XRepo 2.0: a Big Data Information System for Education in Prognostics and Health Management

Nestor Romero[1], Rafael Medrano[1], Kelly Garces[1], David Sanchez-Londono[2], and Giacomo Barbieri[2]

[1] *Department of Systems and Computing Engineering, Universidad de los Andes, Bogotá (Colombia)*
ns.romerob@uniandes.edu.co
re.medrano@uniandes.edu.co
kj.garces971@uniandes.edu.co

[2] *Department of Mechanical Engineering, Universidad de los Andes, Bogotá (Colombia)*
d.sanchezl@uniandes.edu.co
g.barbieri@uniandes.edu.co

ABSTRACT

Within Industry 4.0, Prognostics and Health Management (PHM) holds great potential due to its ability to bring deep insights into the current state of manufacturing equipment. When developing PHM competences in higher education, it is desirable to train students in the development and utilization of the algorithms commonly adopted for PHM analyses. Despite the widespread of PHM datasets, education in PHM is complicated by the unavailability of a platform that standardizes the data format into a unified metamodel. To cope with this, XRepo 2.0 is proposed: a big data information system that allows professors to share PHM sensor data in a standard format within an experimental and educational context. In this work, a metamodel is introduced to represent PHM datasets, and the Hadoop framework is integrated with a document database to enable the management of the large amount of data available today. MapReduce processing engine is utilized to enable teachers to pre-process the data on the cloud infrastructure, which is a crucial aspect for the assessment of the algorithms developed by the students. Finally, a prototype of XRepo 2.0 is deployed on the Azure Cloud and validated with respect to functionality and performance criteria. Given the importance of PHM within Industry 4.0, we expect that XRepo 2.0 will contribute to the unification and sharing of selected sensor data with the academic community for the development of competences in PHM.

1. INTRODUCTION

Current maintenance strategies have progressed from breakdown maintenance to preventive and then to prognostics and health management (Martin, 1994). *Breakdown Maintenance* is the earliest form of maintenance, where no actions are taken to maintain the equipment until it breaks and consequently needs a repair or replacement. In the 1950s, *Preventive Maintenance* strategies were introduced and require maintenance on a time (or usage) interval regardless of the health condition of the asset. Later, *Prognostics and Health Management* (PHM, also known as Predictive Maintenance) emerged and is defined as a condition-driven preventive maintenance program.

The research field of PHM has grown significantly due to the *Industry 4.0* movement and to the advancements in data acquisition, gathering, storing, and analytics (Lee, Ardakani, Yang, & Bagheri, 2015). *Prognostics and Health Management* holds the potential to estimate the current health state and predict the future states of machinery, granting important insights into decision-making processes (Lee, Lapira, Yang, & Kao, 2013). To realize this potential, it is necessary to store, process and analyze the large amount of data that is collected from the many different sensors available today in machinery through the use of the *big data* (Lee, Lapira, Bagheri, & Kao, 2013). Given the above, it becomes highly advantageous for engineering students to be trained in PHM.

Concerning *education* in traditional maintenance engineering (i.e. breakdown and preventive maintenance), different textbooks are already available; e.g. (Ebeling, 2004; Ben-Daya, Kumar, & Murthy, 2016; Moubray, 2001). However, few textbooks can be found on PHM given the novelty of the approach; e.g. (Mobley, 2002; Kim, An, & Choi, 2016). PHM textbooks explain general concepts such as the PHM principles and the different sensors that can be utilized for the PHM analyses. However, practical exercises are not presented limiting the development of competencies in PHM. Therefore,

https://doi.org/10.36001/IJPHM.2021.v12i1.1412

professors and researchers find themselves developing PHM case studies and sharing the sensed data. Examples of such shared data are the Case Western Reserve University Data for rolling bearings (Loparo, 2012), the Milling dataset from the BEST lab at UC Berkeley (Agogino & Goebel, 2007), and the Uniandes unbalanced shaft dataset (Barbieri, Sanchez-Londono, Cattaneo, Fumagalli, & Romero, 2020) amongst others. The available PHM datasets present data in different formats complicating their input into data processing engines; e.g. Matlab, Python, etc. Furthermore, these datasets contain heterogeneous context information without guaranteeing their inclusion of all the information necessary for the PHM analysis. A platform that standardizes the format of PHM datasets would be desirable.

Nowadays, a *big data information system* for sharing PHM-related datasets in education is not a available. This educational information system should enable professors and students to easily access sensor data from multiple sources, presenting their corresponding contextual metadata with a homogeneous format/metamodel. In fact, PHM data are shared with heterogeneous data formats and lack of metadata describing the particular experimental context (Ardila et al., 2020); e.g. the date of acquisition, and the health condition associated to the equipment, amongst others. Thus, the lack of a big data architecture – offering access to data with a standard format and contextual metadata – represents an issue for education in PHM, and would be desirable for the unification and sharing of the case studies currently available for developing competencies in PHM.

The design and implementation of such system – referred to as *XRepo 2.0* – is the focus of this work. Here, the authors propose a big data information system where professors and students are able to upload and access PHM sensor data, along with metadata that describes the experimental and educational contexts of the setting. In this way, users do not need to worry about the heterogeneity of sensor data and the handling of big data, and educators can solely focus on developing PHM competences in students through pedagogical activities.

According to the ISO-13374 normative, the following functionalities related to the data management and processing can be implemented within an information system for PHM:

- *Data Acquisition*: data acquired from sensors and devices are stored.
- *Data Manipulation*: mathematical transformations are applied to maintain the valuable parts of the collected data while removing their unwanted components; e.g. noise etc. Furthermore, features are extracted for the subsequent analyses.
- *State Detection*: the asset state is identified (i.e. healthy or faulty) and – in case of faults – these are detected, isolated and classified.
- *Health Assessment*: the actual health condition of the asset is quantified.
- *Prognostics Assessment*: the future health condition is predicted. This functionality also implies the calculation of the asset Remaining Useful Life (RUL).
- *Advisory Generation*: easy-to-use and effective visualization tools are developed to present the maintenance information and support the decision-making process.

A complete information system for PHM education should provide all the aforementioned functionalities. However, in this design iteration XRepo 2.0 only deals with the Data Acquisition one due to the complexity of the problem.

A number of *technical challenges* have been identified for the introduction of PHM strategies within the maintenance process of an enterprise (Galar & Kans, 2017). A subset of these challenges are relative to the acquisition and management of PHM data, and are also faced in the implementation of a big data information system for education. These are:

- *Data Integration*: data coming from different sensors must sometimes be analyzed. Information from multiple sources should be acquired and integrated.
- *Data Heterogeneity*: PHM data that has previously been made available for research and education tend to express contextual metadata using different syntax and semantics.
- *Data Search Usability*: data might be stored without a logical way to navigate. This results in datasets which cannot easily be comprehended and analyzed.
- *Data Volume*: large volumes of data are generated when a system is monitored through multiple sensors with high sampling rates over extended periods of time.

The architecture proposed within this work is built atop *XRepo 1.0* (eXperiments Repository). XRepo 1.0 is an information system that allows users to store and share PHM datasets in a standard format within an experimental context (Ardila et al., 2020). The first version of the XRepo information system focused on the first three aforementioned challenges. Whereas, XRepo 2.0 has the following main novelties with respect to the former version:

1. *Educational metamodel*: XRepo 1.0 was born as an information system for sharing PHM data, while XRepo 2.0 is targeted to education. Therefore, the XRepo 1.0 architecture has been modified and new contextual metadata has been added to its domain model with the objective to fit educational scenarios. The introduced educational metamodel has been developed through the review of available PHM datasets, and the elicitation of needs relative to education in PHM.

2. *Big data*: considering their potential to store, process and analyze large volumes of data (Lee, Lapira, Bagheri, & Kao, 2013), the present work aims to use big data technologies to address the 'Data Volume' challenge.
3. *Online processing*: considering that the information system is able to manage big data, XRepo 2.0 allows teachers to pre-process the data before sharing them with students. This functionality offers the possibility for professors to process large volumes of data online, without the need to locally download and manage them.

Given the above, the article is structured as follows: Section 2 presents a state-of-the-art analysis of related work. Section 3 summarizes the requirements and the design decisions that guided the development of XRepo 2.0. Section 4 shows the implementation of XRepo 2.0, and Section 5 describes an evaluation of the information system. Finally, Section 6 presents the conclusions and offers possible paths for future research.

2. RELATED WORK

Considering that the main novelties of this work are the introduction of an educational metamodel for PHM datasets and the generation of a big data information system for PHM datasets, the state-of-the-art is divided into the analysis of: (i) PHM datasets (Section 2.1); (ii) big data information systems (Section 2.2).

2.1. PHM Datasets

Multiple sources of PHM datasets are available in the literature and on the web. These sources can be classified as: web-based data-science environments, maintenance-focused big data platforms, and individual datasets available online for download.

Concerning *web-based data-science environments*, Kaggle[1] is the most famous platform that enables users to find and publish datasets. Kaggle has been adopted by different companies to upload their datasets with the purpose of challenging the Kaggle community to develop effective machine learning algorithms that can properly classify their data (Garcia Martinez & Walton, 2014). Given that the datasets uploaded to Kaggle come from different domains, a single data format and contextual metadata is not utilized. Therefore, Kaggle and web-based data-science environments in general do not address the data heterogeneity challenge tackled into XRepo 2.0.

Attempts to create *maintenance-focused big data platforms* can be found as the six step framework for data-driven maintenance (O'Donovan, Leahy, Bruton, & O'Sullivan, 2015), and the big data platform for maintenance data with data analysis capabilities (Yu, Dillon, Mostafa, Rahayu, & Liu, 2020). Even if these platforms enable the sharing of PHM datasets, a metamodel to standardize the format and the contextual information of the data is not utilized.

Finally, *individual datasets* can also be found online available for download. These datasets present heterogeneity both in the format and in the contextual information of the data. Next, few of the available PHM datasets are illustrated.

The content of PHM datasets for educational purposes should be studied from two perspectives: experimental and educational. Regarding the *experimental context*, different datasets are available in the internet. Examples of them are relative to: i) bearing failures (from NASA (Lee, Qiu, Yu, Lin, & Services, 2007), the Case Western Reserve University (Loparo, 2012), and PRONOSTIA (Nectoux et al., 2012)); ii) cutting blade degradation from OCME (Von Birgelen, Buratti, Mager, & Niggemann, 2018); iii) milling machines from BEST lab (Agogino & Goebel, 2007); and iv) unbalanced shaft from Uniandes (Barbieri et al., 2020). By analyzing and comparing these datasets, we figured out the following contextual metadata: system operative ranges, system operative condition, start and end date time of each acquisition, utilized sensors, measured variables, and sensor sampling frequency. However, none of the aforementioned datasets presents all this information in a unified manner (Ardila et al., 2020). In our opinion, this is determined by the lack of an information system for unifying and sharing the available PHM datasets. The existence of such a platform would encourage users to define this information – rather than omitting important contextual metadata. In (Ardila et al., 2020), XRepo information system was proposed for standardizing the contextual metadata of PHM datasets.

Less information is available when looking for contextual metadata from an *educational perspective*. Online guides and tutorials can be found concerning PHM, such as the Azure 'AI Guide for Predictive Maintenance' (Azure, 2020). However, there is a lack of research concerning platforms, architectures or information systems for education in PHM. One attempt at creating an architecture for PHM education might be found in (Kans, Campos, & Håkansson, 2020). Here, authors describe a PHM framework for a remote laboratory. This framework uses the MIMOSA standard for defining both the contextual metadata and the functional layers of the framework. However, their implementation is limited to a physical test-bench, while a platform to upload data and run different algorithms is listed as future work. Finally, no education-specific considerations are made regarding either: a) the MIMOSA-based contextual metadata they use; b) the algorithms that they wish to implement to process the data.

Given the lack of an educational metamodel for sharing PHM datasets, this work will introduce an educational contextual metadata for PHM datasets.

[1] https://www.kaggle.com/

2.2. Big Data Information Systems

According to (Lee, Lapira, Bagheri, & Kao, 2013), a manufacturing information system is characterized by *5C functions*: Connection (sensor and networks), Content (correlation and meaning), Cloud and big data (data on demand and anytime), Community (sharing and social), and Customization (personalization and value). Different technologies and architectures are next illustrated for the presented 5C functions, showing the lack of an information system for sharing PHM datasets in an educational environment.

Connection A wide variety of communication protocols are currently available to enable machinery to communicate among themselves and with a central server. Some of these open-source protocols are MTConnect[2], the OPC Unified Architecture (OPC-UA)[3], the Constrained Application Protocol (CoAP)[4] and MQTT[5]. However, in this design iteration of XRepo, data will be uploaded as batch files and the integration of streaming processing protocols will be investigated in future works.

Content and customization As stated previously, the contents of PHM datasets in an education-focused information system should include both experimental and educational context metadata. The big data PHM architectures were found to either be too broad in scope without proposing a given data metamodel; i.e. web-based data-science environments and maintenance-focused big data platforms. Finally, individual datasets do not use a standard metamodel at all. The development of a metamodel that contains experimental and education metadata, and enables for the description of data from various PHM sources is then necessary.

Cloud and big data Nowadays, different commercial solutions are available for processing big data. For instance, RapidMiner (Kotu & Deshpande, 2014) provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analytics. In the context of PHM, Watchdog Agent (Djurdjanovic, Lee, & Ni, 2003) from the Center for Intelligent Maintenance Systems (IMS) enables PHM analytics by providing algorithms for signal processing and feature extraction, health assessment, performance prediction, and fault diagnosis. However, these platforms allow the data processing but they do not provide a data model – including contextual metadata – that can be straightforwardly exploited for education in PHM.

To integrate the processing functionalities with data representation, customized platforms have been proposed and are built atop existing open-source technologies from the big data ecosystem (Wan et al., 2017). For instance, (Wang, Fan, Huang, & Li, 2019) develop an architecture for the aviation manufacturing made of Apache Kafka[6], Apache Storm[7], Apache HBase[8], and Hadoop Distributed File System (HDFS)[9]. (Canizo, Onieva, Conde, Charramendieta, & Trujillo, 2017) integrate Apache Spark[10], Apache Kafka, Apache Mesos[11] and HDFS to predict failures on wind turbines using a data-driven solution deployed in the cloud. Even if the presented platforms are able to process and store PHM data, they present contextual metadata that are not targeted to education.

Community When developing the first version of XRepo, different datasets were studied (Ardila et al., 2020). These datasets are available in the internet and can be downloaded without any restrictions. This functionality is ideal within a research-oriented perspective. However, functionalities such as different access permissions for different users (e.g. instructors, students, etc.) become relevant within an educational setting. For instance, an instructor might want to hide certain contextual metadata to students for evaluation purposes; e.g. the health condition of an asset, etc. These requirements can not be fulfilled if datasets are freely and fully available for downloading. Instead, our platform has the added value of customizable download permissions.

Discussion The presented state of the art demonstrates the lack of a big data information system that unifies and shares the case studies currently available for developing competencies in PHM. Datasets are currently shared in the internet without standard formats, contextual metadata and permissions. Few information systems have been proposed by integrating technologies from the big data ecosystem, but these are not targeted to education. In this work, we take inspiration from the technologies utilized within these works to develop a big data information system for education in PHM.

3. XREPO REDESIGN PROCESS

In this section, the process that guided the development of the architecture of XRepo 2.0 is summarized.

The definition of an architecture can be divided into three steps (Li, Verhagen, & Curran, 2020): requirements, framework, and architecture. The *requirements* step (Section 3.1)

[2] www.mtconnect.org
[3] opcfoundation.org/about/opc-technologies/opc-ua
[4] tools.ietf.org/html/rfc7252
[5] mqtt.org
[6] kafka.apache.org
[7] storm.apache.org
[8] hbase.apache.org
[9] hadoop.apache.org
[10] spark.apache.org
[11] mesos.apache.org

involves the identification of the needs that the architecture must fulfill. The *framework* step (Section 3.2) requires the developer to define a set of functional layers able to fulfill the identified requirements. Finally, the *architecture* step (Sections 3.3-3.4) involves assigning a specific technological solution to the previously selected functional layers.

3.1. Requirements

The requirements identified to boost the potential of XRepo 2.0 in education are depicted in this section. The main features introduced within *XRepo 1.0* (Ardila et al., 2020) are maintained, since: (i) is able to share PHM data in a standard format; (ii) utilizes a data experimental context built by comparing PHM datasets currently available in the internet; (iii) fulfills the data integration, heterogeneity, and search usability challenges defined in section 1. Since large amounts of data are nowadays collected from the many different sensors available in machinery, the *'Data Volume'* challenge is also expected to be fulfilled with XRepo 2.0.

Given that our research targets *education*, new functional requirements have appeared. In PHM education, it is fundamental to provide students with labeled and unlabeled data (Barbieri et al., 2020). Here, labeled data refers to sensor data associated with the health condition of the equipment, whereas unlabeled data refers to sensor data without the equipment condition. While students use labeled data for the generation of PHM classification models, unlabeled data are utilized for evaluating the accuracy of the developed model. The instructor knows the health state of the unlabeled data and uses this information for the assessment. This means that instructors must be able to pre-process the PHM data for generating different subsets that are labeled or unlabeled. With XRepo 1.0, instructors had to download the data to their local (often resource-limited) environment and pre-process them to obtain the subsets. This task might overload instructors' workstations and is time-consuming. Furthermore, it should be possible to categorize datasets by taking into account different aspects, such as: i) the PHM competences that instructors desire to develop in students (Li et al., 2020) (e.g. diagnostics, prognostics, etc.); ii) the role of the data within the pedagogical activity (Barbieri et al., 2020) (either labeled or unlabeled data); iii) the analyzed failure and mechanical system. These education-oriented considerations brought the need to complement the XRepo 1.0 domain model with an educational context, as the original model was mainly devoted to the experimental part of the data; see Section 4.1 for further details. Finally, given its educational purpose, we consider that the new version of XRepo should be built atop open-source / free technologies and must allow concurrent connection of final users.

Based on the aforementioned target situation, we elicitated a list of requirements for XRepo 2.0 by integrating the challenges shown in section 1 with the 5C functions illustrated in section 2. These requirements are:

1. *Data integration, heterogeneity, and search usability*: the platform must continue to fulfill the challenges achieved with the first XRepo version, and represent the data types commonly used in PHM analyses; i.e. single value and timeseries data (Jardine, Lin, & Banjevic, 2006).
2. *Data Volume*: the platform must fulfill the data volume challenge – not previously addressed in XRepo 1.0 – by implementing big data technologies for the storage, processing and analysis of data.
3. *Connection*: data must be uploaded and downloaded as batch files with an established format.
4. *Content*: data must be represented with two contextual metadata:
   (a) *Experimental context*: must implement the experimental context defined in XRepo 1.0.
   (b) *Educational context*: data must be categorized taking into account different aspects, such as the PHM competences that will develop on the student, the role of the data within the pedagogical activity, and the analyzed failure and mechanical system, amongst others.
5. *Cloud*: instructors must be able to pre-process the data for generating different subsets, such as labeled and unlabeled subsets.
6. *Community*:
   - *Number of users*: a number of 1000 concurrent users was established. This value takes into account the number of potential users present in our university and academic partners.
   - *Platform access*: the information system must allow the assignment of roles to users for them to access to the different functionalities. For instance, professors can modify data, while students can only download them.
7. *Customization*: apart from the categorization of the data using different aspects, instructors must be able to extend the default categories by introducing additional custom fields.
8. *Open-source / free software*: given its educational purpose, XRepo 2.0 must be built atop open-source / free software.

3.2. Framework

With the objective to define a framework able to fulfill the identified requirements, selected Industry 4.0 frameworks have been analyzed. Given the large number of frameworks available in the literature (Kritzinger, Karner, Traar, Henjes, & Sihn, 2018), only frameworks then deployed into architectures have been considered. In particular, the following

frameworks were analyzed: Sezer et al.'s low cost approach to PHM (Sezer, Romero, Guedea, Macchi, & Emmanouilidis, 2018), Redelinghuys et al.'s six layer architecture for Digital Twins (Redelinghuys, Basson, & Kruger, 2019), Schroeder et al.'s AutomationML-based architecture (Schroeder, Steinmetz, Pereira, & Espindola, 2016), and Kans' architecture for a remote condition monitoring lab (Kans et al., 2020). After analyzing these frameworks, we noticed that all presented the functional layers shown in Figure 1. Therefore, these four functional layers were selected for the XRepo 2.0 framework which consists of:

- *Devices and Sensors*: involves data acquisition from physical sensors, which can potentially come from different vendors. This means that data coming from this layer may exist in heterogeneous formats.
- *Integration and Translation*: comprises the integration of data coming from multiple sources and with different formats, and their translation into a single data format.
- *Storage and Processing*: concerns storing and organizing high volumes of data in the cloud by following big data paradigms.
- *Application*: the front-end that either users or other software platforms use to interact with the stored data.

Within this work, only the 'Storage and Processing' and 'Application' layers have been implemented. Whereas, the 'Devices and Sensors' and 'Integration and Translation' are left as future work.

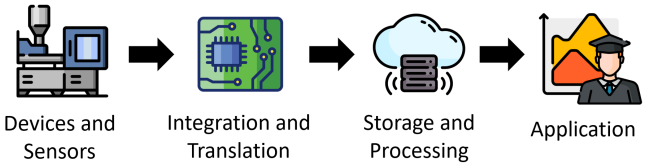

Figure 1. Selected functional layers for the XRepo 2.0 framework

3.3. Architecture: Design decisions

The design decisions taken for implementing the functional layers tackled within this work are next illustrated.

3.3.1. Integration and Translation

Even if the 'Integration and Translation' layer is not implemented in this work, the format of the data outputs from this layer must be specified for the design of the 'Storage and Processing' and 'Application' layers.

Three standard formats were analyzed for this purpose: *OGC-O&M* (Open Geospatial Consortium, 2020), *SHDR* used by the MTConnect platform (MTConnect Institute, 2019), and *MIMOSA's OSA-CBM* (Lebold & Byington, 2002). When sending single value and waveform data, the three data formats were found to represent this information in a 'time, value' form. However, these values are written as verbose XML files, meaning that data transmission channels are unnecessarily overloaded and a significant portion of storage is wasted. As such, storing data as simple *[time, value] vectors* was found to be sufficient.

Other than these [time, value] pairs, the data must be linked to their context information. On the one hand, the indication of the whole context information – along with each [time, value] pair – would make the format too verbose. On the other hand, some context information must be specified to differentiate data from different sensors. As such, the following decisions were taken:

- *ID*: each uploaded dataset must contain an ID, either on its first line or as a parameter during the upload operation. The ID is utilized to link the uploaded dataset to an experimental and educational context, previously defined within the XRepo information system.
- *Sensor and variable information*: each [time, value] pair must contain a tag indicating the sensor utilized for the acquisition and the sensed variable. This information will be utilized to filter the data based on the utilized sensor and/or acquired variable.

An example of data following the defined specifications is shown in Figure 2. Here, 'accel' is used as sensor tag, while 'x_accel' and 'y_accel' respectively indicate two different sensed variables. It must be noticed that data are downloaded from XRepo 2.0 with the same format shown in Figure 2. Therefore, this format constitutes the standard data format utilized to interface with XRepo 2.0.

```
ID,5f447a2e2f24295354f25e88,,
2020-01-20T14:30:38.000,accel,x_accel,-0.35429726
2020-01-20T14:30:38.000,accel,y_accel,-3.3519716
2020-01-20T14:30:38.001,accel,x_accel,4.5130434
2020-01-20T14:30:38.001,accel,y_accel,-1.2245702
2020-01-20T14:30:38.002,accel,x_accel,-15.471789
2020-01-20T14:30:38.002,accel,y_accel,-4.6735085
```

Figure 2. Example of the data syntax interpretable from the XRepo 2.0 'Storage and Processing' functional layer

3.3.2. Storage and Processing

The big data ecosystem has a multitude of architectural options and technologies, each of them with its unique features. (Sahal, Breslin, & Ali, 2020) review the strengths and weaknesses of existing open-source big data technologies and propose a methodology for selecting the most appropriate ones based on the studied scenario. In this work, Sahal et al.'s methodology was utilized to define the XRepo 2.0 architecture. In the methodology, the big data ecosystem is divided into four technical areas: Queue management systems,

Processing platforms, Storage, and Streaming SQL engines. Given the scope of this work and the queue management required by the aforementioned 'Integration and Translation' layer, only processing and storage concerns were analyzed. Thus, streaming SQL engines are not discussed. Design decisions were taken by mapping the needs of XRepo 2.0 to a set of predefined high level functional requirements proposed by (Sahal et al., 2020). Next, the design decisions taken for the storage and processing functionalities are illustrated.

Storage Three types of *storage models* are available for the big data ecosystem (Sahal et al., 2020): File system, Document-based and Column-based. Since XRepo 2.0 requires the storage and manipulation of plain text, the 'File system' storage model is selected. Moreover, the 'Document-based' storage model is also utilized since in future XRepo 2.0 will be integrated with transmission protocols. In summary, both 'File System' and 'Document-based' are selected as storage models.

Concerning the *data schema*, three types are available (Sahal et al., 2020): Structured, Semi-structured, and Unstructured. A Semi-structured data schema is selected since XRepo 2.0 receives and stores data files with a defined format.

Processing Three types of *processing models* are available (Sahal et al., 2020): batch, streaming, and hybrid. In XRepo 2.0, all the processing operations are launched by the users from a graphical interface by selecting the datasets to be processed. The described functionality requires a batch processing model.

3.3.3. Application

In XRepo 2.0, the application layer is implemented as a Web system and the main principle to guide its design was selected as usability. This objective will be achieved by defining elements with 'self-explaining' headers and contents. Moreover, background tasks will be implemented to keep the platform responsive to the user, even while data are being processed. Any action that a user performs on the data (e.g. uploading, searching, executing an algorithm, etc.) will be queued and performed on the background.

3.4. Architecture: Implementation Technologies

The last step of Sahal et al.'s methodology (Sahal et al., 2020) is to compare functional requirements against the features and capabilities of the different big data implementation technologies. Based on this analysis, Hadoop was selected as big data storage and processing platform since matches all the relevant aforementioned design decisions, and has a comprehensive documentation and active community of users and developers. In addition, Hadoop provides a MapReduce processing engine that will be used from teachers to pre-process the data before sharing them with students. To complement the big data storage functionality with a Document-based storage model, MongoDB[12] was selected to store metadata related to the sensor data hosted in Hadoop. The use of this document database will enable to filter sample files based on defined criteria. To implement this filtering functionality, a MapReduce search will be triggered on the HDFS repository.

4. XREPO 2.0: BIG DATA

XRepo stands for *eXperiments REPOsitory* and is a platform to collect, standardize and store experimental data (Ardila et al., 2020). This work extends the functionality of XRepo 1.0 in two aspects: i) storage of high volumes of structured and unstructured experimental data; ii) execution of MapReduce algorithms to organize said data for teaching purposes; iii) integration of an educational context to the already implemented experimental one. To this end, the domain model initially proposed for XRepo 1.0 was extended. Decisions were taken with respect to storage and processing concerns, and implemented into a new prototype. This led to the architecture presented in Figure 3. In the figure, components are grouped in the following functional layers: i) *Web User Interface (UI)*: groups all the front-end and web components; ii) *Security Layer*: handles all the authentication and access to services and information; iii) *Logic Layer*: groups all the service endpoints and the business logic components; iv) *Big Data Repository*: provides the back-end to store and access the unstructured data; v) *Data adapters*: handle transformations from external data formats to the XRepo data model. It is worth noting that the logic layer accesses the semi-structured data stored in the database.

The components already implemented in XRepo 1.0 are shown in Figure 3 with light gray and white boxes, and are:

- *Web UI*: user can utilize the functionalities of the platform through this interface. One of these functionalities is the upload of sample files generated by the Experiment Data Adapters.
- *Experiment Data Adapters*: take data from third party systems and convert them into the XRepo data model.
- *Target System and Experiment Managers*: create, retrieve, update or delete users, target systems, and experiments.
- *Sampling Search*: search for data that fulfills certain criteria.
- *Sample file load/search*: upload and find data as batch files.

Since additional components were developed (dark grey boxes in Fig. 3), the already existing components were

[12] www.mongodb.com

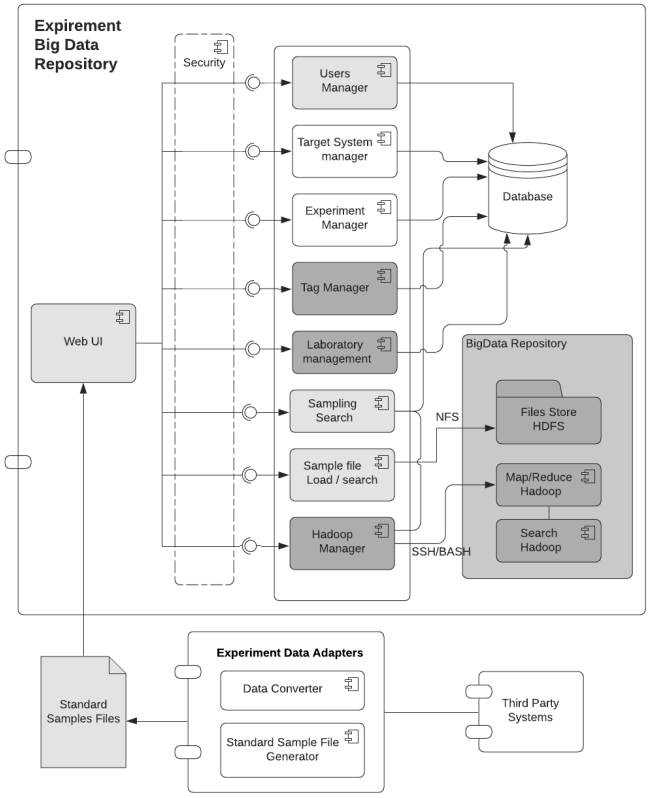

Figure 3. XRepo 2.0 architecture: diagram of the software components

adapted to be integrated with the new ones. It is worth noting that the light grey components required more adjustments than the white ones. The four new components incorporated into the architecture are: Big Data Repository, Hadoop Manager, Laboratory Management, and Tag Manager. All these components are accessed by the user through the Web UI, and are protected using profiles and roles.

The aforementioned new components are detailed in the remaining of this section which is structured as follows: section 4.1 shows the developed domain model which supports both the experimental and educational context of the data, and the big data functionality. Section 4.2 explains the functionality of the big data repository. Section 4.3 details how the big data processing was implemented using MapReduce functions. Finally, Section 4.4 depicts the management components referred to as Hadoop Manager, Laboratory Management, and Tag Manager; see Figure 3.

4.1. Domain Model

The XRepo 1.0 domain model (Ardila et al., 2020) has been extended to represent the educational context and support the big data functionality. This results in a new domain model shown in Figure 4. Due to the introduction of the big data storage, the 'Sample' data is directly stored in the HDFS, while all the other components of the model are stored in MongoDB. The figure illustrates in dark gray color new incorporated elements, in light gray color deeply modified elements, and in white color elements which were slightly modified or not modified at all.

The elements reused from the previous version of XRepo are:

- *Organization*: the main hierarchical classification of the data. This element indicates the owner of the data; e.g. university, research center, etc.
- *TargetSystem*: a physical system with sensors that can be monitored. This includes an 'operative range' that identifies the conditions under which the system can operate.
- *Experiment*: a technical sheet for a window of observation under specific operative conditions.
- *Sampling*: a set of data taken from the 'TargetSystem' in a given experiment. If the experiment includes data collected in different conditions, each of them will belong to a different 'Sampling'. An 'OperativeCondition' is assigned to each 'Sampling', along with the 'Device' used for the data acquisition, and the utilized 'Sensor' specified with the unit of the acquired variable.
- *Sample*: a single data value; i.e. [time, value] pair. A 'Sampling' consists of many samples.

Regarding the *samplings*, the main change implemented in this XRepo design iteration is the representation of their file locations. These are stored in the model as lists of properties, where each property points to a file URL on the HDFS system. The lists are owned by the 'Sampling' and 'Subset' concepts as indicated in Figure 4.

Next, the elements added to the original domain model for integrating the educational context of the data are illustrated:

- *Laboratory*: the big data functionality centers around this concept. It groups the data that teachers want to share with students. The data can be organized into two subsets: labeled and unlabeled. Each subset is generated using a MapReduce algorithm.
- *Algorithm*: this concept stores the mapper and reducer scripts that the system will use to run MapReduce over the samplings. Each algorithm is named as Labeled or Unlabeled indicating the type of subset that it will produce. A Laboratory can be associated to maximum two algorithms, i.e., a labeled and/or an unlabeled algorithm.
- *SubSet*: represents the result of running a MapReduce algorithm over the samplings. The output data is stored in a file on HDFS and is accessed by the students through a shared link. These subsets are named as Labeled or Unlabeled depending on the algorithm utilized for their generation.

- *Failure Mode and Analysis purpose tag libraries*: these are predefined tags associated to common scenarios found on PHM analysis. These tags can be extended by the administrator. The 'Failure Mode' tags represent the standard classification of failure types that a sampling can represent. For instance, the following tags can be used to label a rolling bearing sampling (Cerrada et al., 2018): outer raceway failure, inner raceway failure, or healthy bearing. In turn, 'Analysis Purpose' Tags are associated to laboratories and depicts the educational purpose of a given laboratory. According to (Li et al., 2020), these purposes are: fault diagnosis, prognostic assessment and health management.

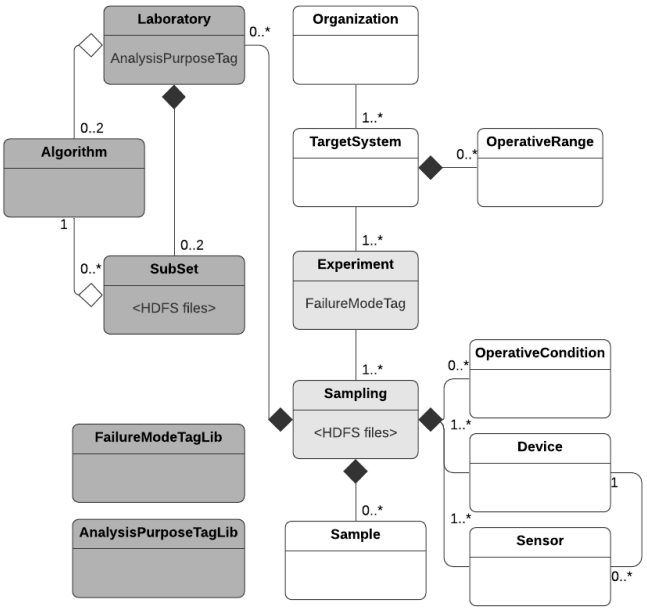

Figure 4. Domain model of XRepo 2.0: integration of the experimental (right-hand side of the figure) and educational context (left-hand side of the figure)

4.2. Big Data Repository

Experimental data are stored on a plain text format on HDFS. When the user uploads a file via the Web UI, the system first validates if the file satisfies the format, and then sends them to the HDFS using the Hadoop Network File System (NFS) gateway. This gateway allows to access to the HDFS from remote servers using the NFS protocol. All the HDFS files are organized using a folder hierarchy. The MongoDB database is used to keep track of the folder structure and the location of the files. XRepo provides access to the HDFS files by listing them in the Web UI. In this way, the end user can directly download the files through the Web UI, without the need to interact with the HDFS.

4.3. Big Data Processing

MapReduce is the engine used to process the data stored on the HDFS. MapReduce is native to Hadoop and provides a powerful paradigm to analyze a vast amount of information (Condie et al., 2010). In this design iteration, XRepo 2.0 is deployed on a single node pseudo-distributed Hadoop configuration. However, this can be scaled up to a multi-node distributed configuration to support larger datasets. Two components execute two types of MapReduce tasks provided by XRepo 2.0. Their execution is controlled by the Hadoop Manager, which is described in section 4.4. These two components are:

- *Search Hadoop*: samplings can be filtered by target system, tags and operative range. User can also select a date range for filtering the samplings. As the date information is stored on the HDFS files, the system launches a default MapReduce task to select the samples within the selected date range.
- *MapReduce Hadoop*: users can execute custom MapReduce algorithms to generate subsets of the data. Prior to this operation, users need to copy and paste the algorithms – developed and tested in their local development environment – to the Web UI, and send them to the system storage for the subsequent execution. When a given algorithm is associated to a laboratory, it can be execute over the associated samplings to produce either a labeled or unlabeled subset.

4.4. Management Components

The following components enable the system to organize and keep track of the execution of MapReduce tasks:

- *Hadoop Manager*: controls and monitors the execution of the aforementioned MapReduce algorithms by using SSH[13] remote commands. This action is launched from the Web UI and reaches the Hadoop server. The execution status is displayed on the Web UI through a progress bar.
- *Laboratory Management*: this component controls the Create, Read, Update and Delete (CRUD) operations associated to laboratories, algorithms and subsets.
- *Tag Manager*: allows to classify the data by using the 'Failure Mode' and 'Analysis Purpose' tags.

4.5. Prototype

The proposed architecture has been implemented through a prototype. The prototype is deployed on the Azure Cloud[14] by using three virtual machines respectively hosting the Web Application server, the Big Data server, and the Database server; see Figure 5. The computation resources (i.e., RAM,

[13] www.ssh.com/ssh
[14] azure.microsoft.com

CPU and disk size) assigned to each machine are shown in table 1. The Web UI is built on top of JHipster[15] which is a framework that integrates Java Spring Boot back-end services[16], Angular web front-end[17], and Gradle Build system[18]. Hadoop 3.1.6 is used as the Big Data storage and processing system. Python 3.0 is chosen as scripting language to write and execute MapReduce algorithms. Finally, MongoDB is utilized as database.

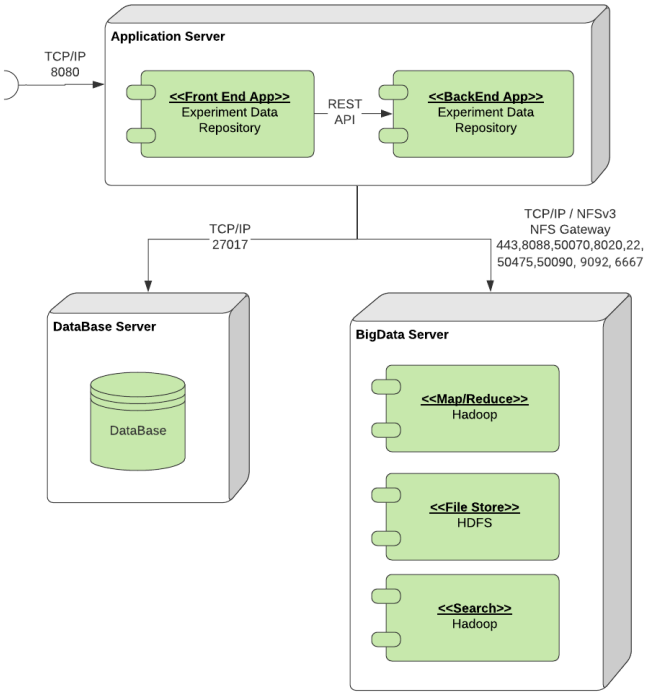

Figure 5. Deployment diagram of XRepo 2.0 on the Azure Cloud

Table 1. Resources of the deployed architecture nodes

Node	VM Type	RAM	vCPUs	Disk
BigData, Application Server	D2 v2	7 GB	2	SSD
DataBase	D3 v2	14 GB	4	SSD

Next, the Graphical User Interface (GUI) is illustrated for the main introduced functionalities:

- *Algorithm*: user can create, update and delete algorithms, as well as viewing the full list of algorithms currently available on the application. Algorithms uploaded by other users can also be displayed. Figure 6 illustrates how XRepo 2.0 display a list view of the introduced algorithms. The edition option (dark blue button) allows the user to associate laboratories to the algorithm. The green button launches the execution of the algorithm over the sampling data of the associated laboratory. It can be noticed that XRepo adds a scaffolding that helps users to get rid of the Hadoop technology complexity.

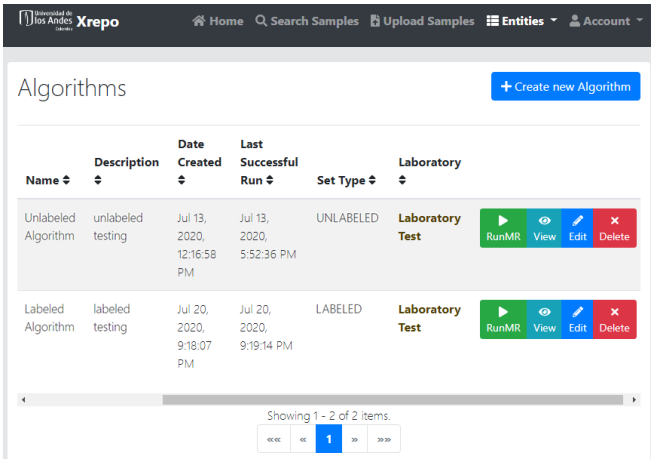

Figure 6. XRepo 2.0 GUI: algorithms list

- *Laboratory*: allows the user to create and edit laboratory entities; see Figure 7. Teachers can share to students labeled and unlabeled subsets generated by the execution of the MapReduce algorithms. The subsets will be available to download until a date established by the professor.
- *MapReduce reports*: allow the users to monitor the progress and status of MapReduce tasks currently running on the system; see Figure 8. It is worth noting that these tasks are associated to the executions of MapReduce algorithms.
- *Shared subsets*: once a subset is generated by an algorithm associated to a laboratory sampling, students can access the labeled/unlabeled subsets and download them directly from the user interface; see Figure 9.

5. VALIDATION

XRepo 2.0 prototype was validated from two fronts: functionality (Section 5.1), and performance intended as load capability (Section 5.2). Then, the XRepo 2.0 verification of the requirements identified in Section 3.1 is illustrated in Section 5.3. Finally, the threads to the validity of the process are discussed in Section 5.4.

5.1. Functionality Tests

The functionality tests focused on validating the base functionality of XRepo 2.0, along with checking the integrity of the data uploaded to the information system.

[15] www.jhipster.tech
[16] spring.io
[17] angular.io
[18] gradle.org

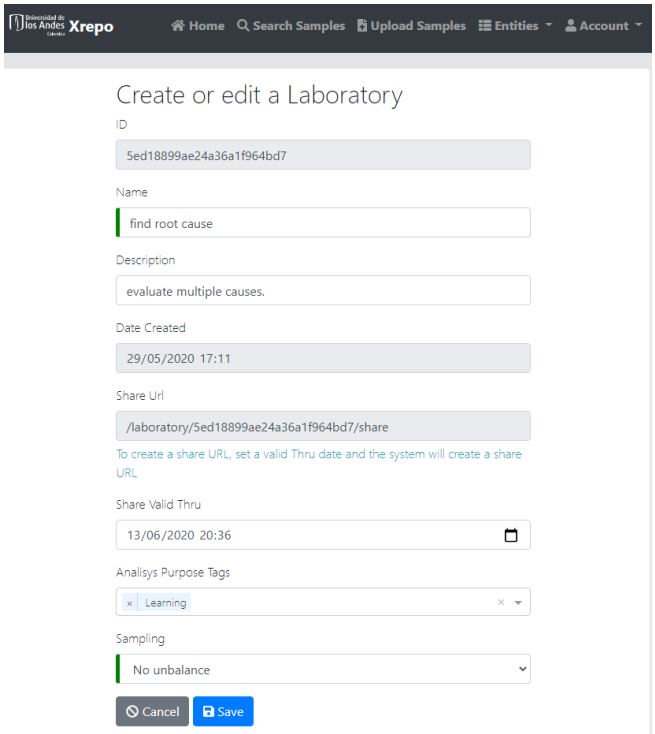

Figure 7. XRepo 2.0 GUI: Laboratory Edit functionality

5.1.1. Objectives

- Verify data upload and management functionalities.
- Verify data integrity.

5.1.2. Methodology

This validation was performed using vibration data for classifying different unbalanced levels of a mechanical transmission actuated with an induction motor; see (Barbieri et al., 2020). A translator was created to convert data from the original case study to the standard XRepo format. After having uploaded the data to the platform, the validation of the management functionalities consisted in the following steps:

1. Manually fill the metadata associated to the samples to be uploaded; i.e. Organization, TargetSystem, Experiment and Sampling.
2. Upload of the samples to XRepo 2.0.
3. Filter the samples by date range to obtain a subset of the data in XRepo 2.0.
4. Introduce a MapReduce algorithm to XRepo 2.0 for creating a labeled subset from the samples.
5. Execute the introduced MapReduce algorithm to generate the subset.
6. Visually explore the results of the MapReduce algorithm to validate the integrity of the obtained subset.
7. As a teacher, share a subset with the students.

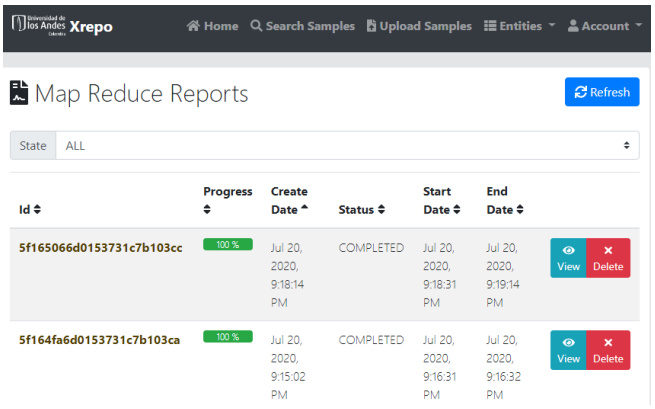

Figure 8. XRepo 2.0 GUI: MapReduce report view

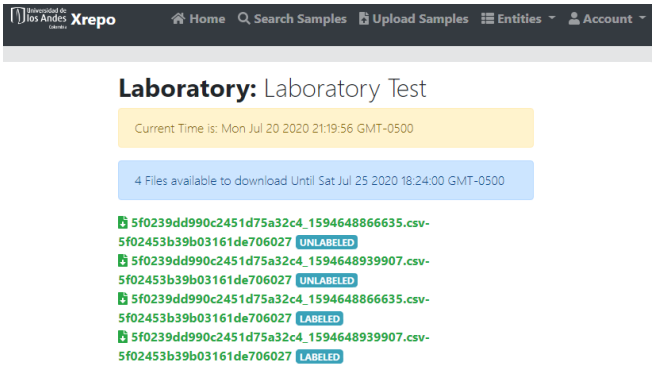

Figure 9. XRepo 2.0 GUI: Shared subsets

8. As a student, download the shared data.

Concerning step 4, the introduction of a MapReduce algorithm was tested through the creation of a labeled subset that included about 70% of the data of a given dataset. A 'Mapper' code was first developed to randomly assign to each sample an integer value between 1 to 100. Then, a 'Reducer' code screened the samples whose associated random number was higher than an established threshold.

With respect to the data integrity validation, the data initially obtained from the acquisition device were compared with the data downloaded from XRepo 2.0. If XRepo 2.0 granted data integrity, the two datasets would be the same. To quickly validate the integrity of the two datasets without the need to individually compare each sample, the following features were calculated:

- *Time domain*: root mean square (RMS).
- *Frequency domain*: amplitude at the motor frequency (about 30 Hz).

These values constitute the main features for detecting unbalanced shafts (Barbieri et al., 2020). Obtaining the same value from the two datasets would imply data integrity, since these features are computed using all the samples of the datasets.

5.1.3. Results and Analysis

Both the data uploading and management functionalities passed the tests by using the files from the unbalanced shaft case study.

Regarding the data integrity validation, two different datasets from the case study were uploaded to XRepo 2.0: "No unbalance" and "Highest unbalance" (Barbieri et al., 2020). The information was then downloaded through the Web UI, and data analysis was performed on both the original dataset and the dataset downloaded from XRepo 2.0. This analysis involved the calculation of the two aforementioned features (i.e. amplitude at motor frequency and RMS) for the two datasets. As shown in Table 2, the values obtained from the two datasets were equivalent.

In a nutshell, the manual validations and data comparisons showed that that XRepo 2.0 behaves as expected from the functional and data integrity perspectives.

Table 2. Calculated time and frequency domain features. The equivalence of the original data and the ones uploaded to XRepo 2.0 can be appreciated.

Dataset	No unbalance		Highest unbalance	
	Peak	RMS	Peak	RMS
Original	2.18	5.33	20.93	36.25
XRepo 2.0	2.18	5.33	20.93	36.25

5.1.4. Discussion

From a user viewpoint, the platform was able to perform the expected functionalities and guarantee data integrity. In addition, many *usability benefits* were found with respect to the previous version of XRepo and other data repositories available on the Internet (Lee et al., 2007; Loparo, 2012; Nectoux et al., 2012; Von Birgelen et al., 2018; Agogino & Goebel, 2007; Barbieri et al., 2020):

- *Filter capability*: the ability to search for data that fulfills certain criteria was especially useful when downloading files that belong to a certain timeframe or a certain operative condition.
- *Laboratories*: the ability to create different laboratories with different purposes from a single sampling enables the potential to use the same dataset for different educational purposes.
- *Data upload*: regarding the uploading process, being able to select a target sampling for an uploaded file instead of hard coding the sample ID in the file was a clear advantage over the previous version of XRepo.
- *Data format*: having a single standardized format allows domain experts to easily translate the data from the XRepo format to third party formats; e.g. the one required by Mathwork's Matlab.
- *Progress bar*: most launched procedures (e.g., file uploading, algorithm execution, etc.) can be tracked with a progress bar from the Web UI. This information becomes important while handling files with large dimensions.

Some opportunities of *improvement* were found in the prototype during the execution of the tests and will be addressed as future work:

- *Generation of subsets*: it is currently not possible to create labeled and unlabeled subsets whose intersection is null.
- *Delete datasets from the UI*: the current Web UI does not allow to delete previously uploaded data. The 'delete' functionality may be required in cases where files are unnecessary, duplicated or uploaded by mistake.
- *Default template for data context*: it was found that creating multiple samplings or laboratories requires the user to introduce certain metadata multiple times. A default template functionality would be useful.
- *Data context visualization tree*: having a tree view of all the currently created elements (e.g. Organization, System, etc.) might be useful to verify that all the elements are correctly nested.
- *Usability*: some usability improvements to the uploading process may be implemented; e.g. select multiple files at once for the upload, or select a file to be uploaded by dragging it from the user desktop to the Web UI.

5.2. Load Tests

This section illustrates the test of the most critical functionalities of XRepo 2.0 from a performance perspective.

5.2.1. Objective

Get insights about the performance and resource consumption of XRepo 2.0 when executing their most critical functionalities, through the simulation of user concurrency during time ranges.

5.2.2. Methodology

Figure 10 illustrates the steps followed to perform this validation. Each step is next detailed:

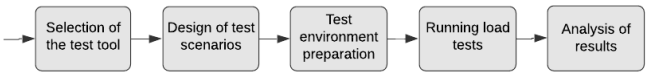

Figure 10. Methodology for the Load Tests

1. *Selection of the test tool*: Apache JMeter 5.3[19] was selected as test tool taking into account the following criteria:

[19] jmeter.apache.org

- Flexibility for setting up test scenarios.
- Documentation made available by developers' communities.
- Use in a large number of software testing projects.

2. *Design of test scenarios*: the functionalities of 'Samples Upload' and 'Samples Search' were identified as critical for the final users. Therefore, the designed load test scenarios focused on these two functionalities. Based on the educational context targeted by XRepo 2.0, we estimated the number of users that concurrently may request the two functionalities. Table 3 presents the number of users and rise time – time the test tool spends to send the requests to the platform – estimated for each test. Three test scenarios were created for the two functionalities taking into account that input parameters and usage can vary. The three test scenarios where executed in parallel to simulate concurrent users performing different actions within XRepo 2.0.

Table 3. Number of users and rise time selected for the concurrent test scenarios of the 'Samples Upload' and 'Samples Search' functionalities

No.	Test Scenario	Number of users	Rise time
1	Upload File Samples	500	1 second
2	Search for date with all fields	200	1 second
3	Search for dates with some fields	300	1 second
	Total	1000 simultaneous users	3 seconds

3. *Test environment preparation*: for the load testing process, the XRepo 2.0 components were deployed in the prototype infrastructure presented in Section 4.5. Before running the load tests, the information created in the data sources during the functionality tests was eliminated for restoring the environment to its original state. Files of 1.5MB were uploaded to test the 'Samples Upload' functionality.
4. *Running Load Test*: the designed test scenarios were run and reports were generated based on the execution results; i.e., Comma Separated Value (CSV) files.
5. *Results analysis*: based on the reports, the performance and resource consumption of XRepo 2.0 were analyzed.

5.2.3. Results and Analysis

The test took 19 minutes and 17 seconds to be executed. The results and analysis are presented based on two criteria: performance and resource consumption.

Performance Table 4 summarizes the platform response time (average, minimum, and maximum) and inflection point for each test scenario. The response time (or latency) covers the network delay and overall processing time of the platform components. The inflection point corresponds to the maximum number of requests that the server is able to process before starting to return errors. These errors are commonly due to exceeded platform capacity. These results brought evidence on the following aspects:

Table 4. Performance results for the concurrent test scenarios of the 'Samples Upload' and 'Samples Search' functionalities

No.	Test scenario	Inflection point	Response Time Success (ms)
1	Upload File Samples	372 requests	Avg.: 271536.9651 Max: 482327 Min: 45003
2	Search for dates with all fields	59 requests	Avg.: 161559.1379 Max: 232456 Min: 63965
3	Search for dates with some fields	91 requests	Avg.: 172511.4444 Max: 232690 Min: 67318

- The parallel execution of the three test scenarios contributed to reach higher response times more rapidly.
- Response times increase over time as more requests are sent.
- In the current set environment, the overall inflection point is of 522 requests disparately distributed among the tested scenarios. Thus, the inflection point is more rapidly reached in the 'Search' functionalities with respect to the 'Upload' ones. 'Search' functionalities use MapReduce functions that run faster with large files of minimum 300MB. In contrast, we had at our disposal files whose size is smaller (i.e. 1.5MB size) compared with that of standard Big Data files. As a consequence, the HDFS repository performance was hampered.

Resource consumption Table 5 summarizes the resource consumption of the three architecture nodes referred to as Big Data server, Database server, and Application server; see Section 4.5. The table shows the results of monitoring the CPU, Memory and Disk usage of said nodes during the execution of the tests. The following behaviours can be observed in the reports:

- CPU usage of the database server did not exceed 40%. This is obtained considering that the studied functionalities trigger database operations (mainly updates and queries) which are not particularly resource-intensive.
- The Web application server had several CPU consumption spikes going up to 100% and lasting over long periods of time. These peaks were reached when the server received a high number of concurrent requests. At these points, the server increased its CPU utilization to the

Table 5. Server Results for the concurrent test scenarios of the 'Samples Upload' and 'Samples Search' functionalities

Server	CPU % used	Memory % used	Disk I/O % used
Big data	Avg.: 10.104 Max: 81.772 Min: 0	Avg.: 16.964 Max: 17.623 Min: 14.78	Avg.: 0.263 Max: 82.4 Min: 0
Database	Avg. 5.321 Max: 31.658 Min: 0	Avg: 8.161 Max: 8.574 Min:7.423	Avg: 0.072 Max: 14.44 Min: 0
Web App	Avg: 60.508 Max: 100 Min: 0	Avg: 31.340 Max: 35.574 Min: 16.561	Avg.: 0.150 Max: 86.97 Min: 0

maximum available to try to resolve as many requests as possible.

- The value of 86.97% on disk usage in the Web server represents an outlier during the execution of the tests. The reason is that the server temporally stores in its disk drive (SSD) the links and file paths required by the NFS gateway to access to the HDFS during the uploading of the sample files.
- Across the execution of the tests, the Big Data server experienced different CPU usage spikes that reached 82% of CPU usage, followed by a normalization. During the high peaks, the server received a high number of requests associated to MapReduce searches over the uploaded sample files.

5.2.4. Discussion

The concurrent test scenarios allowed to establish the XRepo 2.0 platform capacity, i.e. a maximum of 522 requests per second distributed among the most critical system functionalities.

Increasing response times is an expected behavior in load testing. The results provided evidence on the behaviour of each architecture node when the number of concurrent requests increases in time. The database server has enough resources to deal with more requests. In turn, the HDFS server is able to manage a number of requests that range from 50 to 90. Finally, the application server was the most critical blocking point of the architecture. In future work, some strategies may be applied to improve the behaviour of the Big Data and Web application servers:

- *Containers*: deploy the Web application components in containers instead of virtual machines. This allows to horizontally scale the Web server in a straightforward fashion depending on the user demand.
- *Load balancer*: introduce a load balancer to distribute the requests among the different containers. Here, it is important to configure policies that check the type of requests and the available resources to make a good distribution of the requests among the Web application servers.
- *Size of files*: increase the size of files that users upload in the platform, since the behavior of the HDFS server is optimal with large files. Currently, the platform supports the upload of files with a maximum size of 1.5GB. The reason to this is a limitation on the Web browser. If the file size increased, it would be necessary to make further development at front-end level. For instance, a native client should be created to send files to the platform in background, even if the Web browser is closed.

5.3. Level of Requirement Satisfaction

Finally, how XRepo 2.0 fulfills the requirements identified in Section 3.1 is summarized as follows:

1. *Data integration, heterogeneity, and search usability* $\rightarrow$ the three presented challenges are achieved from XRepo 2.0 considering that: (i) samples can be integrated within previously defined experimental contexts; (ii) a standard format has been established for the uploading and downloading of the data; (iii) MapReduce routines can be run for filtering the data by given criteria.
2. *Data Volume* $\rightarrow$ the big data challenge was achieved by: (i) integrating the Hadoop framework with the MongoDB document database; (ii) using the processing capabilities of MapReduce. However, the big data functionality was tested with data available in the internet whose size varied from around 1MB to 1.5GB per file; see (Lee et al., 2007; Loparo, 2012; Nectoux et al., 2012; Von Birgelen et al., 2018; Agogino & Goebel, 2007; Barbieri et al., 2020). In the near future, it is desirable to execute additional tests to evaluate XRepo 2.0 operation under scenarios of larger data volume; i.e., terabytes of information.
3. *Connection* $\rightarrow$ datasets are uploaded and downloaded as batch files with a standard format; see Figure 2.
4. *Content*:
   (a) *Experimental context* $\rightarrow$ all PHM data in XRepo 2.0 are assigned to an experimental context by attaching a sampling ID to the data.
   (b) *Educational context* $\rightarrow$ an educational context for the PHM data was integrated with the experimental one. Now, data can also be categorized via: (i) 'Analysis Purpose' tags: indicating the competences that students will foster by analyzing said data; (ii) Subset: representing the role of the data within the pedagogical activity (i.e. labeled, unlabeled data); (iii) 'Failure Mode' tags: showing the analyzed failure and mechanical system.
5. *Cloud* $\rightarrow$ a batch processing model was implemented through MapReduce. Instructors can now pre-process the data online, before sharing it with students.

6. *Community*:
    - *Number of users* → load tests indicated that the overall throughput of the system is 522 requests per second, as opposed to the projected 1000. However, this is a good starting point taking into account that the tested functionalities are currently used by 1 instructor and about 60 students enrolled in a PHM course of the University of los Andes. We consider to reach the initially planned number of users by applying the strategies mentioned in Section 5.2.4.
    - *Platform access* → several roles with different permissions to the platform functionalities can be assigned to the users of the information system.
7. *Customization* → instructors can add additional fields to the 'Analysis Purpose' and 'Failure Mode' tags. Moreover, customized subsets can be generated, apart from the labeled and unlabeled ones.
8. *Open-source / free software* → the main components of the architecture are based on Hadoop and MongoDB, both of which are free to use and open-source.

5.4. Threats to Validity

With respect to the functionality validation, a detected risk is that the platform developers tested the functionality, thus likely causing closed and faultless workflows. To reduce this risk, we asked domain users to perform functionality tests to determine whether or not a specific functionality met the initial requirements. The datasets used in the validations came from two sources: i) unbalanced shaft dataset developed from the authors (Barbieri et al., 2020); ii) datasets publicly available in the internet. The unbalanced shaft dataset served as a controlled scenario to demonstrate the integrity of the data uploaded to XRepo 2.0. The datasets available in the internet served to test the platform capacity, since these files have a higher size compared to the unbalanced shaft one. The format of these datasets had to be adjusted to the XRepo 2.0 one. However, this change does not represent a threat, since the transformation step was carried out by domain users not involved in the XRepo 2.0 software development.

6. CONCLUSION AND FUTURE WORK

Given the importance of *Prognostics and Health Management* within Industry 4.0, it becomes highly advantageous for engineering students to be trained in PHM. However, datasets to develop competences in PHM are currently shared using experimental contexts with different information and without education-oriented metadata for the definition of pedagogical activities. Given that, the objective of this research work was to develop a big data information system for *education in PHM* where professors and students are respectively able to upload and access PHM sensor data with a standard format and contextual metadata. The objective has been reached by introducing *XRepo 2.0*: an information system built using open-source software from the big data ecosystem, and implementing a domain model able to represent both the experimental and educational context of the data. A prototype of XRepo 2.0 has been deployed on the Azure Cloud and validated with respect to functionality and performance criteria. The validation process demonstrated the ability of XRepo 2.0 to share PHM data within educational scenarios and established a maximum platform capacity of 522 users concurrently working on the system.

The proposed information system provides three main *benefits* with respect to: (i) XRepo 1.0; (ii) PHM big data platforms proposed in literature; (iii) the current practice of directly sharing PHM datasets without standard format and contextual metadata. These are:

1. *Educational metamodel*: apart from the standard data format and experimental context established in XRepo 1.0, data can be categorized with different criteria targeted to boost XRepo 2.0 in education, such as the PHM competences that will develop on the student, the role of the data within the pedagogical activity, and the analyzed failure and mechanical system, amongst others.
2. *Big data*: the presented information system is able to manage large amount of data by integrating the Hadoop framework with the MongoDB document database.
3. *Online processing*: considering that the information system is able to manage big data, XRepo 2.0 provides the ability for teachers to execute customized MapReduce algorithms with the objective to pre-process the data before sharing them with students.

This work contributes to *education* in PHM since has the objective to unify and share selected sensor data for the development of competences in PHM. Moreover, the design decisions taken for its implementation and the utilized technological solutions may be customized and adapted from an *enterprise* for the acquisition and sharing of PHM data within the company.

Notably, the proposed information system constitutes a preliminary concept that in the future should be further validated and improved. Some *future works* identified are:

- *Streaming processing*: currently the data is manually collected and uploaded in the information system through the Web UI. It is desirable to directly send the data from the physical equipment to XRepo 2.0. The processing of live feeds of data should be investigated by either adopting a lambda architecture (Wang et al., 2019) or capturing and messaging protocols such as MQTT[20].
- *Online processing for students*: in the current version of XRepo 2.0, online processing is only allowed to instructors. In future work, this functionality may also be

[20] mqtt.org

extended to students for facilitating the analysis process making it independent from the computational resources available from the students.

- *Functionality improvements*: during the functionality tests, few opportunities of improvements were identified, such as the generation of subsets whose intersection is null and the implementation of a visualization tree for the data context, amongst others.
- *Big data improvements*: during the load tests, few opportunities of improvements were identified, such as the utilization of containers instead of virtual machines for the Web application, the introduction of a load balancer to distribute the requests among the different containers, and the efficient processing of files with small dimensions.

OPEN SOURCE REPOSITORY

XRepo 2.0 information system is available for download under the GNU GPLv3 license at: `github.com/SELF-Software-Evolution-Lab/StandardIoTDataManager`. This repository includes a `README.md` file detailing the required steps to install XRepo 2.0 on a machine that will act as a server. It also contains a wiki with tutorials on how to use XRepo 2.0 once installed.

ACKNOWLEDGMENT

Authors would like to thank Jose Carlos Mendoza for the support during the validation process.

REFERENCES

Agogino, A., & Goebel, K. (2007). Milling data set. *NASA Ames Prognostics Data Repository, BEST Lab: Berkeley, CA, USA*.

Ardila, A., Martinez, F., Garces, K., Barbieri, G., Sanchez-Londono, D., Caielli, A., ... Fumagalli, L. (2020). XRepo - Towards an information system for prognostics and health management analysis. *Procedia Manufacturing*, *42*, 146–153.

Azure. (2020). *Azure ai guide for predictive maintenance solutions.* Retrieved 2020-09-28, from `https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/predictive-maintenance-playbook`

Barbieri, G., Sanchez-Londono, D., Cattaneo, L., Fumagalli, L., & Romero, D. (2020). A Case Study for Problem-based Learning Education in Fault Diagnosis Assessment. *IFAC-PapersOnline*.

Ben-Daya, M., Kumar, U., & Murthy, D. P. (2016). *Introduction to maintenance engineering: modelling, optimization and management*. John Wiley & Sons.

Canizo, M., Onieva, E., Conde, A., Charramendieta, S., & Trujillo, S. (2017). Real-time predictive maintenance for wind turbines using big data frameworks. In *2017 ieee international conference on prognostics and health management (icphm)* (pp. 70–77).

Cerrada, M., Sánchez, R.-V., Li, C., Pacheco, F., Cabrera, D., de Oliveira, J. V., & Vásquez, R. E. (2018). A review on data-driven fault severity assessment in rolling bearings. *Mechanical Systems and Signal Processing*, *99*, 169–196.

Condie, T., Conway, N., Alvaro, P., Hellerstein, J. M., Elmeleegy, K., & Sears, R. (2010). Mapreduce online. In *Nsdi* (Vol. 10, p. 20).

Djurdjanovic, D., Lee, J., & Ni, J. (2003). Watchdog agent—an infotronics-based prognostics approach for product performance degradation assessment and prediction. *Advanced Engineering Informatics*, *17*(3-4), 109–125.

Ebeling, C. E. (2004). *An introduction to reliability and maintainability engineering*. Tata McGraw-Hill Education.

Galar, D., & Kans, M. (2017). The impact of maintenance 4.0 and big data analytics within strategic asset management. In *Maintenance performance and measurement and management (mpmm).*

Garcia Martinez, M., & Walton, B. (2014). The wisdom of crowds: The potential of online communities as a tool for data analysis. *Technovation*, *34*(4), 203–214.

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510.

Kans, M., Campos, J., & Håkansson, L. (2020). A remote laboratory for Maintenance 4.0 training and education.

Kim, N.-H., An, D., & Choi, J.-H. (2016). *Prognostics and health management of engineering systems: An introduction*. springer.

Kotu, V., & Deshpande, B. (2014). *Predictive analytics and data mining: concepts and practice with rapidminer*. Morgan Kaufmann.

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022.

Lebold, M., & Byington, C. S. (2002). OSA-CBM architecture development with emphasis on XML implementations. *Maintenance and Reliability Conference (MARCON)*.

Lee, J., Ardakani, H. D., Yang, S., & Bagheri, B. (2015). Industrial big data analytics and cyber-physical systems for future maintenance & service innovation. *Procedia Cirp*, *38*, 3–7.

Lee, J., Lapira, E., Bagheri, B., & Kao, H.-A. (2013). Recent advances and trends in predictive manufacturing sys-

tems in big data environment. *Manufacturing Letters*, *1*(1), 38–41.

Lee, J., Lapira, E., Yang, S., & Kao, A. (2013). Predictive manufacturing system - Trends of next-generation production systems. In *Ifac proceedings volumes* (Vol. 46, pp. 150–156).

Lee, J., Qiu, H., Yu, G., Lin, J., & Services, R. T. (2007). *NASA Ames Prognostics Data Repository.* IMS, University of Cincinnati.

Li, R., Verhagen, W. J., & Curran, R. (2020). A systematic methodology for Prognostic and Health Management system architecture definition. *Reliability Engineering and System Safety*, *193*.

Loparo, K. (2012). *Bearings vibration data set* (Tech. Rep.). The Case Western Reserve University Bearing Data Center.

Martin, K. (1994). A review by discussion of condition monitoring and fault diagnosis in machine tools. *International Journal of Machine Tools and Manufacture*, *34*(4), 527–551.

Mobley, R. K. (2002). *An introduction to predictive maintenance*. Elsevier.

Moubray, J. (2001). *Reliability-centered maintenance*. Industrial Press Inc.

MTConnect Institute. (2019). *Part 3.0 - Streams Information Model.* Retrieved from `https://www.mtconnect.org/standard-download20181`

Nectoux, P., Gouriveau, R., Medjaher, K., Ramasso, E., Chebel-Morello, B., Zerhouni, N., & Varnier, C. (2012). PRONOSTIA: An experimental platform for bearings accelerated degradation tests. In *Ieee international conference on prognostics and health management.*

O'Donovan, P., Leahy, K., Bruton, K., & O'Sullivan, D. T. (2015). An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. *Journal of Big Data*, *2*(1), 1–26.

Open Geospatial Consortium. (2020). *Observations and Measurements 2.0 Examples.* Retrieved 2020-04-20, from `http://schemas.opengis.net/om/2.0/examples/`

Redelinghuys, A., Basson, A. H., & Kruger, K. (2019). A Six-Layer Digital Twin Architecture for a Manufacturing Cell. *Service Orientation in Holonic and Multi-Agent Manufacturing*, *1*, 273-284.

Sahal, R., Breslin, J. G., & Ali, M. I. (2020). Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case. *Journal of Manufacturing Systems*, *54*, 138–151.

Schroeder, G. N., Steinmetz, C., Pereira, C. E., & Espindola, D. B. (2016). Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange. *IFAC-PapersOnLine*, *49*(30), 12–17.

Sezer, E., Romero, D., Guedea, F., Macchi, M., & Emmanouilidis, C. (2018). An Industry 4.0-Enabled Low Cost Predictive Maintenance Approach for SMEs. In *Ieee international conference on engineering, technology and innovation (ice/itmc).*

Von Birgelen, A., Buratti, D., Mager, J., & Niggemann, O. (2018). Self-Organizing Maps for Anomaly Localization and Predictive Maintenance in Cyber-Physical Production Systems. In *Procedia cirp* (Vol. 72, pp. 480–485).

Wan, J., Tang, S., Li, D., Wang, S., Liu, C., Abbas, H., & Vasilakos, A. V. (2017). A manufacturing big data solution for active preventive maintenance. *IEEE Transactions on Industrial Informatics*, *13*(4), 2039–2047.

Wang, W., Fan, L., Huang, P., & Li, H. (2019, 12). A new data processing architecture for multi-scenario applications in aviation manufacturing. *IEEE Access*, *54*, 83637–83650.

Yu, W., Dillon, T., Mostafa, F., Rahayu, W., & Liu, Y. (2020). A global manufacturing big data ecosystem for fault detection in predictive maintenance. *IEEE Transactions on Industrial Informatics*, *16*(1), 183–192.