proceedings

MDPI

Overview of a Project devoted to Release an open-source Software Tool for the Creation, Feeding and Querying of a NoSQL Metadata Repository about UML Class Diagrams †

Gaetanino Paolone [1], Romolo Paesani [1], Martina Marinelli [1] and Paolino Di Felice [2,*] {0000-0003-3552-0199}

[1] Gruppo SI S.c.a.r.l., 64100 Teramo, Italy; g.paolone@softwareindustriale.it; r.paesani@softwareindustriale.it; m.marinelli@softwareindustriale.it;
[2] Department of Industrial and Information Engineering and Economics, University of L'Aquila, 67100 L'Aquila, Italy; paolino.difelice@univaq.it
* Correspondence: paolino.difelice@univaq.it; Tel.: (+39-320-423-2540)
† Presented at 2nd International Electronic Conference on Applied Sciences, 15–31 Oct. 2021 (on line).

Abstract: UML Class diagrams are a relevant artifact in the model driven development of today Web applications, since them describe the structure of the software system to be developed. Tool support is needed to manage these models and metadata about the models are a precondition to implement reuse strategies. While setting up metadata repositories, architects have to make important design decisions. This paper gives an overview about the objectives, beneficiaries, architecture and technologies of an ongoing software project. Its goal is to release an open-source software tool devoted to the creation, feeding and querying of a NoSQL metadata repository about UML class diagrams.

Keywords: UML; MDA; MVC; Spring framework; metadata; repository; class diagram; open-source

Citation: Paolone, G. et al.. *Proceedings* **2021**, *68*, x. https://doi.org/10.3390/xxxxx

Published: date

1. Introduction

Enterprise Web applications are an essential part of the computer-based information system of most organizations. In the last ten years, the authors carried out an intense research activity in the field of automatic code generation of this category of software applications [1-3]. The pillars on which is based our approach are UML and the Model Driven Architecture (MDA).

This paper describes the objectives, beneficiaries, architecture and technologies of an ongoing project, whose goal is to release an open-source software tool (called **`xMetaRep`**) devoted to the creation, feeding and querying of a NoSQL metadata repository about UML class diagrams. The latter are a relevant specification technique to describe the structure of the software system to be developed. The main findings of a recent review spanning from 2000 to 2019 showed that UML class diagrams were the most used artifacts in Software Engineering research [4].

The usage of **`xMetaRep`** is a precondition for speeding up the development process of high quality Enterprise Web applications through the reuse of UML artifacts. Very Small Entities (VSEs) developing such a category of software are the beneficiaries of the software tool. VSEs are the global software industry's dominant force since many offerings are coming from them. We adopt the following definition of VSEs taken from ISO/IEC TR 29110-1:2016(E): "A VSE is considered to be an entity that engages in systems or software engineering activities at any point, including development, integration, or maintenance."

The conceptual and technological choices underlying the ongoing project were driven by a careful investigation of the state of the art. The latter is not part of the paper. The merits of this short paper derive from the intrinsic value and relevance of the project itself, which implements information and suggestions coming from the current state of

the art of the following three research topics belonging to the domain of the development of Web applications when adopting the Model Driven Engineering:

- the growing interest in building UML model repositories (e.g., [5-6]);
- the relevant role that the reuse of UML models plays to accelerate the overall development process of Web applications and, at the same time, to ensure a high level of quality of the final code è [7];
- the urgency to support VSEs operating in the software domain by offering them easy-to-use open-source products [8].

The paper is structured as follows. Section 2 is about the materials and methods used in the project; while Section 3 describes the results and discusses them. Section 4 provides an overview of the future work.

2. Materials and Methods

The project is composed of three phases. The first two are already finished. They can be summarized as follows. We started from the study of the state of the art about repositories of UML artifacts (Phase 1). The final design decision was to organize the VSE's repository as three distinct components:

- a repository about the modeling artifacts, briefly Model Repository [9];
- a repository about the generated code; and
- a repository containing metadata about modeling artifacts result of past projects, briefly Metadata Repository [9].

Phase 2 focused on the formalization of the structure of the Metadata Repository and its implementation as a NoSQL database. NoSQL is an umbrella term for different technologies. The most prominent are known as: Key-value DBs, Document DBs, Column DBs, and Graph DBs. Key-value DBs are the simplest NoSQL DBs. They store a set of key-value pairs. Redis implements this model. Document DBs contain key-value pairs, which can be any sort of value, array, or even another document. MongoDB implements this model. Column DBs organize data into columns rather than rows, however they essentially operate in the same manner as tables do in relational DBs. Apache Cassandra implements this model. Graph DBs are for general-purpose use particularly with unstructured data and social networks. Neo4j is a popular system example.

To build the Metadata Repository about class diagrams the flexibility featured by Document DBs is fully satisfactory. Flexibility is necessary to make the DB schema independent of the internal organization of the classes in the class diagram. Specifically, flexibility is needed to model the name and datatypes of attributes and the name and I/O parameters of the operations. Both these features are highly variable moving from one class to another. To solve the issue, we took into account two alternative technologies: MongoDB and PostgreSQL (from version 11.0, the latter system fully supports the storage of documents in the `jsonb` format, as MongoDB does). Both DB systems are open-source. The use of free software is mandatory for SMEs to limit the costs. The final decision was to adopt PostgreSQL because (since version 11) it closed the gap that motivated the rise and development of NoSQL technologies; moreover, PostgreSQL provides capabilities that NoSQL technologies simply cannot, namely: a powerful query language, a sophisticated query optimizer, data normalization, joins, and referential integrity. The positive side effect of the availability of a powerful query optimizer is that PostgreSQL outperforms MongoDB in almost all cases, as it has been proved in many recent studies (e.g., [10-11]). For example, in [11] authors loaded a dataset of 200 million records of JSON documents in MongoDB (Community Server1, ver. 4) and in PostgreSQL (ver. 11) using the `jsonb` datatype. The goal of the experiment was to compare the performance of the two systems on four custom written queries over one year of GitHub archive data. PostgreSQL was found to be between 35-53% faster on three of the four queries, and 22% slower on the other one.

Several advantages comes from storing metadata about UML models within a company NoSQL repository. First, NoSQL databases overcome "pure" relational ones in term

of flexibility. Second, NoSQL databases guarantee a high level of interoperability. Third, a company database ensures cooperation of modelers and developers in the development of a system. Fourth, querying the metadata inside the company's repository helps reuse of the artifacts that best fit the requirements of new projects. Fifth, studying UML diagrams from previous high quality projects can help novice modelers to learn from the experience of senior ones. This latter motivation has been pointed out by Gosala et al. in [12].

As part of Phase 2 of the project, a parser was implemented which is responsible for the automatic extraction of metadata describing the artifacts in the Model Repository and their subsequent upload into the Metadata Repository.

At present we are implementing the User Interface towards the Metadata Repository (Phase 3).

3. Results and discussion

Figure 1 shows the architecture of **`xMetaRep`**:

- the VSE's Model Repository contains the XMI files about UML Class diagrams. These files are the input data to the overall process for building and feeding the Metadata Repository about such a category of artifacts;
- the User Interface is composed of three software components: **`xMR Creator`**, **`xMR Parser`** and **`xMR Query Builder`**. They support, in turn: (a) the creation of an empty instance of the NoSQL DB; (b) the extraction of metadata from the XMI files and copying them into the NoSQL DB; (c) the formulation of three predefined templates of queries against the NoSQL DB;
- the NoSQL DB layer denotes the Metadata Repository about UML Class diagrams in the VSE's Model Repository.

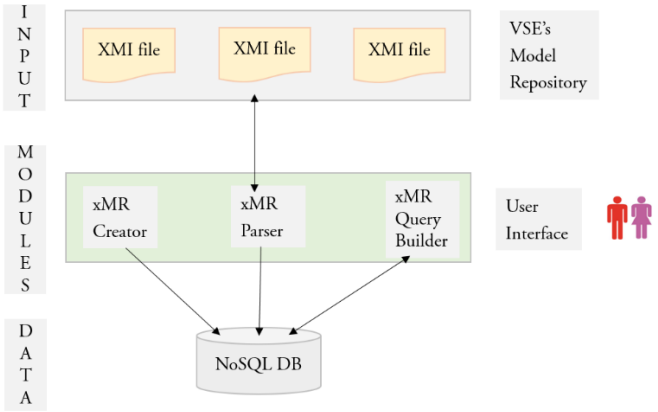

Figure 1. Architecture of **`xMetaRep`**.

The following sentences are about the components of **`xMetaRep`**, namely the Modules and the Data layer. For practical reasons, the presentation of the latter comes first.

Figure 2 shows the schema of the seven tables that structure the metadata repository underlying the **`xMetaRep`** system, their primary key (the red attribute), and the join paths among them (i.e., the "primary key – foreign key" pairs). The Primary Key constraint is the mechanism offered by the database technology for implementing OCL (Object Constraint Language) intra-UML-element constraints. The Referential Integrity constraint between the foreign key of the *slave* table and the primary key of the *master* table is the mechanism featured by the relational database technology for implementing OCL inter-UML-element constraints. The **`list`** attribute in tables **`operation`** and **`attribute`** are both defined as **`JSONB`** data type. This data type adds flexibility to the metadata repository since it makes the schema of the database independent of the internal organization of the classes inside the class diagrams.

The schema of the proposed Metadata Repository is derived from a UML metamodel.

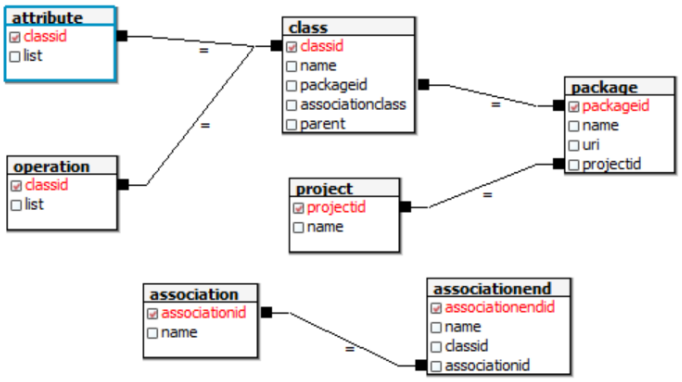

Figure 2. The tables of the NoSQL (PostgreSQL) database.

From the side of a VSE's modeler, **`xMR Creator`** looks like a GUI with the button "Do you want to create the Metadata Repository (Y/N)?". By selecting Y(es), an empty instance of the NoSQL DB is created.

`xMR Parser` takes as input the XMI file about a single Class diagram present in the Model Repository and generates a list of **`INSERT INTO`** NoSQL commands to be stored into the Metadata Repository. The values in those commands are the metadata which provide a summary description of the Class diagram.

The automatic step the parser is responsible for is fundamental for the success of the experiment of building and keeping updated a corporate metadata repository about UML class diagrams, since, as remarked for example in [13], one of the desirable properties of repositories is that their content does not depend on a single person or a small group of people. In the current version of the parser, the extraction of the content of the XMI file and its formatting as (NoSQL) records works for XMI files coming from `StarUML`. The extension of this component to other UML tools is part of our next work.

The parser has been implemented as a Java Web application which adheres to the Model-View-Controller (MVC) pattern. The Spring framework[1] was used for creating the application. Spring is the most adopted Java framework all over the world. Figure 3 shows the subset of the full stack of the Spring framework that was used in the development of the parser. In order to simplify things, beside Spring we used also Spring Boot. A 2021 public survey carried out by JRebel [14], in the Java development community, reported that 62% of the respondents were working with Spring Boot. Overall, the Web application consists of 20 classes, 15 interfaces and 2 views. The latter are 2 JSPs that implement the User Interface of the Web application. The first page concerns the loading of the XMI file from the modeler, while the second page shows the result of the parsing operation (either "Success" or "Error").

Figure 4 shows the processing flow of the developed Spring MVC Web application.

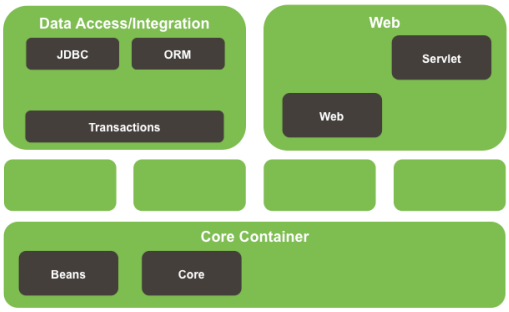

Figure 3. The Spring's modules used to develop the parser.

[1] https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/overview.html

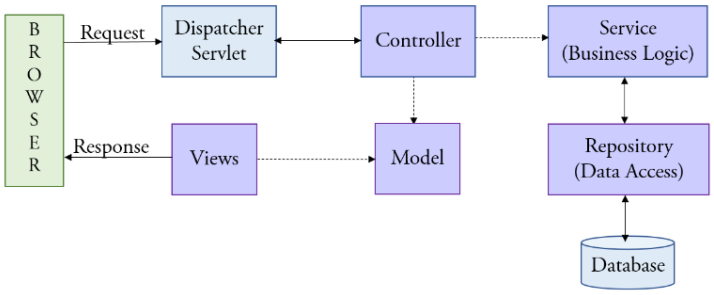

Figure 4. The flow of the parser Web application. Violet rectangles denote classes implemented by us, while the blue ones denote components provided by the Spring framework.

Table 1 and Table 2 list the technologies used to develop the **`xMR Parser`**.

Table 1. List of the used software technologies.

Layer	Adopted software technologies
User Interface	JSP, JavaScript, Bootstrap
Business Logic	Intellij Idea, Java, Spring, Spring Boot, Hibernate, JDOM Parser
Data	PostgreSQL

Table 2. List of the used software technologies.

Adjacent Layers	Technology
User Interface – Business Logic	JSP, Spring
Business Logic – DBMS	Spring, Spring Boot
DBMS – DB	Hibernate

The **`xMR Query Builder`** component will allows VSE's modelers to pose queries against the NoSQL DB. The output of the queries is a set of records in the usual tabular form. This component features three distinct templates which implement the Search interface Hamid talks about in [15]. The templates are briefly described hereafter.

Template 1 (*Simple search*) The VSE's modeler may invoke the displaying of the tuples of any of the database's tables.

Template 2 (*Run a Simple Query*) Two options are available. Both allow the VSE's modeler to call the running of a predefined query.

Option 1

The predefined query joins the tables: **`class`**, **`package`**, and **`project`** (Figure 2). The structure of the output records is <*class name*, *package name* (containing the class), *project name*>.

Option 2

Through this Web page it will be possible to run a query which joins the tables: **`class`**, **`attribute`**, **`operation`**, and **`package`**. The structure of the set of records returned is <*class name*, *list of attribute* (of the class), *list of operation* (of the class), *package name* (storing the UML Class diagram), *URI* (to the package)>.

Template 3 (Advanced Query Builder Page) This page creates a connection to the PostgreSQL server and, then, reaches the NoSQL DB. At that point, the VSE's modeler may write SQL queries against the metadata repository according to his needs. The more he is skilled in the usage of SQL, the more the queries may be powerful.

3. Conclusion and Future work

This paper provided an overview of a project devoted to release an open-source software tool (**`xMetaRep`**) for the creation, feeding and querying of a NoSQL metadata repository about UML class diagrams. VSEs developing Enterprise Web applications through the

reuse of UML artifacts are the beneficiaries of the software tool. The availability of a centralized company repository containing metadata about UML diagrams implemented as a NoSQL database represents the ultimate alternative to the today scenario where software engineers have to search manually these artifacts inside a huge number of files stored into different folders. Currently, we are working on the **`xMR Query Builder`** component.

Author Contributions: "Conceptualization, Di Felice and Paolone; methodology, Di Felice; software, Paesani; validation, Paolone and Paesani; formal analysis, Di Felice; data curation, Marinelli; writing Di Felice; funding acquisition, Paolone. All authors have read and agreed to the published version of the manuscript."

Funding: The research was funded by Software Industriale. https://www.softwareindustriale.it/en/gruppo-si-and-university.

Informed Consent Statement: Not applicable.

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

References

1. Paolone, G.; Di Felice, P.; Liguori, G.; Cestra, G.; Clementini, E. A Business Use Case Driven Methodology -- A Step Forward. In *Proceedings of the 5th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE), Athens, Greece, 22–24 July **2010**; pp. 221–226. Available online: https://pdfs.semanticscholar.org/ 70c0/819791d0f68bdf624c18c8a5a2f2a512e9f9. pdf (accessed on 7 July 2021).
2. Paolone, G.; Marinelli, M.; Paesani, R.; Di Felice, P. Automatic Code Generation of MVC Web Applications. *Computers*, vol. 9, no.3, 56 **2020**; https://doi.org/10.3390/computers9030056.
3. Paolone, G.; Paesani, R.; Marinelli, M.; Di Felice, P. Empirical Assessment of the Quality of MVC Web Applications Returned by xGenerator. *Computers* **2021**, 10(2), 20; https://doi.org/10.3390/computers10020020 - 04 Feb 2021.
4. Koç, H.; Erdoğan, A.M.; Barjakly, Y.; Peker, S. UML Diagrams in Software Engineering Research: A Systematic Literature Review. Proceedings 2021, 74, 13. https://doi.org/10.3390/proceedings2021074013
5. Di Rocco, J.; Di Ruscio, D.; Iovino, L.; Pierantonio, A. Collaborative repositories in model-driven engineering. IEEE Software, vol. 32, no. 3, **2015** 28–34.
6. Basciani, F.; Di Rocco, J.; Ruscio, D.D.; Iovino, L.; Pierantonio, A. Model Repositories: Will They Become Reality? In Proc. of the 3rd Inter. Workshop on Model-Driven Engineering on and for the Cloud, and 18th Inter. Conference on Model Driven Engineering Languages and Systems (MoDELS 2015), Ottawa, Canada, Sept. 29 **2015**.
7. Gomes, P.; Gandola, P.; Cordeiro, J. Helping Software Engineers Reusing UML Class Diagrams. 7th International Conference on Case-Based Reasoning, Belfast, Northern Ireland, UK, August 13-16, 2007, R.O. Weber and M.M. Richter (Eds.): ICCBR 2007, LNAI 4626, pp. 449–462, **2007**.
8. Ponsard, C.; Deprez, J.-C. Helping SMEs to Better Develop Software: Experience Report and Challenges Ahead. 2018 ACM/IEEE 40th International Conference on Software Engineering: Software Engineering in Practice, May 27-June 3, **2018**, Gothenburg, Sweden.
9. Mayr, C.; Zdun, U.; Dustdar, S. Reusable Architectural Decision Model for Model and Metadata Repositories. In: de Boer F.S., Bonsangue M.M., Madelaine E. (eds) *Formal Methods for Components and Objects*, **2008**. Lecture Notes in Computer Science, vol. 5751, 2009 1–20. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04167-9_1.
10. Makris, A.; Tserpes, K.; Spiliopoulos, G.; Zissis, D.; Anagnostopoulos, D. MongoDB Vs PostgreSQL: A comparative study on performance aspects. Geoinformatica, vol. 25, pp. 243–268, 2021.
11. Tortosa, A.H.; et al. Performance benchmark PostgreSQL / Mongodb. A white paper by Ongres. https://www.enterprisedb.com/white-papers (accessed on Oct.2021).
12. Gosala, B.; Chowdhuri, S.R.; Singh, J.; Gupta, M.; Mishra, A. Automatic Classification of UML Class Diagrams Using Deep Learning Technique: Convolutional Neural Network. Applied Sciences 11, **2021** 4267 https://doi.org/10.3390/app11094267.
13. Schlick, R.; et al.: A Proposal of an Example and Experiments Repository to Foster Industrial Adoption of Formal Methods. In: Margaria T., Steffen B. (eds) *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*. ISoLA **2018**. Lecture Notes in Computer Science, vol 11247. Springer, Cham. https://doi.org/10.1007/978-3-030-03427-6_20.
14. *2021 Java Developer Productivity Report*. A technical report by JRebel, Perforce Software, Inc. **2021** https://mma.prnewswire.com/media/1422901/2021_java_developer_productivity_report.pdf?p=pdf (accessed on July 2021).
15. Hamid, B. A. Model Repository Description Language – MRDL. In: Kapitsaki G., Santana de Almeida E. (eds) *Software Reuse: Bridging with Social-Awareness*. International Conference on Software Reuse, 2016. Lecture Notes in Computer Science, vol 9679, **2016** 350–367. Springer, Cham. https://doi.org/10.1007/978-3-319-35122-3_23.