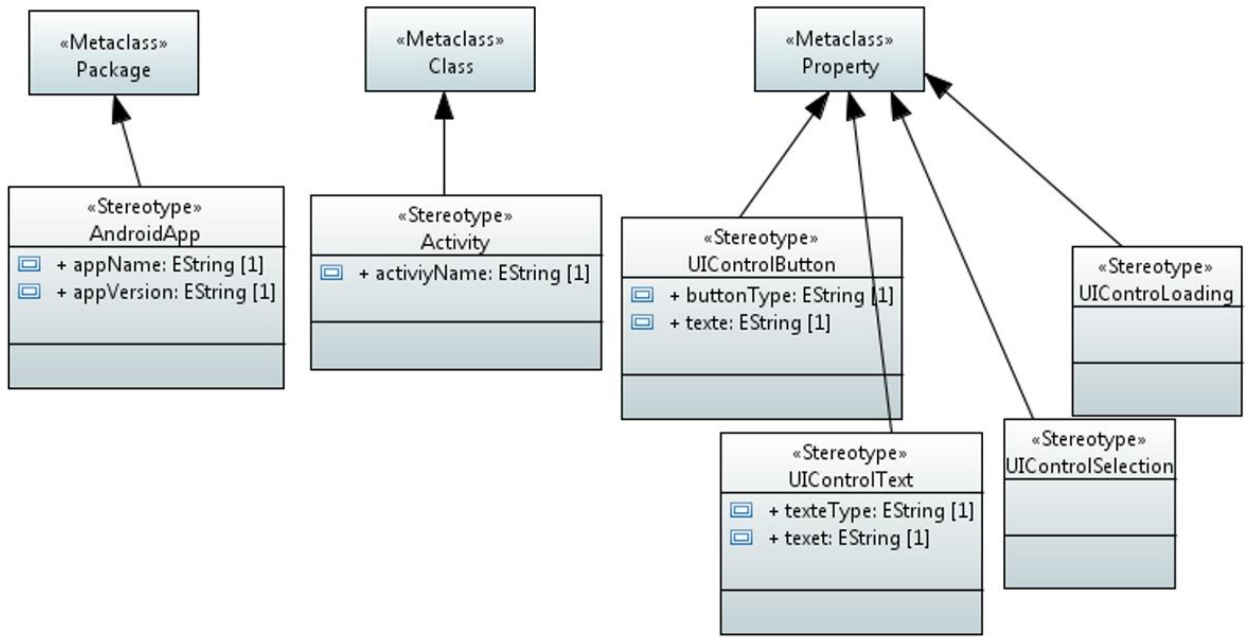
«Metaclass» Package
«Metaclass» Class
«Metaclass» Property
«Stereotype» AndroidApp
+ appName: EString [1]
+ appVersion: EString [1]
«Stereotype» Activity
+ activiyName: EString [1]
«Stereotype» UIControlButton
+ buttonType: EString [1]
+ texte: EString [1]
«Stereotype» UIControLoading
«Stereotype» UIControlText
+ texteType: EString [1]
+ texet: EString [1]
«Stereotype» UIControlSelection

Figure 4.
Stereotypes defined by our UML Profile.

Table 1 presents the key components of the proposed UML profile, detailing the UML metaclasses they extend and the corresponding code artifacts generated from each element.

Table 1.
Mapping between UML profile elements, Extended UML Metaclasses, and the Generated code.

Stereotype	Extended UML element	Generated code
AndroidApp	EPackage	Generates the AndroidManifest.xml and core application setup.
Activity	EClass	Produces an Android Activity class, including lifecycle methods (onCreate, onStart, etc.) and mapped business logic.
UIControlButton	EProperty	Generates a button in the Android XML layout (<Button>), with appropriate action listeners (onClick).
UIControlText	EProperty	Maps to an EditText component in the Android XML layout (<EditText>), with input validation and data binding functionality.
UIControlLoading	EProperty	Generates progress bars or loading indicators in the layout file (<ProgressBar>), with logic for visibility control.
UIControlSelection	EProperty	Maps to dropdown or selection controls (Spinner or RadioButton), with event handling for user input.

It is noteworthy that the transformation phase generally necessitates the involvement of experienced practitioners. This stage represents a pivotal link between the reverse-engineered Platform-Independent Models (PIMs) and the generation of Platform-Specific Models (PSMs) targeting the Android platform. It ensures a seamless transition from the structural abstractions of the web application to the concrete components of the mobile application.

3.3. Forward-Engineering

This phase is facilitated by JAAB (Java-Android Application Builder), a code generation tool specifically that we developed for our approach using the Acceleo. Acceleo supports model-to-text transformations by utilizing template-based generation and offers comprehensive authoring capabilities for creating custom code generators. It enables the automated production of various source code artifacts from models conforming to the Eclipse Modeling Framework[2] (EMF).

Within WA2MA approach, JAAB is instrumental in automating the forward engineering phase. It processes the enriched UML class diagrams, representing the refined Platform-Specific Model (PSM) tailored for Android applications, and applies a set of predefined transformation templates. These templates systematically generate structured Java source code encompassing the user interface components, business logic, data management, and database interaction layers of the target mobile application.

One of JAAB's key advantages lies in its adaptability. Domain experts can customize the templates and generation rules to address specific application requirements, ensuring that the generated code aligns with the functional and non-functional demands of the project. This flexibility makes JAAB a robust and extensible solution for generating high-quality, maintainable Android application code within an MDA-based reengineering process.

Templates play a pivotal role within the JAAB tool, forming the foundation for automating the generation of Android application code. The design and implementation of these templates typically necessitate the involvement of professionals with expertise in both web and mobile application development. Serving as structured blueprints, these templates define the transformation rules required to produce Android code that seamlessly integrates with underlying databases and web services.

Given that the target applications are predominantly data-driven, the JAAB tool automates the generation of Android Java code by applying predefined templates that encapsulate core data management operations, specifically following the Create, Read, Update, and Delete (CRUD) paradigm. For each business entity defined within the Platform-Specific Model (PSM), JAAB systematically produces the necessary Java classes and XML interface layouts. These components cover various application screens and functionalities, including user interface elements, data processing, and interactions with remote databases or web services. The automated process ensures the consistent and efficient implementation of the mobile application's functional requirements, while reducing manual coding efforts and potential errors. Among the generated classes are:

- Entity_Activity.java: for displaying a list of instances related to a specific business entity.
- New_Entity_Activity.java: Provides the interface and functionality to add new instances of the entity.
- Edit_Entity_Activity.java: Enables to view and update the details of an individual entity instance.

In addition to the Java classes, JAAB generates user interface layouts described using XML files, such as Entity.xml, New_Entity.xml, and Edit_Entity.xml. These XML files define the visual structure and user interface components for the corresponding activities, ensuring a consistent and intuitive user experience.

On the server side, the tool also produces a series of PHP scripts to manage the backend operations associated with each CRUD process. These include:

- Get_all_Entitys.php for retrieving all instances of an entity,
- Create_Entity.php for inserting new records,
- Get_Entity_details.php for fetching specific entity details,
- Update_Entity.php for modifying existing records,
- Delete_Entity.php for removing entity instances from the database.

To support these server-side scripts, JAAB generates a db_connect.php file, which establishes the database connection required by the PHP scripts. When necessary, the tool is also capable of producing SQL scripts to create the database schema from scratch, further streamlining the deployment process.

https://projects.eclipse.org/projects/modeling.emf.emf[2]

Beyond application code and database scripts, JAAB automates the generation of additional essential files that are integral to an Android application's configuration and operation. These include:

- AndroidManifest.xml, which declares essential application metadata, permissions, and components.
- JSONParser.java, a utility class designed to handle JSON data parsing, facilitating communication between the mobile client and the backend services.

Listing 1 presents the *generateManifestFile.mtl* module, which encapsulates the transformation logic responsible for producing the AndroidManifest.xml file as part of the overall code generation workflow.

```
[comment encoding = UTF-8 /]
[modulegenerateManifestFile('http://www.eclipse.org/uml2/5.0.0/UML')]
[templatepublicgenerateManifestFile(aPackage : Package)]
[file ('AndroidManifest.xml',false,'UTF-8')]
<?xml version="1.0" encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="[aPackage.name/]"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk
android:minSdkVersion="8" />
<uses-permission android:name="android.permission.INTERNET"/>
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
                   [for(aClass : Class|aPackage.packagedElement->filter(Class))]
                            [for(s: Stereotype|aClass.getAppliedStereotypes()) ? (s.name='Activity')]
         <activity
         android:name=".[aClass.name.toUpperFirst()/]Activity"
         android:label="@string/app_name" >
         </activity>
         <activity
         android:name=“. New[aClass.name.toUpperFirst()/]Activity"
         android:label="@string/app_name" >
         </activity>
         <activity
         android:name=".All[aClass.name.toUpperFirst()/]sActivity"
         android:label="@string/app_name" >
         </activity>
         <activity
         android:name=".Edit[aClass.name.toUpperFirst()/]Activity"
         android:label="@string/app_name" >
         </activity>
                            [/for]
                   [/for]
</application>
</manifest>
[/file]
[/template]
```

Listing 1. Generate Manifest File.mtl module.

This template-driven approach enhances the reengineering process by enabling the rapid and consistent generation of customized code, in line with the Platform-Specific Model (PSM). It ensures efficiency and uniformity across various mobile applications, while facilitating the seamless integration of databases and web services. As a result, the conceptual models derived from the Platform-Independent Model (PIM) are effectively transformed into fully operational Java Android code.

4. Evaluation and Discussion

In this section, we conduct an evaluation of the proposed approach's effectiveness by applying it to a representative case study.

4.1. Case Study

This subsection introduces a comprehensive case study to demonstrate the practical implementation of the proposed approach. The case study highlights the reengineering process, emphasizing the transformation of a legacy web application into a mobile application within the context of Model-Driven Architecture (MDA).

For this purpose, we assume an Employee Management System (EMS), a business-oriented web application that constitutes a core module of a larger Enterprise Resource Planning (ERP) suite. The EMS facilitates the management of human resources, including employee records and scheduling of meetings. Within the ERP environment, EMS operates in conjunction with other integrated modules, such as finance, supply chain management, and customer relationship management (CRM) to deliver a cohesive and comprehensive organizational management solution.

4.1.1. Reverse-Engineering

The initial phase of the reengineering process is reverse engineering, during which the existing web application is systematically analyzed using the OntoWeR tool. This phase focuses on extracting the fundamental components and their interrelationships from the application's source code. The extracted information is then used to construct a Platform-Independent Model (PIM), represented as a conceptual UML model that abstracts the system's structure and functionality, independent of any specific implementation technology.

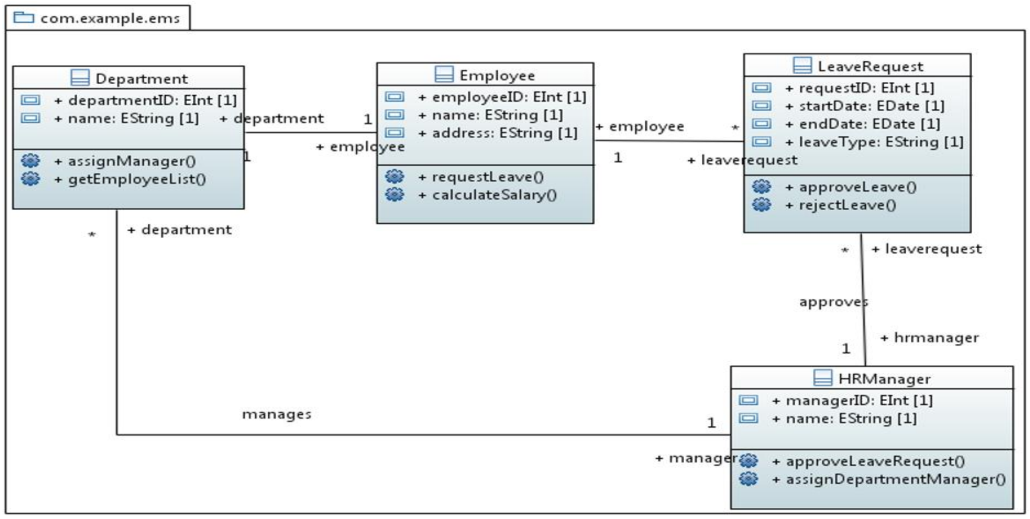

Figure 5.
PIM Class Diagram Extracted by OntoWeR.

As a result of the reverse-engineering process, a UML class diagram was generated, as illustrated in Figure 5. This model captures the core entities of the Employee Management System (EMS), including primary classes such as *Employee*, *Department*, and *HRManager*, along with their associations. The class diagram constitutes a crucial Platform-Independent Model (PIM), serving as the foundational blueprint for the subsequent transformation phase.

4.1.2. Transformation

The second phase of the reengineering process is the model transformation stage, in which the Platform-Independent Model (PIM), derived during reverse engineering, is refined with platform-specific annotations to create a Platform-Specific Model (PSM). At this stage, the UML profile specifically designed for Android platform, previously described, is applied to the conceptual model.

In this transformation phase, each class representing a core business entity is systematically annotated with relevant stereotypes that align with Android components. For example, the *Employee* class is marked with the stereotype «Activity», indicating its role as an Android activity responsible for managing user interactions such as displaying employee details, adding new records, or updating existing ones.

Class attributes are also extended with stereotypes that define their mapping to Android user interface components. For instance, attributes of the *Employee* class requiring user input are annotated with the «UIControlText» stereotype, which corresponds to Android's *EditText* element. This ensures that the appropriate UI components are generated automatically in the subsequent forward-engineering phase.

Furthermore, the expressiveness of the UML model is enhanced through the use of tagged values, providing additional metadata and constraints for each stereotype. These tagged values specify field behaviors such as validation rules, required inputs, or hints for data formatting, thereby enriching the semantic detail of the PSM.

This annotation process is applied consistently across all relevant entities in the model, including classes such as *Department*. By applying the UML profile systematically, the transformed model preserves the original business logic while adapting it for implementation on the Android platform.

The result of this transformation phase is an enriched UML class diagram, illustrated in Figure 6 that serves as the definitive PSM, guiding the automated generation of the Android application code in the next phase.

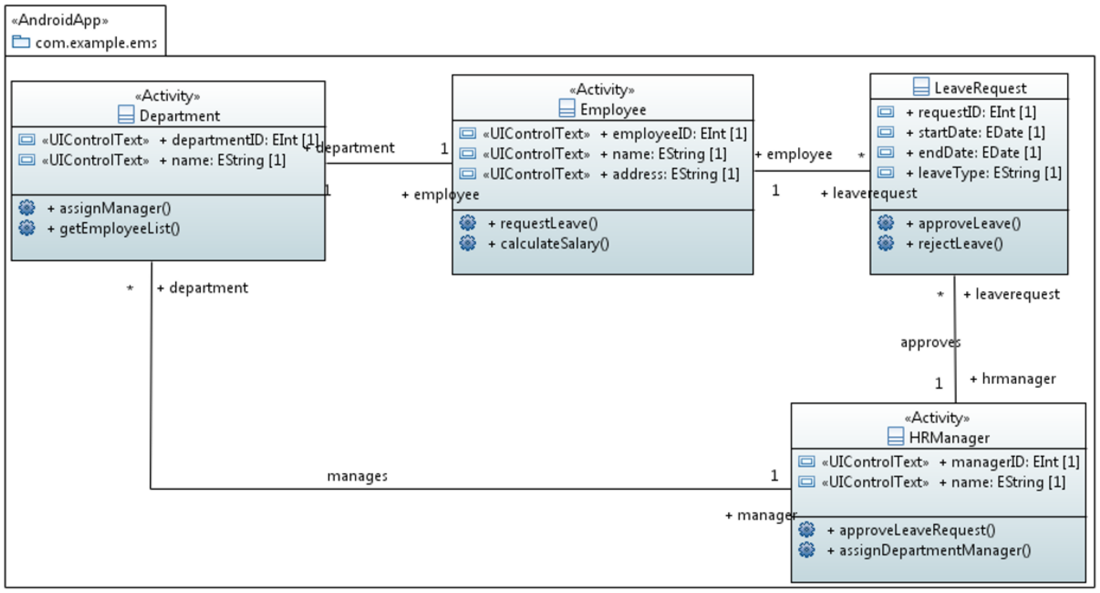

Figure 6.
PSM Class Diagram Annotated with Android-Specific UML profile.

4.1.3. Forward-Engineering

In the final stage of the reengineering process, the enriched Platform-Specific Model (PSM), represented by the annotated class diagram, is used as input for the JAAB (Java-Android Application Builder) tool. Developed using the Acceleo framework, JAAB automates the generation of a substantial portion of the Android application's source code, in addition to producing PHP scripts that enable seamless communication between the mobile application and the backend database. When required, JAAB can also generate SQL scripts for initializing or recreating the database schema.

Throughout the code generation phase, the tool's Main Module orchestrates the execution of specialized sub-modules. Each sub-module is responsible for generating distinct artifacts of the Android application, including the AndroidManifest.xml file, the core Java classes, XML files defining the user interface, and the PHP scripts managing database interactions. These backend scripts implement the CRUD (Create, Read, Update, Delete) operations for the business entities defined within the model.

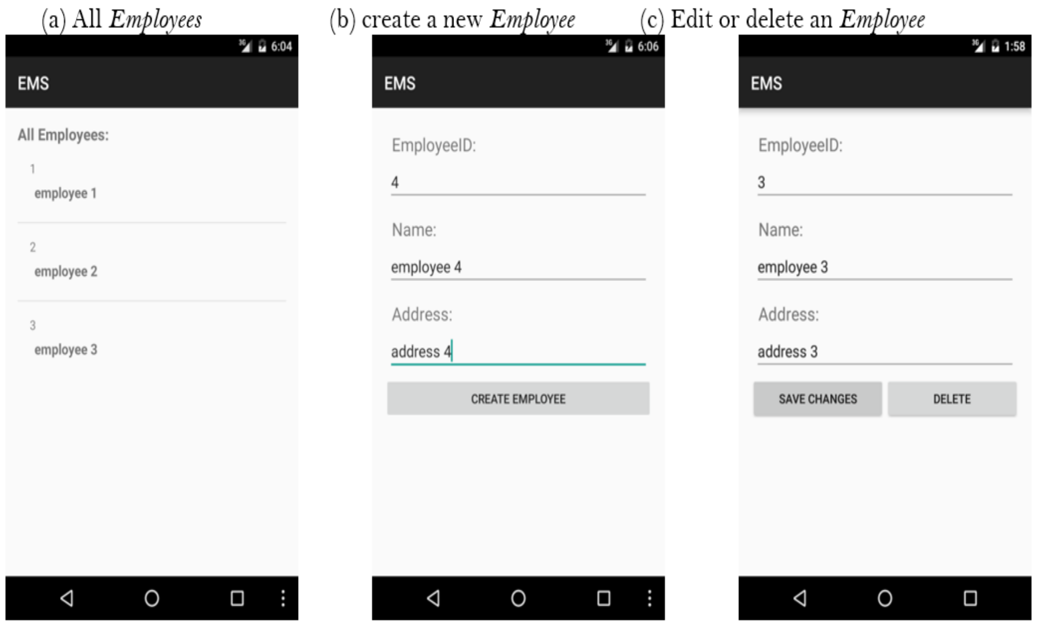

Figure 7.
Screenshots from the generated mobile application.

Figure 7 showcases screenshots from the user interface of the generated Android application for the Employee Management System (EMS). The images illustrate key screens that support typical CRUD functionalities for the Employee entity, including the list view, the activity for adding a new Employee, and the activity for editing or deleting existing records.

4.2. Discussion

The effectiveness of the proposed Model-Driven Architecture (MDA)-based approach was assessed by analyzing the artifacts automatically generated from the Platform-Specific Model (PSM) using the JAAB tool.

As summarized in Table 2, for each business entity, the tool produces four Java classes that define Android activities responsible for data presentation and user interaction. In addition, five XML files specify the layout and design of the corresponding user interfaces. The tool also generates five PHP scripts per entity, implementing CRUD operations and facilitating communication between the mobile application and the back-end database.

Furthermore, six common files are produced to support the entire application, including the AndroidManifest.xml and JSON parser classes, which provide essential configuration and integration functionality. This systematic generation process highlights the effectiveness of the MDA-based approach in transforming platform-specific models into a fully functional mobile application.

Table 2.
Overview of the generated files.

Category	File Type	Number of Files	Purpose
Android Activities	Java (.java)	4 per entity	Defines business logic and activities (e.g., Create, Read, Update, Delete) associated with each business entity.
GUI Layouts	XML (.xml)	5 per entity	Specifies the user interface components for activities (e.g., form layouts for customer creation, product listing).
Database Access	PHP (.php)	5 per entity	Manages CRUD operations between the Android application and the backend database (e.g., retrieving, updating, and deleting data).
Common Application Files	Mixed (XML, Java, SQL)	6	Includes global application files (e.g., AndroidManifest.xml, JSONParser.java, db_connect.sql) and other supporting files used across the app.

The generated code exhibits several strengths, alongside areas that offer potential for further refinement. These aspects are summarized as follows:

4.2.1. Quantitative Assessment of Time and Effort Reduction

A primary advantage of the JAAB tool lies in its capacity to automate the generation of repetitive code, particularly for CRUD operations. Internal evaluations indicate that manually implementing such functionalities for a typical business entity requires approximately 8 to 10 hours, depending on the complexity of the data model and user interface specifications. In contrast, the JAAB tool produces equivalent code in less than one minute. This demonstrates a substantial reduction in development time, exceeding 90%, allowing developers to allocate greater focus to custom logic and application-specific enhancements. Furthermore, the time efficiency gained scales proportionally when applied to larger applications with multiple business entities.

4.2.2. Code Quality and Maintainability

Although the generated code successfully implements CRUD operations and core functionalities, an evaluation of its quality and maintainability was conducted. The artifacts produced by the JAAB tool exhibit a consistent and standardized structure, adhering to established Android and PHP development practices. This uniformity facilitates readability, comprehension, and ease of adaptation.

Overall, the maintainability of the generated code is considered high, as it supports straightforward extension and modification. Nevertheless, future enhancements to the JAAB tool could introduce greater modularity by decomposing large code files into smaller, reusable components, thereby improving code organization and maintainability.

Additionally, the proposed approach would benefit from an integrated testing phase. As noted by [20], the success of mobile applications is shaped by factors such as business relevance, customer value, and user-perceived quality. Therefore, incorporating usability testing is essential to evaluate user satisfaction and ensure the acceptance of the generated applications.

4.2.3. Enhancing the Transformation Phase and Templates Library

Although the automated code generation facilitated by the JAAB tool effectively accelerates the reengineering process in alignment with Model-Driven Architecture (MDA) principles, opportunities for refinement remain, particularly within the model transformation phase. The current UML profile could be extended with additional platform-specific stereotypes to better capture mobile-centric concerns, such as lifecycle management, event handling, and user interaction flows. For example, introducing elements like «Service» and «BroadcastReceiver» within the Platform-Specific Model (PSM) would enable the automated generation of background services and notification mechanisms, which are integral to Android applications.

Moreover, the template library, central to the model-to-text transformation process, could be expanded to support more advanced Android features, including seamless integration with third-party APIs. As these capabilities have become standard in modern mobile development, extending the templates would further minimize manual intervention following code generation. By continually enriching the PSM and its associated generation templates, the JAAB tool could support a broader range of use cases and domains, extending its utility beyond basic data-driven applications.

5. Conclusion

Provide Software reengineering has demonstrated its effectiveness as a cost-efficient strategy for adapting legacy applications to new platforms and extending their functionality. This paper presented WA2MA, a model-driven reengineering approach designed to enable the seamless migration of web applications to mobile platforms. The proposed approach is structured into three principal phases:

- Reverse engineering: Extracting Platform-Independent Models (PIMs) that capture the structural aspects of existing web applications.
- Transformation: Refining the PIM into a Platform-Specific Model (PSM) through the application of a dedicated UML profile tailored to address Android platform requirements.
- Forward engineering: Automatically generating a substantial portion of the mobile application's source code from the PSM, including the implementation of CRUD operations and core application logic.

The core contribution of WA2MA lies in its ability to automate the generation of a mobile application's foundational structure, including comprehensive CRUD functionality, directly from an annotated UML class diagram derived during the reverse engineering process. This automation reduces development time, minimizes manual coding errors, and provides a solid basis for further customization and the integration of platform-specific features.

Future work will focus on several key enhancements. A primary objective is the enrichment of the UML profile used during the transformation phase. This will involve introducing additional stereotypes and tagged values to better capture mobile-specific concerns, such as lifecycle management, background services, and event handling. Another area of improvement is the expansion of the model-to-text (M2T) code generation templates to support the behavioral aspects of mobile applications, including user interactions and security mechanisms. This will increase the approach's versatility and applicability across a wider range of application types. Finally, the framework will be extended to support multiple mobile platforms, such as iOS, by developing platform-specific UML profiles and corresponding PSM transformation rules.

Transparency:

The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Copyright:

References

[1] M. K. Pinheiro, C. Souveyet, P. Roose, and L. A. Steffenel, *The evolution of pervasive information systems.* Cham, Switzerland: Springer, 2023.

[2] D. Bouchiha, *Reengineering legacy systems towards new technologies. In Encyclopedia of Information Science and Technology*. Hershey, PA, USA: IGI Global, 2021.

[3] H. Bruneliere, 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.

[4] A. Reis and A. R. da Silva, "XIS-Reverse: A model-driven reverse engineering approach for legacy information systems," in *International Conference on Model-Driven Engineering and Software Development*, 2017, vol. 2: SCITEPRESS, pp. 196-207.

[5] H. Benouda, M. Azizi, R. Esbai, and M. Moussaoui, "MDA approach to automate code generation for mobile applications," presented at the Mobile and Wireless Technologies 2016, 2016.

[6] M. Lachgar and A. Abdali, "Modeling and generating native code for cross-platform mobile applications using DSL," *Intelligent Automation & Soft Computing*, vol. 23, no. 3, pp. 445-458, 2017.

[7] T. Koji, M. Sato, Y. Nakamura, H. Tanaka, and K. Suzuki, "A novel approach to predictive modeling in time series analysis," *Journal of Data Science and Analytics*, vol. 12, no. 3, pp. 215–230, 2023.

[8] K. Matsui and S. Matsuura, "MDD for smartphone application with smartphone feature specific model and GUI builder," presented at the the Ninth International Conference on Software Engineering Advances (ICSEA 2014), 2014.

[9] A. Ribeiro and A. R. da Silva, "Xis-mobile: A dsl for mobile applications," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, 2014, pp. 1316-1323.

[10] T. Channonthawat and Y. Limpiyakorn, "Model driven development of Android application prototypes from Windows navigation diagrams," presented at the 2016 International Conference on Software Networking (ICSN), 2016.

[11] S. Vaupel, G. Taentzer, J. P. Harries, R. Stroh, R. Gerlach, and M. Guckert, "Model-driven development of mobile applications allowing role-driven variants," presented at the Model-Driven Engineering Languages and Systems: 17th International Conference, MODELS 2014, Valencia, Spain, September 28–October 3, 2014. Proceedings 17, 2014.

[12] C. Rieger and H. Kuchen, "A process-oriented modeling approach for graphical development of mobile business apps," *Computer Languages, Systems & Structures*, vol. 53, pp. 43-58, 2018.

[13] S. Rehman, R. M. K. Ullah, S. Tanvir, and F. Azam, "Development of user interface for multi-platform applications using the model driven software engineering techniques," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018: IEEE, pp. 1152-1158.

[14] B. Bouougada, D. Bouchiha, Y. Ouhammou, and M. Malki, "Re-engineering web application towards linked data: A model-based approach," *International Arab Journal of e-Technology*, vol. 5, no. 2, pp. 58-70, 2018.

[15] R. Mehra, V. Naik, R. Purandare, and K. Malik, "KIRKE: Re-engineering of web applications to mobile apps," in *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2016, pp. 135-142.

[16] G. Fernandes, F. Portela, and M. F. Santos, "PWA and pervasive information system–a new era," presented at the Trends and Innovations in Information Systems and Technologies: Volume 3 8, Hiroshima, Japan, 2020.

[17] P. Bisht, "Web applications security re-engineering in cloud with machine learning," *International Journal of Scientific Research in Multidisciplinary Studies*, vol. 9, no. 11, pp. 79–90, 2023. https://doi.org/10.13140/RG.2.2.13884.21126

[18] A. McNeile, *MDA: The vision with the hole.* London, W8 4AN, UK: Metamaxim Ltd. 48 Brunswick Gardens, 2003.

[19] S. M. Benslimane, M. Malki, D. Bouchiha, and D. Benslimane, "Ontower: An ontology based web application reverse-engineering approach," *International Review on Computers and Software*, vol. 1, pp. 52-58, 2006.

[20] P. Weichbroth, "Usability testing of mobile applications: A methodological framework," *Applied Sciences*, vol. 14, no. 5, p. 1792, 2024. https://doi.org/10.3390/app14051792