informatics

MDPI

Article

E-MDAV: A Framework for Developing Data-Intensive Web Applications

Paolo Bocciarelli *, Andrea D'Ambrogio, Tommaso Panetti and Andrea Giglio

Department of Enterprise Engineering, University of Rome Tor Vergata, Via del Politecnico 1, 00133 Rome, Italy; dambro@uniroma2.it (A.D.); tommaso.panetti@tiscali.it (T.P.); giglio.andrea@gmail.com (A.G.)
* Correspondence: paolo.bocciarelli@uniroma2.it

Abstract: The ever-increasing adoption of innovative technologies, such as big data and cloud computing, provides significant opportunities for organizations operating in the IT domain, but also introduces considerable challenges. Such innovations call for development processes that better align with stakeholders needs and expectations. In this respect, this paper introduces a development framework based on the OMG's Model Driven Architecture (MDA) that aims to support the development lifecycle of *data-intensive web applications*. The proposed framework, named *E-MDAV (Extended MDA-VIEW)*, defines a methodology that exploits a chain of model transformations to effectively cope with both forward- and reverse-engineering aspects. In addition, E-MDAV includes the specification of a reference architecture for driving the implementation of a tool that supports the various professional roles involved in the development and maintenance of data-intensive web applications. In order to evaluate the effectiveness of the proposed E-MDAV framework, a tool prototype has been developed. E-MDAV has then been applied to two different application scenarios and the obtained results have been compared with historical data related to the implementation of similar development projects, in order to measure and discuss the benefits of the proposed approach.

Keywords: business information systems; model-driven engineering; low-code development; data-intensive web applications

check for updates

Citation: Bocciarelli, P.; D'Ambrogio, A.; Panetti, T.; Giglio, A. E-MDAV: A Framework for Developing Data-Intensive Web Applications. *Informatics* **2022**, *9*, 12. https://doi.org/10.3390/informatics9010012

Academic Editor: Antony Bryant

Received: 13 January 2022
Accepted: 7 February 2022
Published: 12 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Over the past few years the IT sector has witnessed the appearance of technologies denoted as *disruptive* [1], such as *cloud computing* and the *Internet of Things (IoT)*. The cloud computing paradigm enables the use of a computing infrastructure in an on-demand and pay-per-use fashion, by composing IT resources provided as *services* that can be accessed on the Internet [2]. IoT platforms extend the ability to exchange data over the Internet to a large set of objects of various types.

This paper deals with the impact in terms of the application development lifecycle, by addressing the case of *data-intensive applications*, which are particularly relevant due to the availability of an ever-increasing amount of data [3]. For such applications, implementing CRUD (create, read, update, and delete) operations contribute to a large part of the required development effort (up to 80% according to [4]). Thus, automating the code generation for CRUD operations significantly reduces application development effort and time.

In addition to the data-intensive nature, this paper addresses the case of *web-based applications* deployed onto cloud-based platforms, which are setup through workflows (e.g., platform configuration, container instantiation and deployment, etc.) that largely benefit from the use of automated approaches [5]. Moreover, it is worth noting that the cloud-computing paradigm may refer to either *server* or *serverless* computing models. In this respect, this work takes into consideration the server-based cloud computing model.

Finally, data-intensive web applications are also characterized by frequently changing user requirements, which require the adoption of rapid (i.e., largely automated) development processes to properly face the regular (up to weekly or daily) delivery of application releases, especially when cloud resources are involved [6]. This has led to the introduction of the so-called *DevOps* approach, which reduces the gap between development and operation phases by supporting a continuous delivery paradigm [7,8].

In this context, the adoption of the aforementioned technologies bring significant opportunities for dealing with the inherent characteristics of the addressed data-intensive web applications, but also introduce the following challenges in terms of cost-effectiveness:

- The application development process shall be appropriately tailored to effectively take advantage of using a service-oriented approach in the cloud;
- The development and configuration of a complex web application, which includes various components (middleware, databases, application services, etc.), and its deployment on a cloud-based execution platform, require heterogeneous skills;
- Developing the executable code to create (as well as read from and write to) large databases consisting of a wide set of tables and fields, is a repetitive and effort-consuming activity that can easily be error-prone if carried out manually;
- The testing environment usually requires a huge hardware resources stack for replicating the operation environment due to the large amount of data stored in the databases;
- Traditional software development approaches might fail when smooth cooperation between development and IT operations are required to achieve frequent releases and reduce the time to market.

In order to properly address such challenges, this paper introduces *E-MDAV (Extended MDA-VIEW)*, a development framework that yields a significant degree of automation in the lifecycle of data-intensive web applications. The E-MDAV framework extends a previous contribution (see Section 2 for further details) and is inspired by DevOps approaches. The increased degree of automation is obtained using principles, methods, tools and standards introduced by *Model Driven Architecture (MDA)* [9], the framework promoted by the Object Management Group (OMG) to address model-driven software development.

This paper addresses methodological-, design- and validation-related perspectives.

From the methodological perspective, E-MDAV introduces a methodology that aims to effectively support the involved stakeholders in addressing the aforementioned challenges, by providing automation and guidance throughout the application development process.

Specifically, the methodology, which is discussed in detail in Section 5, aims at pursuing the following goals, as further detailed in Section 5.2:

- Ease the application development by introducing appropriate model-driven transformations for generating implementation stubs, test artifacts and configuration files;
- Exploits *model-to-text* transformations for supporting the development of optimized CRUD operations;
- Provide actual guidance for enacting a continuous delivery approach, that is, for handling the development, integration and modification of the various application's releases, as prescribed by the DevOps methodology.

Specifically, the proposed methodology addresses both forward- and reverse-engineering activities, as explained below:

- When applied to *forward-engineering* activities, the model transformation chain at the basis of E-MDAV enables developers to generate and deploy the required executable application from the specification of the conceptual *data model*, which is specified in UML. Specifically, the automated code generation addresses the development of both the application and the database schema, as well as the deployment over a cloud platform, thus avoiding the execution of effort-consuming and error-prone manual activities;
- When applied to *reverse-engineering* activities, the proposed methodology allows for the generation of the conceptual data model from an existing physical database. Such a

feature helps with achieving consistency between the physical database and its related conceptual data model, thus providing effective support to maintenance activities.

From the design-related perspective, this work introduces the reference architecture of an MDA-compliant tool, which supports and partially automates the execution of the activities under the responsibility of various stakeholders. Since *automation* is one of the pillars of *continuous integration* (CI) and delivery [7], the adoption of MDA's standards allows for the specification and implementation of relevant model transformations that automates several activities of the proposed methodology.

Finally, under a validation perspective, the paper discusses the application of a prototypal E-MDAV implementation to a real-world example.

In this respect, the adoption of an MDA-based approach gives *models* a primary role in E-MDAV. Models, which captures the stakeholders' requirements, actually drive the implementation of web applications as they are taken as input by a transformation chain in charge of generating the application code and the relevant configuration artifacts. On the other hand, appropriate reverse transformations allows stakeholders to keep models constantly aligned to the database schema of existing applications.

The nature of the proposed methodology makes E-MDAV a valuable tool for supporting the execution of rapid and frequent application releases. Moreover, it also allows developers to easily face any required changes on the database schema, by supporting the seamless regeneration and redeployment of the web application.

The rest of this paper is organized as follows: Section 2 reviews relevant literature and positions the proposed contribution with respect to existing approaches. Section 3 provides the required background concepts. Section 4 discusses how the proposed framework addresses relevant challenges in the development of data-intensive web applications. Section 5 illustrates the methodology at the basis of E-MDAV, while Section 6 describes the reference architecture of a supporting tool. Section 7 discusses the evaluation of the prototype E-MDAV implementation, and finally, Section 8 gives concluding remarks, as well as directions for future work.

2. Related Work

The automated generation of applications is a research field largely investigated in the literature. *Scaffolding*, whose popularity has been pushed by the *Ruby on Rails* framework [10], is a widely used approach for automated code generation. It makes use of a scaffolding engine to generate application code from a set of predefined templates and a specification provided by developers.

Increasing the level of automation throughout the application lifecycle is addressed by model-driven development approaches, which enhance the advantages of a traditional model-based approach (e.g., improved quality, enhanced communication and stakeholder engagement, increased productivity, enhanced knowledge transfer, and reduced risks) and also introduces appropriate model transformation for generating refined models and executable code from abstract models [11]. The adoption of such approaches leads to a different use of models that are no longer used only for specifying the system to be developed, but also to actually play a productive role in the development process.

The use of such approaches enables a radical shift in terms of modelling activities, from a strictly contemplative use of models to a more productive model use.

Specifically, this paper exploits MDA (Model Driven Architecture), the model-driven development effort promoted by the Object Management Group (OMG) [9]. MDA provides principles and standards to develop a software application by specifying and executing a set of model transformations, which take as input models at higher level of abstraction and yield as output models at lower level of abstraction, until an executable model of the application is obtained. The E-MDAV framework is inspired by MDA and introduces model transformation approaches for the automated development of data-intensive web applications.

In [12], model-driven techniques are assessed in terms of the benefits they are intended to bring when applied to a database re-engineering process. The proposed method, which is applied to a relational data migration scenario, introduces appropriate *model-to-model* and *model-to-text* transformations to generate the database implementation starting from its relevant data model. Such an approach follows a *dumb-code* generation process, according to which an application source code is first generated and then manually compiled to produce the final executable application.

Differently from such an approach, E-MDAV allows users to directly generate the web-based application using a so-called *smart-code* generation process. In a smart-code generation process, the executable application is generated at run-time in a single step, without the need of manually compiling the source code, as shown in Figure 1. The benefits of smart-code generation are reduced system configuration effort, increased efficiency and effectiveness, limited know-how required for its use, and improved maintainability, since data model updates do not require a compilation activity to be carried out.

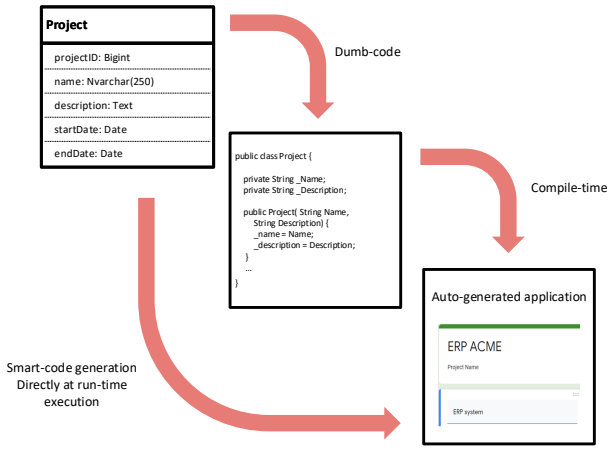

Figure 1. Dumb-code vs. smart-code generation [13].

Several contributions can be found dealing with the adoption of MDA for supporting the application development lifecycle, such as [14–16]. In [14], a model-driven approach for supporting the modelling of a complex system is presented. Specifically, the paper introduces MUVIEMOT, a set of domain-specific modelling languages for the specification and model-driven development of multiview modelling tools. In this respect, E-MDAV does not focus on multiview modelling of systems and mainly focuses on the development, deployment and maintenance of web applications. In [15], an MDA extension, named Context-aware Quality Model Driven Architecture (CQ-MDA), is introduced to address quality control in pervasive computing environments. The proposed approach extends MDA and introduces two novel model layers to explicitly consider quality and resource awareness in the development process, namely the CPIM (Contextual Platform Independent Model) and the CPSM (Contextual Platform Specific Model), which extend the PIM and the PSM, respectively. Differently from such contributions, E-MDAV is based on a *tailoring* of MDA rather than an extension. The proposed methodology takes into consideration a PIM-to-PSM transformation in order to support the automated generation of web application code. Furthermore, E-MDAV considers reverse-engineering issues to keep the application models aligned with the implementation.

In [16], a model-driven approach for developing web applications is discussed. Specifically, the paper introduces a supporting tool, namely *Model-driven Architecture for Web Application (MoDAr-WA)*, which implements a complete transformation chain, from the highest MDA model layer, e.g., the Computation Independent Model (CIM) to the lowest one, e.g., the executable code. The obtained application implementation is shown to conform to the Model-View-Control (MVC) web paradigm. Differently, the E-MDAV methodology introduced in this paper allows for the generation of the entire web ap-

plication (including the user interface) both in a forward-engineering approach, from a UML-based data model, and in a reverse-engineering approach, deriving such a data model from an existing database.

Among other relevant contributions, the need for concrete model-driven development approaches to support reverse-engineering activities has been underlined in [17], which presents an extensive literature review to discuss the state of the art for what concerns the adoption of model-driven approaches to support reverse engineering.

Regarding modelling-related issues, the E-MDAV framework adopts UML (Unified Modelling Language), which is widely acknowledged as the standard modelling language for implementing model-based approaches in the software development field [18]. As stated in [19], UML is a mature enabling tool for database design. The UML language enables conceptual, logical, and physical database definition, by use of its standard profiling-based extension mechanism, which enables the UML specification of ad hoc data definition languages (DDLs) for specific database management systems (DBMSs). Regarding the notation used for serializing data, physical database models share common features with XMI, the XML Metadata Interchange OMG's standard for exchanging metadata information via XML, as reported in [20], where an XMI-based serialization has been adopted for reverse-engineering purposes. E-MDAV makes use of both to extract the representation of the application model, in terms of UML classes, relationships, attributes, and additional model elements, and to use them to generate the web application at run-time.

Finally, as mentioned in Section 1, it is worth noting that E-MDAV extends the MDA-VIEW framework illustrated in a previous contribution [21]. The MDA-VIEW framework focuses on the automated generation of a web visual interface to an existing database. Thus, MDA-VIEW requires an existing database implementation from which a data model is first derived and then used for driving the generation of the web application code. In this respect, MDA-VIEW does not provide any support for ensuring the consistency among the several artifacts and between the model and the application. Differently, E-MDAV provides a complete MDA-based methodology that fully supports the development and maintenance of data-intensive web applications, taking into account both the forward- and the reverse-engineering processes. The E-MDAV framework exploits a model transformations chain that significantly reduces the effort required for the generation and configuration of the target web application, ready to be deployed onto a cloud-based platform. E-MDAV has been inspired by DevOps' principles, to reduce the gap between development and operation stages and thus enact frequent release cycles. Moreover, differently from MDA-VIEW, the E-MDAV implementation is fully based on open-source technologies. Further details which clarify how E-MDAV overcomes the most relevant limitations of the previous MDA-VIEW framework are provided in Sections 5.1 and 5.2.

3. Background

This section provides the necessary background about the main concepts at the basis of this paper, i.e., model-driven development, DevOps, and cloud computing.

3.1. Model-Driven Development and Model-Driven Architecture

Model-driven development (MDD) is an approach to software design and implementation that addresses the rising complexity of execution platforms by focusing on the use of formal models [22]. The founding pillars of MDD are constituted by model transformations. An appropriate chain of *model-to-model* and *model-to-text* transformations is specified and executed in order to progressively translate abstract models into more refined models until executable artifacts or models that meet the desired level of abstraction are generated.

One of the most important initiatives driven by MDD is the *Model Driven Architecture (MDA)*, the Object Management Group (OMG) incarnation of MDD principles [9].

The Model-Driven Architecture prescribes that various models have to be specified throughout the application development lifecycle. Such models, which provide different views of the system, are specified from viewpoints focusing on particular system concerns.

MDA identifies three different viewpoints: the computation-independent viewpoint, which focuses on the system requirements and its environment, the platform-independent viewpoint, which describes the system operations hiding any detail on the underlying execution platform, and finally, the platform-specific viewpoint which extends the platform-independent viewpoint with the description of a specific execution platform. The models defined from such viewpoints are denoted as Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM), respectively.

The application development process is thus carried out by executing an appropriate set of model transformations to transform a CIM to a PIM and then a PIM to one or more PSMs. Finally, PSMs are used to generate the actual executable application code.

As underlined in Section 5, the development process at the basis of E-MDAV has been defined as a tailoring of MDA, which specifically focuses on the implementation of web applications. Specifically, relevant artefacts are obtained by executing a set of *model-to-text* transformations which take as input a UML data model constituting the application PSM. Moreover, E-MDAV also deals with the reverse-engineering process and exploits a *model-to-model* transformation for generating the data model from the ER model of an existing database.

The following main standards have been introduced as part of the MDA effort:

- *Meta Object Facility (MOF)*: for specifying technology neutral metamodels (i.e., models used to describe other models) [23];
- *XML Metadata Interchange (XMI)*: for serializing MOF metamodels/models into XML-based schemas/documents [24];
- *Query/View/Transformation (QVT)* and *MOF Model To Text (MOFM2T)*: for specifying *model-to-model* and *model-to-text* transformations, respectively [25,26].

Figure 2 outlines how MDA standards are related to each other. The input of the model-driven development process is constituted by Model M_A, which is an instance of metamodel MM_A. The model transformation generates as output model M_B, which, in turn, is an instance of metamodel MM_B. Both MM_A and MM_B are specified in terms of *MOF Model* constructs. The model transformation that generates the expected output is specified by use of QVT, the declarative language for specifying model-to-model transformations. In order to be handled by the QVT Transformation Engine, the input and the output models have to be serialized as an XML-based document which is obtained by applying the rules specified by the XMI standard. The XMI standard is also used for deriving the XMI schemas (i.e., MM_A XMI Schema and MM_B XMI Schema) from relevant metamodels to be used for validating XMI documents. Finally, if the output model is of text type (e.g., executable code, text documents, configuration files, etc.) the required model-to-text transformation is specified by use of the *MOFM2T metamodel* standard.

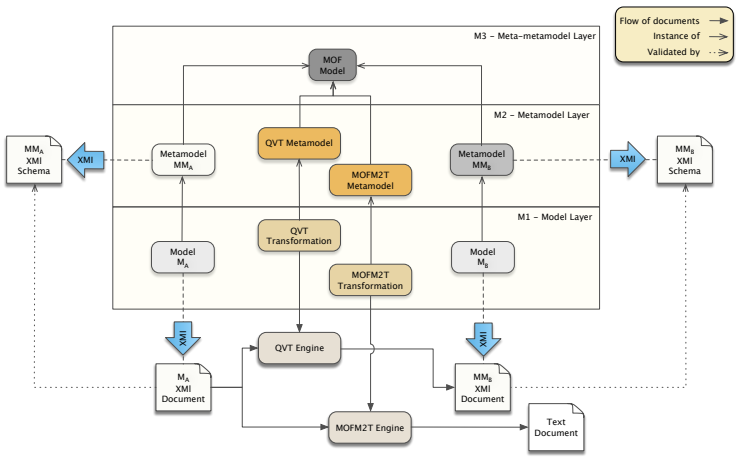

Figure 2. Overview of MDA standards.

3.2. DevOps

DevOps (Development and Operations) is a software lifecycle-management approach that aims at obtaining the agile integration of development, delivery and IT operations [7].

The most relevant benefit that DevOps obtains is a better communication and collaboration between development and operations areas, reducing problems caused by teams' miscommunication and improving the final product's quality and value [27].

DevOps is founded on the following pillars [28], as outlined in Figure 3:

- *Continuous Integration*, that is, the practice in which software components, developed by different teams, are regularly integrated so as to progressively obtain the whole system;
- *Continuous Testing*, that consists of the automatic execution of a test suite whenever a software change is released;
- *Continuous Deployment*, that focuses on the automation of the deployment process;
- *Continuous Delivery Pipeline*, which is the result of DevOps application. The Continuous Delivery consists of the execution of a workflow that, for each software release, introduces automation to support builds' release, test execution and deployment activities.

In this context, the E-MDAV framework can be considered a valuable option for supporting the DevOps application. Specifically, model transformations can be seen as an effective tool to automate and support the continuous delivery pipeline:

- The development of each software component can be obtained or supported by model transformations that are used to generate a large part of the source code from the abstract model, thus reducing the cost and time required for the implementation, improving the quality of the obtained code, and ultimately, easing the system's continuous integration process;
- Testing activities can be automated by generating test data and test scripts, so as to effectively support the system's continuous testing process;
- Deployment scripts can also be automatically generated by execution platform models of the software to be developed in order to ease the continuous deployment and delivery process.

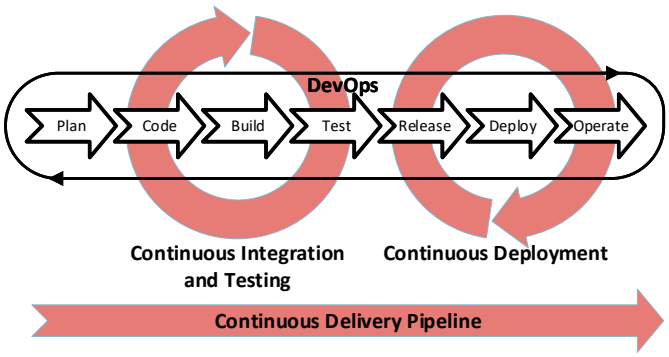

Figure 3. DevOps with Continuous Integration, Continuous Deployment and Pipeline.

3.3. Cloud Computing: Overview and Delivery Paradigms

Cloud computing is a computing paradigm in which a pool of resources such as computing nodes, storage, network connections and entire applications, are provided as services available throughout the Internet [2]. According to such a paradigm, users are enabled to identify and configure the required set of computing resources in an *on-demand* and *pay-per-use* fashion, to dynamically satisfy the operational requirements of the needed computational infrastructure. As a consequence, system and application developers are not required to deal with the setup and maintenance of costly and complex computing

infrastructures, which are instead made available as services in the cloud, also providing the highest level of scalability [29].

Resources in the cloud are provided according to an *everything-as-a-service* delivery model, which can be specified as follows [30]:

- *(Software as a Service, SaaS)*: provisioning of entire applications;
- *(Platform as a Service, PaaS)*: provisioning of development platforms;
- *(Infrastructure as a Service, IaaS)*: provisioning of infrastructures in terms of storage and computing nodes.

The cloud computing paradigm is founded on the concept of *containerization*. A container is a self-contained software entity which wraps up the code of a software application along with all needed dependencies so as to make it easily, quickly and reliably deployed and executed in the cloud. Containers can be moved from an environment to another and provide an approach for virtualizing an operating system as multiple containers that can be deployed and executed over the same operating system instance. Compared to *virtual machines*, which provide a different hardware virtualization approach to execute multiple operating system instances over the same platform, containers constitute a lightweight and more portable alternative [31].

4. Application Development Challenges in Data-Intensive Domain

The ever-increasing availability of broadband network connections and powerful mobile and internetworked devices, the recent achievements in the big data domain, as well as the actual affirmation of the cloud computing paradigm that makes the storage and computational capability virtually unlimited, pushes the relevance of data in application development [32]. Scientists and practitioners must tackle the emerging challenges in the data-intensive computing domain so as to identify effective and scalable approaches to address the most relevant problems [33].

There is a lack of common agreement in defining the *data-intensive* concept. In this respect, this work takes into consideration the taxonomy proposed in [34], as summarized in Figure 4. According to such a contribution, the software development is a challenging task, whose complexity addresses two different directions: the computational complexity and the data complexity. Regarding the data complexity, the concept of being *data-intensive* is introduced to identify those applications which need to handle large and heterogeneous amount of data, also including the case of distributed sources of information.

According to such a perspective, this paper focuses on the development of a data-intensive application which must be able to interact with a heterogeneous database structured in a large number of tables and entities, and must not require the adoption of complex computation algorithms (top-left corner of Figure 4).

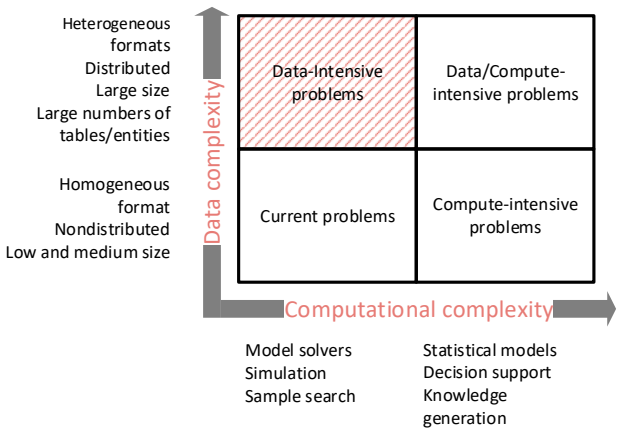

Figure 4. Data and computational complexity in Software Development.

In this domain, such data-intensive applications very often have to be available to a large number of users (e.g., as for e-government applications), who need to be provided with interfaces for easing the access of remote databases. Indeed, in order to achieve the required scalability as well as to ensure the ability of interacting with distributed source of information, the addressed application has also to be *web-based* and *cloud-ready*.

As discussed in Section 5, the E-MDAV framework has been designed to address the above-mentioned scenario. Specifically, the proposed method leverages *automation-based techniques* in order to effectively support the development of data-intensive web applications which are ready to be deployed onto a cloud infrastructure according to an SaaS service model.

Finally, it should be underlined how the addressed data-intensive development scenario raises several issues and challenges:

- In the development of applications interacting with large databases, repetitive and error-prone activities for developing the executable code to create (and also read to and write from) hundreds of tables and fields have to be carried out. Such scenarios should significantly benefit by the effective adoption of automation-based techniques;
- The testing environment usually requires a huge hardware resources stack for replicating the operation environment due to the large amount of data stored in the databases;
- Traditional software-development approaches might fail when smooth cooperation between development and IT operations are required to achieve frequent releases and reduce the time to market.

In this respect, as an answer to the above-mentioned issues and challenges, E-MDAV is founded on the use of automation-based techniques, has been designed to develop cloud-ready applications and promotes the use of DevOps-based approach. Specifically, DevOps and automation have been identified as effective solutions [7,35] to achieve (i) the quality-aware development and continuous delivery of products and (ii) the effective handling of products' maintenance and improvements.

5. E-MDAV Development Process

This section discusses the methodology which constitutes the conceptual basis of E-MDAV. As stated in Section 1, the proposed framework has been designed as an extension and a significant improvement of the MDA-VIEW preliminary release, which has been introduced in [21].

For the sake of completeness, the next section briefly outlines the rationale and the objectives of MDA-VIEW. In order to highlight the advantages brought by E-MDAV, the most relevant issues experienced using the preliminary version of the framework are also outlined.

5.1. MDA-VIEW Objectives and Limitations

The MDA-VIEW framework shares the same objective of E-MDAV, which is to provide automated support for generating data-intensive web-based applications. More specifically, the MDA-VIEW framework is inspired by MDA and aims at supporting the rapid development of web-based applications for intensive data management. As outlined in Figure 5, the MDA-VIEW tool takes as input the actual implementation of a relational database, in order to first determine the relevant data model and then use the latter for generating *at run time* the application, without requiring the production of any additional code. This kind of approach is called *smart-code generation* in comparison to the *dumb-code generation*, in which the application's source code is first generated and then manually compiled to produce the executable code.

According to the above-mentioned idea, the main input used by the MDA-VIEW tool for generating a web application is the physical implementation of a relational database. This constitutes a valuable solution for generating the executable code of an application constituting the database application interface, especially when the data-intensive domain is addressed.

Despite the several benefits derived from the application of such a framework in real cases, as has been largely discussed in [21], some issues have to be underlined, which push the need for developing a new framework release:

- Notwithstanding the framework rationale has been largely inspired by MDA's principles, the tool implementation includes the use of commercial and non-MDA-compliant components and technologies;
- The proposed methodology requires an existing database implementation.
- MDA-VIEW does not specifically introduce automation to ensure consistency among the several artefacts developed along the entire application lifecyle: data model, database implementation, web application;
- MDA-VIEW does not provide any guidance for supporting the deployment, the testing or the operation of the web application.

In order to face the above-mentioned limitations, the MDA-VIEW has been extended, as detailed in the next section.

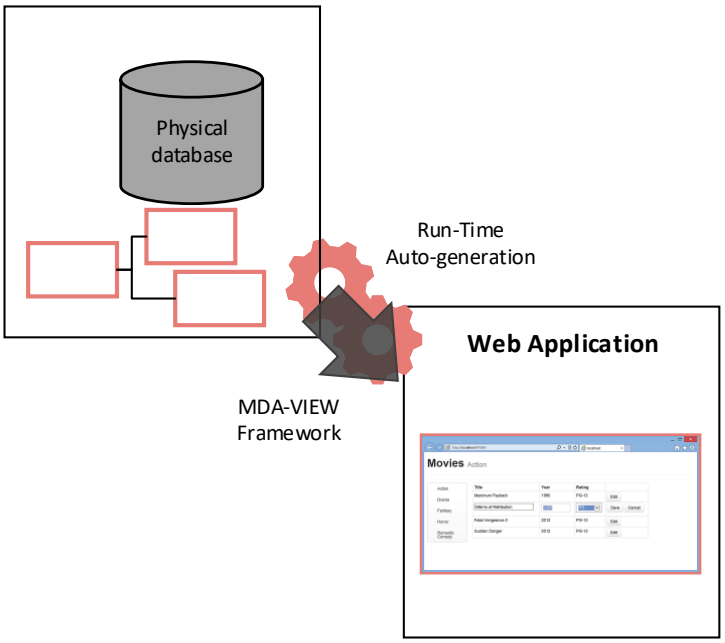

Figure 5. MDA-VIEW Development Process.

5.2. E-MDAV Rationale and Benefits

The E-MDAV development process' rationale, which has been designed to tackle the MDA-VIEW shortcomings highlighted in the previous section, is depicted in Figure 6.

In this respect, as stated in Section 1, in order to adequately face the needs of the addressed domain, the methodology has been based on the concepts of *automation* and *continuous delivery and deployment*. The underlying development process has been inspired by DevOps' principles, thus several E-MDAV activities have been mapped to the DevOps phases. Thus, the main objective of E-MDAV is to ease and improve the software application development process. As underlined in Section 3, the methodology illustrated in Figure 6 has been designed as a *tailoring* of the general MDA development process, which has been *inspired* by DevOps. Specifically, the proposed methodology mainly focuses on the generation of the web application implementation. The core of the development process is constituted by the *model-to-model* and *model-to-text* transformations, provided by E-MDAV, which take as input the web application data model and yield as output the required set of executable artefacts, along with the files needed to support the execution of testing activities, creating the actual DB Schema implementation and handling the application deployment over a cloud-based infrastructure. In this respect, the web application data model, according

to the MDA terminology, constitutes the PSM. The model, which is specified as a UML class diagram, is annotated by an appropriate UML stereotype, to include the information needed for driving the subsequent execution of the *model-to-text* transformation. Differently, regarding the reverse-engineering process, a *model-to-model* transformation maps the PSM representing the ER DB Schema to the corresponding PSM representing the data model. Note that E-MDAV can be easily extended by introducing appropriate *model-to-model* transformations to generate PSMs from annotated CIMs/PIMs.

It should be underlined that, as stated in Section 1, this paper is mainly framed around a *conceptual* and *methodological* layer. Even if the E-MDAV methodology *assumes* the availability of a concrete supporting tool, which provides the above-mentioned model transformations along with the needed operational environment, its implementation is out of the scope of this work. Differently, this paper proposes a reference architecture that aims at driving the implementation of a supporting tool. Finally, as clarified in the next section, in order to evaluate both the methodology and the reference architecture, a *tool prototype* has been developed. In this respect, Section 7 discusses the application of E-MDAV to two application scenarios.

Figure 6 also shows the artifacts produced at each step and the actors, tools and IT resources involved in the task execution, and clarifies the several stakeholders involved in each task.

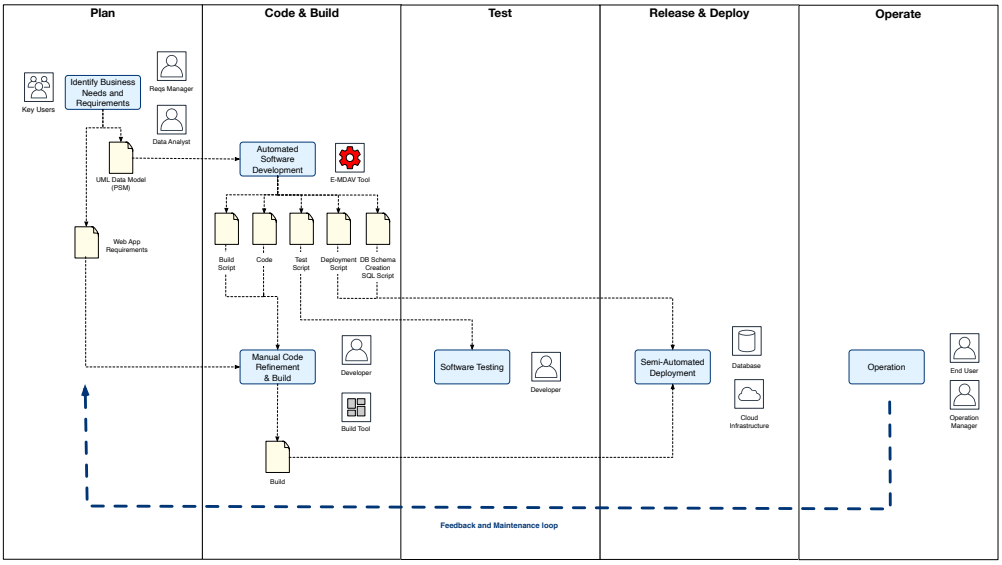

Figure 6. E-MDAV Development Process.

- **Business Needs and Requirements Identification:** during the first step, a set of interviews are taken in order to identify the web application *requirements* and to specify the *abstract data model* in terms of a UML Class Diagram representing the most relevant domain entities and their relationships.
- **Automated Software Development:** at this step, the several artefacts needed to support the application implementation and deployment are generated by use of a chain of `model-to-model` and `model-to-text` transformations. In this respect, E-MDAV effectively supports the IT specialists directly involved in the application development, which are not required to build the needed artefacts from scratch. Specifically. the E-MDAV tool takes as input the UML data model and generates as output:

  1. The *source code* of the web application and the *SQL scripts* for creating the related DB schema. It should be underlined how in the *data-intensive* domain, software developers greatly benefit by the automation of these steps, as they are raised by the execution of repetitive, effort-consuming and error-prone writing of the needed code for the CRUD operations;
  2. The *test scripts*, which will be used for the application testing;

3. The *build scripts*, needed to automatically build the executable package(s) to be deployed;
4. The *deployment scripts*, as the proposed methodology includes the support for automating the creation and the deployment of needed containers according to the specific cloud infrastructure adopted for hosting and executing the software.

- **Manual Code Refinement and Build:** even though the MDA-based automated code generation eases the development process and reduces the required effort, a manual refinement of the generated code is always needed. It should be underlined that while it is acknowledged that *model-to-text* transformations can be effectively used for generating stubs, skeletons and well-focused code snipplets (e.g., those needed for creating the interface for reading and writing a large number of different fields from/to a database), such approaches might fail in generating the implementation of complex business logic and behavioural algorithms, which always require the human intervention to be implemented. An MDA-based methodology requires that the input software model (i.e., the Platform Independent Metamodel—PIM) is *annotated* so as to include the information needed to drive the chain of model transformations. The process of annotating the PIM (so as to obtain the so-called marked PIM), which is not part of traditional development processes, might be so complex and effort-consuming to be as costly as the manual software development. Thus, the adoption of an MDA-based development approach allows us to obtain a proper balance between *automation* and *manual development*, in order to maximise the benefits while avoiding any time or budget overruns.
- **Software Testing:** this activity is supported by the script generated by the E-MDAV tool.
- **Automated Deployment:** this step deals with the deployment of the software packagers onto the chosen cloud infrastructure, and also with the creation of the physical database. Both activities are supported by the scripts generated by E-MDAV and make use of specific tools provided by related DB and cloud software vendors.
- **Operations:** during the software operation, several reasons e.g., corrective, perfective or adaptive software maintenance, might lead to a new development iteration. When a new process iteration is started, the transformation chain allows all the involved stakeholders, from business managers to developers and operational managers, to promptly revise, complete and deploy a new application version, reducing development time (and costs) and making sure to keep models constantly aligned to the implementation.

In order to appropriately address the challenges introduced in Section 1, the E-MDAV methodology has been structured to support the various activities that encompass the application's development lifecycle, with the objective of becoming an effective DevOps enabler. In this respect, one of the most relevant objectives of E-MDAV is the achievement of *continuous delivery* by seamlessly integrating the application development (Dev) with the application operation (Ops). Specifically, E-MDAV introduces appropriate model transformations for supporting (i) the implementation of the required web application, (ii) the relevant testing activities and (iii) the application deployment and configuration, as pointed-out in Section 5.2.1. Finally, during the application operation, E-MDAV also provides appropriate support for addressing the application's maintenance and modification, as highlighted in the *Application Change Request* scenario discussed in Section 5.2.1.

In this context, stakeholders need to be supported by automated tools that enable their effective interactions along the whole development process [7]. In this respect, with E-MDAV:

1. Data and application models are used to identify and specify the application requirements and features;
2. The required software artifacts are generated throughout automated procedures;
3. The achieved degree of automation provides effective support in managing each application release as testing and deployment phases are specifically addressed by the automated generation of the required scripts;

4. In case of issues affecting a new release, the proposed methodology also supports the execution of roll-back activities for restoring a previous application version.

In order to better highlight how the proposed methodology can effectively support different development needs, the next Section 5.2.1 discusses two operational scenarios. The evaluation of the methodology and its application in a concrete case is also discussed in Section 7.

5.2.1. Application Scenarios

In order to describe how the proposed methodology can be applied to concrete cases, this section illustrates two application scenarios, namely Direct Application Development and Application Change Request.

Direct Application Development

This scenario is related to the development of the web application which is needed and promoted by relevant organization's stakeholders. As shown in Figure 7, the required flow of activities executed to achieve the scenario objectives follow the process discussed in the previous section. Once the web application requirements have been identified and the UML abstract data model has been specified, application developers make use of the required tool for building and testing the application. In this respect, the automation of such activities is founded on the use of the E-MDAV tool, which is able to generate the application source code, the database schema and the set of configuration scripts needed for supporting the testing and the deployment. Regarding the application deployment, it should be underlined that the E-MDAV methodology focuses on the development of *cloud-ready* application. According to this perspective, it is assumed that the execution platform is available as a cloud-based infrastructure. From a conceptual point of view, the E-MDAV methodology is not tied to any vendor-specific cloud infrastructure or DBMS. The required degree of flexibility is inherently achieved by the adoption of MDA standard and technologies. The E-MDAV tool's outputs are generated by executing a chain of model transformations, whose last step is constituted by a *model-to-text* transformation in charge of mapping the PSM to vendor/technology-specific code and scripts. Thus, different *model-to-text* transformations have to be provided in order to ensure the compliance with different execution platforms and technologies. Specifically, the development script (e.g., a YAML descriptor for building the needed Docker container) is given as input to the cloud-specific deployment interface. Similarly, SQL scripts are executed by the relevant SQL engine for generating the DB schema which has to be connected to the web application.

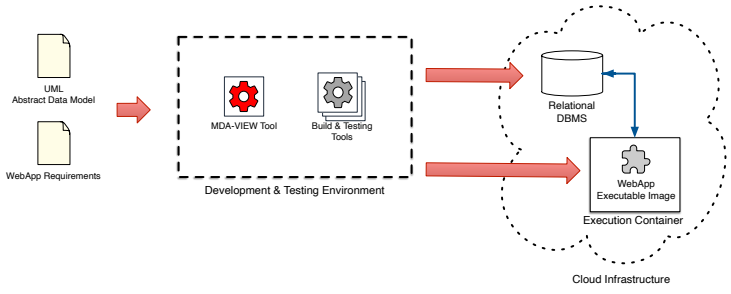

Figure 7. Development of a web application.

Application Change Request

This scenario takes into consideration an existing application which needs to be modified. Indeed, a change request constitutes the trigger of the subsequent evolutive/corrective maintenance process to be addressed. Several reasons may lead to a change request, e.g., a final user who detects an error might claim for a corrective maintenance, or a manager or an external partner which needs a new application's feature might request to start an evolutive maintenance process. Figure 8 summarizes such a scenario. The first task to be executed is the generation of a data model which correctly maps the existing physical

database. It is worth noting that this step is required whether an abstract model is not available or if such a model is no longer aligned with the DB schema (e.g., due to a change in the database which has not been followed by the related conceptual model update). Once the *as_is* data model is derived, data analysts and software engineers proceed to the change request analysis so as to specify the updated data model which includes the required changes. Then, a new development activity is carried out, similarly to what has been discussed in the previous scenario. Finally, the updated application is ready to be redeployed for the operation.

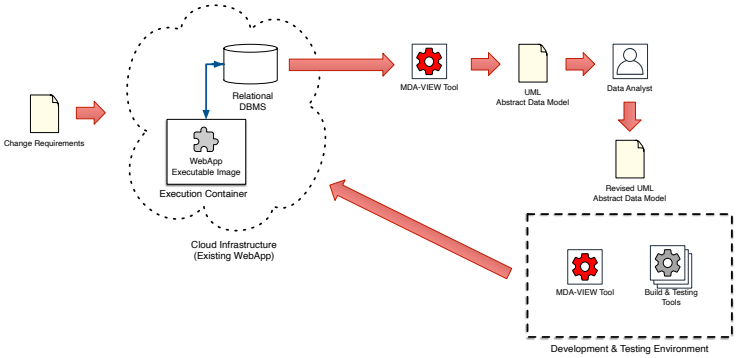

Figure 8. Change Request of an existing web application.

6. E-MDAV Reference Architecture and Prototype Implementation Details

As stated in Section 1, in addition to the identification of a methodology that addresses the development of web-based data-intensive applications, this paper also aims to propose an abstract architecture for the design and implementation of a supporting tool. In this respect, this section introduces the proposed E-MDAV reference architecture and provides some details about the technologies adopted for the implementation of a prototype version of the aforementioned supporting tool.

The E-MDAV architecture, as depicted in Figure 9, consists of three layers: the uppermost one provides the implementation of user interface components, the lowermost layer includes the databases and the needed software for ensuring the data persistence, whereas the intermediate layer, which is hereafter discussed, provides the implementation of the application's logic.

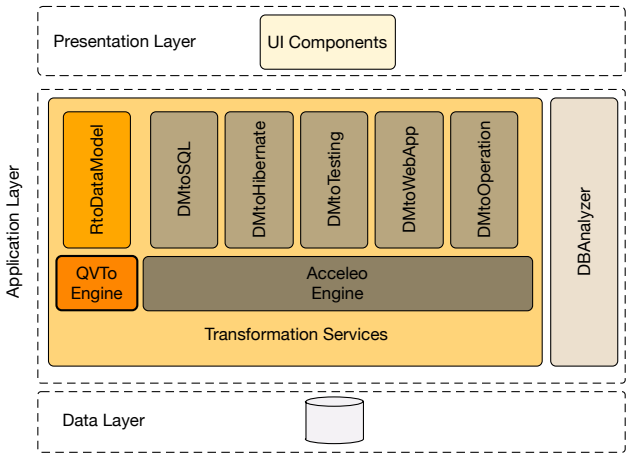

Figure 9. E-MDAV Tool Abstract Architecture.

It is structured into the two following main components:

- **DBAnalyzer**: is the component responsible for analysing the structure of an existing relational BD to generate the XMI representation of the *Entity-Relationship (ER)* model.

- **Transformation Services**: provides the implementation of the model transformations at the basis of the E-MDAV methodology, as summarized in Figure 6. More specifically, the *model-to-model* transformation has been specified by using the QVT [25] standard, while for its implementation and execution, the QVTo Eclipse project [36], which provides a complete implementation of the OMG QVT operational language, has been used. The specification of the several *model-to-text* transformations have been based on the MOF Model-to-Text (MOFM2T) standard [26], while the related implementations have been founded on Acceleo [37], the Eclipse plugin which provides an implementation of the MOFM2T standard.
  - ***RtoDataModel***: is the *model-to-model* transformation that maps an ER model to a UML model representing the application's data model. As discussed in Section 5.2.1, it is used in a *reverse-engineering* process to support the maintenance of an existing web application, or in a wider and more general perspective, to allow the generation of a data model from an existing database, so as to enable the on-going adoption of the proposed E-MDAV methodology even when the application implementation has been already completed.
  - ***DMtoSQL***: is the *model-to-text* transformation that takes as input the application's data model specified in UML and yields as output the SQL scripts needed for generating the database schema.
  - ***DMtoHibernate***: is the *model-to-text* transformation responsible for generating the web application component required for handling the data persistence. In this respect, as discussed later on in this section, as the web application architecture includes the Hibernate [38] technology, the *DMtoHibernate* transformation generates the required code and scripts for configuring the web app's hibernate component, starting from the application's data model.
  - ***DMtoWebApp***: is the *model-to-text* transformation that takes as input the application's data model and generates as output the code of the web application. Such a transformation has been designed so to use the Java language and to exploit the Apache Velocity technology [39] to ease the implementation of the presentation layer according to an MVC pattern.
  - ***DMtoTesting***: is the *model-to-text* transformation responsible for generating the JUnit components for supporting the execution of the required tests that, according to the E-MDAV methodology, have to be carried out before the release of each component to the operational environment. The input of the transformation is constituted by the UML Data Model.
  - ***DMtoOperation***: is the *model-to-text* transformation for generating the script for enacting the automated deployment of the application to the operational environment. It should be underlined that, as discussed in Section 5, the adoption of a transformation-based approach makes E-MDAV flexible enough to support the deployment of the web application both *on premises* and on a cloud infrastructure.

As a result, Figure 10 shows the logical architecture of the web application generated by adopting an E-MDAV tool that actually implements the above-mentioned abstract architecture.

The web application is founded on a three-tier architecture whose data layer includes a relational DB, which, in turn, uses an Hibernate engine for ensuring data persistence. The web app requires an application container to be deployed on, and its implementation has been based on the MVC pattern. In order to clearly separate the implementation of the *view* and the *model*, the web app implementation makes use of the Java-based template engine Apache Velocity. It should be underlined that to ensure the highest degree of flexibility, the *model-to-text* transformations used for generating the application code and the related scripts do not hard-code any reference to a specific database or container. An appropriate configuration allows for the actual specification of the needed technology.

In order to evaluate the applicability of the E-MDAV methodology and the technical soundness of the reference architecture, a tool prototype compliant with such a reference architecture has been implemented.

Specifically, the prototype implementation has been based on the Eclipse Modeling Framework (EMF) [40] and includes:

- The preliminary implementation of the various model-to-model and model-to-text transformations on the basis of the *Transformation Services* and the *DB Analyzer* components, as depicted in Figure 9;
- The *model-to-model* transformation engine QVTo [25], which is available as an Eclipse plugin;
- The *model-to-text* transformation engine Acceleo [37], which is available as an Eclipse plugin.

The next section discusses its application in two real cases.

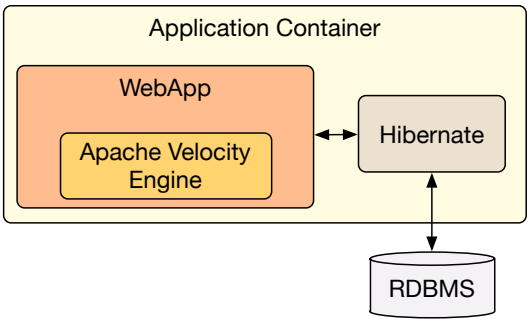

Figure 10. Architecture of the generated web application.

7. Evaluation of the E-MDAV Framework

As stated in Section 1, this paper addresses the development of web-based and data-intensive applications from different points of view. First, the paper focuses on the methodological perspective, and in this respect, investigates how a model-driven approach can be effectively introduced in the development process, as outlined in Section 5. Moreover, this work has also described how the E-MDAV methodology can actually be implemented and executed; in this respect, the reference architecture for a supporting tool has been illustrated in Section 6. Finally, to showcase the applicability and the validity of the proposed methodology, and to assess the soundness of the proposed reference architecture, this section discusses the results of the E-MDAV application to a concrete case.

In this respect, to achieve such an objective, a prototype implementation of a support tool, based on the reference architecture illustrated in Figure 9, has been developed.

It is worth noting that the validation hereby illustrated didn't mean to deal with field tests and collect data for assessing the prototype performance or for evaluating the concrete technologies adopted for its implementation. Differently, the E-MDAV prototype has been developed as a *proof-of-concept* to support the actual execution of the activities outlined in Figure 6, so as to assess the technical soundness of both the methodology and the reference architecture.

In this respect, the following application scenarios have been considered.

1. **Web Application for Tax Management**: this scenario deals with the development of a data-intensive web application for the land registry service of an Italian city with a population of over 500,000 citizens. The web application shall collect and manage data related to property taxes and land management and shall ensure 24/7 operations. The data model that constitutes the main input to the E-MDAV development process counts over 300 entities. Finally, the software implementation shall adopt a cloud-based PaaS (Platform-as-a-Service) execution platform.
2. **Collaborative Information Systems (CIS)**: this scenario focuses on the development of a *project management* application, which shall effectively support the planning,

execution and monitoring of software development project. The application shall also be based on the the most relevant project management approaches e.g., Project Management Body of Knowledge (PMBOK) [41] and PRojects IN Controlled Environments (PRINCE2) [42]. The software application shall be available as a service in the cloud, according to a Software-as-a-Service (SaaS) paradigm delivery model.

In both scenarios, the application development provides the opportunity to exploit a DevOps process that can be implemented by using a *Continuous Integration (CI)* environment, such as Jenkins. In such a case, the data model, which is developed by use of an appropriate tool (e.g., Enterprise Architect) constitutes the main input given to E-MDAV. The artefacts generated by the E-MDAV tool (as discussed in Section 5) are given as input to the CI environment, which is charge of enacting the CI pipeline and handling the various releases (e.g., alpha, beta and production), as outlined in Figure 11.

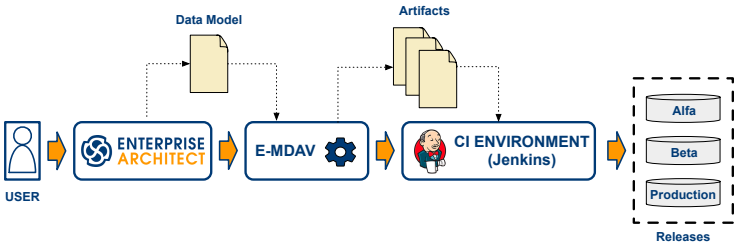

Figure 11. E-MDAV in a DevOps CI pipeline.

The evaluation process has been based on the following strategy:

- The effort spent by each professional role has been measured;
- Historical data related to past projects addressing a similar context and related to the development of applications with similar complexity have been collected;
- A comparison has been carried out in order to determine possible benefits that the application of the E-MDAV methodology is able to lead to.

Specifically, regarding the above-mentioned application scenarios where the E-MDAV methodology has been applied, Table 1 outlines the roles and their main responsibilities, whereas Figure 12 shows the related effort distribution. The effort distribution refers to the example scenario dealing with a web application for tax management implementation. Data have been retrieved from the company's accounting system, which reports the number of hours spent daily by each team member, according to the professional role and the activities carried out.

Regarding the analysis of historical data, all the considered development projects dealt with applications whose architectures are based on the MVC design pattern. According to such a design strategy, a software application is structured into three different layers, i.e., the *Model*, which represents the state of the system, the *View*, which deals with the visual representation of the state, and the *Control*, which handles the input issued by users to possibly update the system state [43].

Table 1. Roles and responsibilities.

Role	Responsibilities
Business Analyst	Requirements Elicitation, Functional Analysis
Software Designer	Data and Application Model Specification
Software Developer	Implementation, Testing

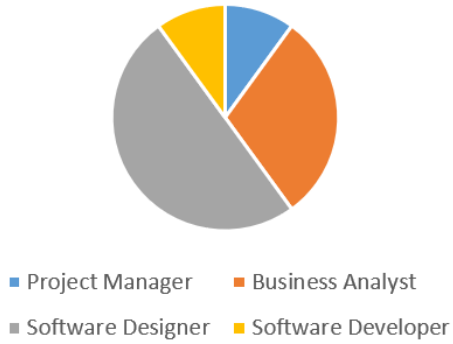

Figure 12. Effort distribution for different roles (E-MDAV case).

Specifically, historical data have been collected by using the projects account systems over the past 5 years. The related effort distribution is shown in Figure 13.

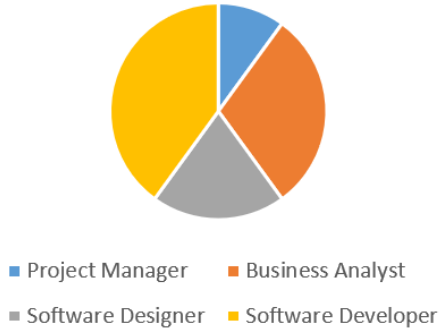

Figure 13. Effort distribution for different roles (Traditional MVC approach).

Finally, Figure 14 compares the effort required for developing the addressed applications in the above-mentioned cases: adopting a traditional MVC approach (blue bar) or adopting the proposed E-MDAV frawework (orange bar).

The *COnstructive COst MOdel (COCOMO)* has been used to estimate the effort of both the application of a traditional approach and the adoption of the proposed E-MDAV methodology. Specifically, the *intermediate model* and the *organic development mode* have been used for COCOMO application. The obtained effort estimation for the E-MDAV case has then been validated by collecting real data at project execution time.

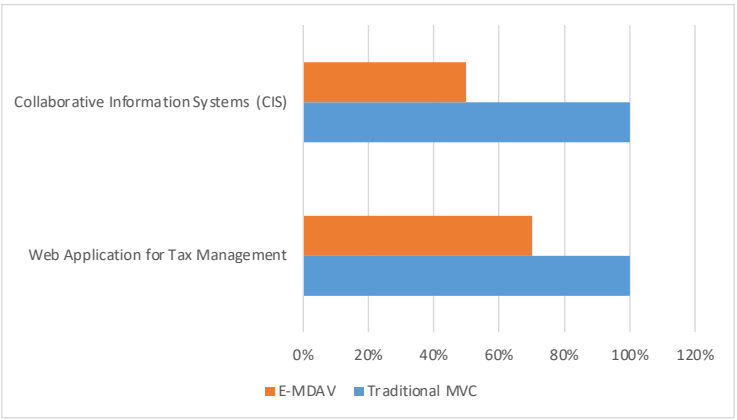

Figure 14. MVC and E-MDAV project effort comparison.

The conducted analysis highlights how the application of E-MDAV leads to a significant reduction in the application development cost. Moreover, the inherent automation the MDA eases the several activities encompassed by the development process, and ultimately,

allows for enhancing the effectiveness of the development process and the quality of the software product. Specifically the following benefits and goals can be obtained by the E-MDAV adoption:

- Improved consistency among artefacts produced at the various development process steps thanks to the adoption of a transformation-based approach, even when maintenance activities (e.g., artefacts/code changes) have to be carried out;
- Reduction in the required effort, time and cost;
- Reduction in the *time to market*;
- Improved cooperation between Dev and Ops teams as both are supported by the automation provided by E-MDAV;

8. Conclusions

This paper has introduced *E-MDAV (Extended MDA-VIEW)*, a framework for supporting the development process of *data-intensive* and *web-based* applications. In order to be able to effectively support agile development processes such as DevOps, E-MDAV has been founded on the pillar of *automation*. In this respect, methods and tools from the OMG's Model Driven Architecture have been used in order to implement a flow of model transformation able (i) to generate the required artefacts from abstract data models of the application to be developed, and (ii) to take as input the schema of an existing database for generating the corresponding data model and provide an effective support for conducing the application maintenance and/or reverse-engineering-related activities.

E-MDAV has been evaluated in two different application scenarios and the effort required for implementing the related software applications has been measured. Finally, a comparison with data gathered from previous projects has revealed how the adoption of E-MDAV leads to a different distribution of the effort among the various involved professionals, and ultimately, to a significant reduction in the required development time and costs. Moreover, the application of E-MDAV also allows for the achievement of several nontangible benefits such as a more effective development process, a reduced time to market and an improvement in the application quality. As the evaluation has been based on a prototypal implementation, ongoing work includes the development of a fully fledged supporting tool and the execution of a more extensive evaluation campaign.

Author Contributions: Original idea, A.D. and T.P.; conceptualization and methodology, P.B. and A.D.; software, T.P.; validation, analysis and discussion of the results, all authors; writing—original draft preparation, all authors; writing—review and editing, all authors, supervision, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not employ or report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abdel-Basset, M.; Chang, V.; Nabeeh, N.A. An intelligent framework using disruptive technologies for COVID-19 analysis. *Technol. Forecast. Soc. Chang.* **2021**, *163*, 120431. [CrossRef] [PubMed]
2. Mell, P.M.; Grance, T. *The NIST Definition of Cloud Computing*; Technical Report; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2011.
3. Al-Mekhlal, M.; Khwaja, A.A. A Synthesis of Big Data Definition and Characteristics. In Proceedings of the 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), New York, NY, USA, 1–3 August 2019; pp. 314–322.
4. Albert, M.; Cabot, J.; Gómez, C.; Pelechano, V. Automatic generation of basic behavior schemas from UML class diagrams. *Softw. Syst. Model.* **2010**, *9*, 47–67. [CrossRef]

5. Rajavaram, H.; Rajula, V.; Thangaraju, B. Automation of microservices application deployment made easy by rundeck and kubernetes. In Proceedings of the 2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 26–27 July 2019; pp. 1–3.
6. Tolosana-Calasanz, R.; Bañares, J.Á.; Colom, J.M. Model-driven development of data intensive applications over cloud resources. *Future Gener. Comput. Syst.* **2018**, *87*, 888–909. [CrossRef]
7. Ebert, C.; Gallardo, G.; Hernantes, J.; Serrano, N. DevOps. *IEEE Softw.* **2016**, *33*, 94–100. [CrossRef]
8. D'Ambrogio, A.; Falcone, A.; Garro, A.; Giglio, A. On the Importance of Simulation in Enabling Continuous Delivery and Evaluating Deployment Pipeline Performance. In Proceedings of the Italy INCOSE Conference on Systems Engineering (CIISE 2018), Rome, Italy, 28-30 November 2018; pp. 53–59.
9. OMG. MDA Guide Revision 2.0 (ormsc/14-06-01). 2003. Available online: https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf (accessed on 12 January 2022).
10. Hansson, D.H. Ruby on Rails Guides, v.7.01. 2020. Available online: http://guides.rubyonrails.org (accessed on 12 January 2022).
11. Schmidt, D. Guest Editor's Introduction: Model-Driven Engineering. *Computer* **2006**, *39*, 25–31. [CrossRef]
12. Ruiz, F.J.B.; Molina, J.G.; García, O.D. On the application of model-driven engineering in data reengineering. *Inf. Syst.* **2017**, *72*, 136–160. [CrossRef]
13. Brambilla, M.; Cabot, J.; Wimmer, M. Model-driven software engineering in practice 2nd Edition. *Synth. Lect. Softw. Eng.* **2017**, *3*, 1–207. [CrossRef]
14. Bork, D.; Karagiannis, D. Model-driven development of multi-view modelling tools the muviemot approach. In Proceedings of the 2014 9th International Conference on Software Paradigm Trends (ICSOFT-PT), Vienna, Austria, 29–31 August; p. IS-11.
15. Alti, A.; Boukerram, A.; Roose, P. Context-aware quality model driven approach: A new approach for quality control in pervasive computing environments. In *European Conference on Software Architecture*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 441–448.
16. Essebaa, I.; Chantit, S.; Ramdani, M. MoDAr-WA: Tool Support to Automate an MDA Approach for MVC Web Application. *Computers* **2019**, *8*, 89. [CrossRef]
17. Raibulet, C.; Fontana, F.A.; Zanoni, M. Model-driven reverse engineering approaches: A systematic literature review. *IEEE Access* **2017**, *5*, 14516–14542. [CrossRef]
18. OMG. Unified Modeling Language, Version 2.5.1. 2017. Available online: https://www.omg.org/spec/UML/2.5.1/ (accessed on 20 December 2021).
19. Nailburg, E.J.; Maksimchuk, R.A. *UML for Database Design*, 1st ed.; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2001.
20. Iftekhar, N.; Warsi, M.R.; Zafar, S.; Khan, S.; Biswas, S.S. Reverse engineering of relational database schema to UML Model. In Proceedings of the 2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON), Aligarh, India, 8–10 November 2019; pp. 1–6.
21. Panetti, T.; D'Ambrogio, A. A complexity-less approach for automated development of data-intensive web applications. In Proceedings of the 2018 International Symposium on Networks, Computers and Communications (ISNCC), Rome, Italy, 19–21 June 2018; pp. 1–6.
22. Atkinson, C.; Kühne, T. Model-driven development: A metamodeling foundation. *Softw. IEEE* **2003**, *20*, 36–41. [CrossRef]
23. OMG. Meta Object Facility (MOF) Specification, Version 2.4.2. 2017. Available online: https://www.omg.org/spec/MOF/2.4.2/ (accessed on 12 January 2022).
24. OMG. XMI - XML Metadata Interchange, Version 2.5.1. 2015. Available online: https://www.omg.org/spec/XMI/2.5.1/ (accessed on 12 January 2022).
25. OMG. Meta Object Facility (MOF) Query/View/Transformation, Version 1.3. 2016. Available online: https://www.omg.org/spec/QVT/1.3/ (accessed on 12 January 2022).
26. OMG. MOF Model to Text Transformation Language (MOFM2T), 1.0. 2008. Available online: https://www.omg.org/spec/MOFM2T (accessed on 12 January 2022).
27. Virmani, M. Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In Proceedings of the Fifth International Conference on the Innovative Computing Technology (INTECH 2015), Galicia, Spain, 20–22 May 2015; pp. 78–82.
28. Erich, F.; Amrit, C.; Daneva, M. A Qualitative Study of DevOps Usage in Practice. *J. Softw. Evol. Process* **2017**, *29*, e1885. [CrossRef]
29. Buyya, R.; Srirama, S.N.; Casale, G.; Calheiros, R.; Simmhan, Y.; Varghese, B.; Gelenbe, E.; Javadi, B.; Vaquero, L.M.; Netto, M.A.; et al. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–38. [CrossRef]
30. Menascé, D.A.; Ngo, P. Understanding Cloud Computing: Experimentation and Capacity Planning. In Proceedings of the 2009 Computer Measurement Group Conference, Dallas, TX, USA, 6–11 December 2009.
31. Karmel, A.; Chandramouli, R.; Iorga, M. *NIST Special Publication 800-180: NIST Definition of Microservices, Application Containers and Virtual Machines*; NIST: Gaithersburg, MD, USA, 2016.
32. Chen, C.P.; Zhang, C.Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Inf. Sci.* **2014**, *275*, 314–347. [CrossRef]

33. Stavrinides, G.L.; Karatza, H.D. Scheduling data-intensive workloads in large-scale distributed systems: Trends and challenges. In *Modeling and simulation in HPC and cloud systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 19–43.
34. Ian Gorton, D.K.G. *Data-Intensive Computing: Architectures, Algorithms, and Applications*; Cambridge University Press: Cambridge, UK, 2012. [CrossRef]
35. DICE Consortium. DICE Enables Quality-Driven DevOps for Big Data. 2016. Available online: http://www.dice-h2020.eu/2016/04/25/dice-enables-quality-driven-devops-for-big-data-a-white-paper (accessed on 12 January 2022).
36. Eclipse Foundation. Eclipse QVT Operational Project. 2016. Available online: https://projects.eclipse.org/projects/modeling.mmt.qvt-oml (accessed on 12 January 2022).
37. Eclipse Foundation. Acceleo. 2012. Available online: https://www.eclipse.org/acceleo/ (accessed on 12 January 2022).
38. Red Hat. Hibernate ORM. 2021. Available online: https://hibernate.org/orm (accessed on 12 January 2022).
39. Apache foundation. The Apache Velocity Project. 2021. Available online: https://velocity.apache.org (accessed on 12 January 2022).
40. Eclipse Foundation. Eclipse Modeling Framework Project (EMF). 2008. Available online: http://www.eclipse.org/modeling/emf/?project=emf (accessed on 12 January 2022).
41. Project Management Institute. A Guide to the Project Management Body of Knowledge. 2021. Available online: https://www.pmi.org (accessed on 12 January 2022).
42. Axelos. PRojects IN Controlled Environments - PRINCE2. Available online: https://www.axelos.com/best-practice-solutions/prince2 (accessed on 12 January 2022).
43. Pop, D.P.; Altar, A. Designing an MVC model for rapid web application development. *Procedia Eng.* **2014**, *69*, 1172–1179. [CrossRef]