Process and Organizational Data Integration from BPMS and Relational/NoSQL Sources for Process Mining

Andrea Delgado and Daniel Calegari

Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, 11300, Uruguay

Keywords: Process Mining, Data Science, Process and Organizational Data Integration, Process Improvement.

Abstract: Business Process execution analysis is crucial for organizations to evaluate and improve them. Process mining provides the means to do so, but several challenges arise when dealing with data extraction and integration. Most scenarios consider implicit processes in support systems, with the process and organizational data being analyzed separately. Nowadays, many organizations increasingly integrate process-oriented support systems, such as BPMS, where process data execution is registered within the process engine database and organizational data in distributed potentially heterogeneous databases. They can follow the relational model or NoSQL ones, and organizational data can come from different systems, services, social media, or several other sources. Then, process and organizational data must be integrated to be used as input for process mining tasks and provide a complete view of the operation to detect and make improvements. In this paper, we extend previous work to support the collection of process and organizational data from heterogeneous sources, the integration of these data, and the automated generation of XES event logs to be used as input for process mining.

1 INTRODUCTION

Business Process Management (BPM) (van der Aalst, 2013; Dumas et al., 2018; Weske, 2019) focuses on business processes in organizations, covering their lifecycle from modeling, configuration, execution and evaluation, to support their continuous improvement. Data science (van der Aalst, 2016; IEEE, 2020) has emerged in recent years as an interdisciplinary discipline to deal with the management, analysis, and discovery of information in large volumes of data that are generated at high speed (velocity), with great variety, and also considering its veracity (the three V) (Furht and Villanustre, 2016), which is stored in structured or unstructured forms.

Process Mining (van der Aalst, 2016) is a discipline within Process Science (van der Aalst, 2016), and Data Science (van der Aalst, 2016; IEEE, 2020) that can be seen as a bridge between those areas, and has been developed in the last two decades to provide techniques, algorithms, and tools to discover information from process execution data. These data are registered within traditional organization's information systems (IS) or BPM systems (BPMS), where events that occur within each process instance (case) are registered in a so-called event log. Process mining provides three main approaches (van der Aalst, 2016): i) discovering processes from event logs, i.e., generating process models based on execution process data; ii) process conformance, i.e., checking the actual execution in event logs against existing or discovered BP models; and iii) enhancing BP models with other information such as roles and resources involved.

Organizations face several challenges regarding their daily operation and technical support infrastructure and the large amount of data they continuously gather from different and heterogeneous sources. These sources include relational and NoSQL databases nowadays, distributed within the organization or several organizations working together, traditional IS with implicit business processes, and BPMS with explicit business processes. A key challenge is to seize all this data and get information and value from it to improve their business. It involves collecting, integrating, and processing process and organizational data in an integrated manner to get a complete picture of their processes and organizational data.

In previous works (Delgado and Calegari, 2020; Calegari et al., 2021) we have presented the problem of dealing with the compartmentalized vision of processes on the one hand and organizational data on the other. We introduced a model-driven proposal for data integration using an integrated metamodel in which data is collected, and a matching algorithm re-

Delgado, A. and Calegari, D.
Process and Organizational Data Integration from BPMS and Relational/NoSQL Sources for Process Mining.
DOI: 10.5220/0011322500003266
In *Proceedings of the 17th International Conference on Software Technologies (ICSOFT 2022)*, pages 557-566
ISBN: 978-989-758-588-3; ISSN: 2184-2833

lates it. This integrated data is transformed from the metamodel to many targets, e.g., an event log for process mining. In (Delgado et al., 2021), we envision a generic Application Programming Interface (API) to collect data from any BPMS, which was initially proposed and implemented in a previous work (Delgado et al., 2016).

A significant drawback that presents the approach is the high coupling of the ETL process to the data model type and the specific implementation of the data source. Within the whole proposal, we envision a generic API to collect data from any database, both relational or NoSQL, but it was not previously defined in detail.

This paper presents an extension of the former proposal with the joint definition of both APIs that allow us to collect process and organizational data from heterogeneous sources. In particular, the generic API for organizational data will enable us to decouple the ETL process both from the data model type and the specific database implementation. The extension includes changes in the metamodel to adapt the data concepts to other data models and changes in the model-driven approach to automatically generate an extended event log that contains the corresponding organizational data for each event (activity) of the process.

The rest of the article is organized as follows: In Section 2 we introduce key concepts related to the main elements included in our proposal. In Section 3 we discuss related work. In Section 4 we describe our proposal, including the definition of the process and organizational data integration with a generic API, as well as preliminary results. In Section 5 we provide some examples of applications. Finally, in Section 6 we present some conclusions and future work.

2 BACKGROUND

Process execution data is registered from IS or BPMS from the organization's daily operation, based on implicit or explicit process models that depend on the existing settings. Most data are mainly scattered in several heterogeneous databases, which can be relational or NoSQL, with no explicit relationship between the execution of the process and the associated organizational data it manages. This compartmentalized vision of processes on the one hand and organizational data on the other is not adequate to provide the organization with the evidence-based business intelligence necessary to improve their daily operation. A typical scenario in an organization regarding the ecosystem of systems and infrastructure needed to support its daily operation is depicted in Figure 1.

As it can be seen in Figure 1, process and organizational data can come from heterogeneous sources, such as IS, BPMS, services, Internet of Things (IoT) settings, social media interacting with IS or BPMS, which are registered in distributed (probably not linked) databases, such as the process engine database, or several organizational databases both relational and NoSQL. This scenario can be extended to inter-organizational collaborative processes involving several organizations, where apart from these settings, organizations interact using interchanging messages that also contain data, which is also registered. In (Calegari et al., 2021) we discussed two main scenarios for inter-organizational collaborative BPs, one that involves direct interaction between participants and the other with interaction via an Interoperability Platform (InP) which registers all interactions. Inter-organizational collaborative processes add even more complexity to the data integration problem.

In previous works (Delgado and Calegari, 2020; Calegari et al., 2021) we have presented the problem of dealing with the compartmentalized vision of processes on the one hand and organizational data on the other. We introduced a model-driven proposal for data integration using an integrated metamodel in which data is collected and a matching algorithm to reconstruct the relationships between them. We have also defined an extension of the eXtensible Event Stream (XES) (IEEE, 2016) format, called an extended event log. An XES log represents events grouped in traces (cases) for a given process, and it is used as standard input for process mining. It provides an extension mechanism for defining new attributes to events. We include the associated data entities and attributes as defined by the matching algorithm for each event within the extended event log.

3 RELATED WORK

The idea of connecting databases with process data was mainly focused on the perspective of process mining and not on the exploitation of both sources of information altogether, i.e., process data and organizational data. Examples of this are the work in (Claes and Poels, 2014) in which the authors analyze the exploitation of database events as a source of information for event logs. Also, in (Berti and van der Aalst, 2020), the authors propose building multiple viewpoint models from databases, providing a holistic view of a process.

Nevertheless, some works are tackling the integration of both sources of information. In (de Murillas

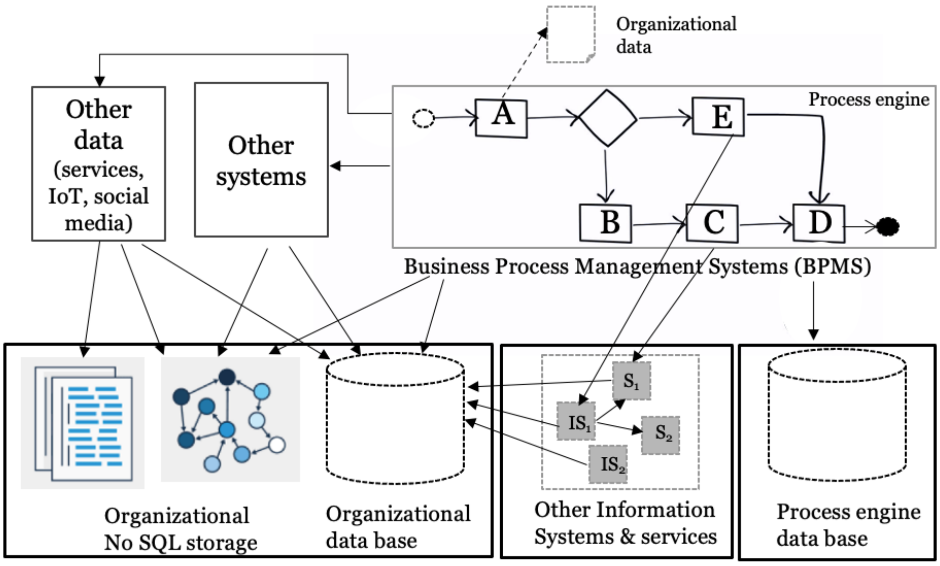


Figure 1: Linking process and organizational data from heterogeneus sources extended from (Delgado and Calegari, 2020).

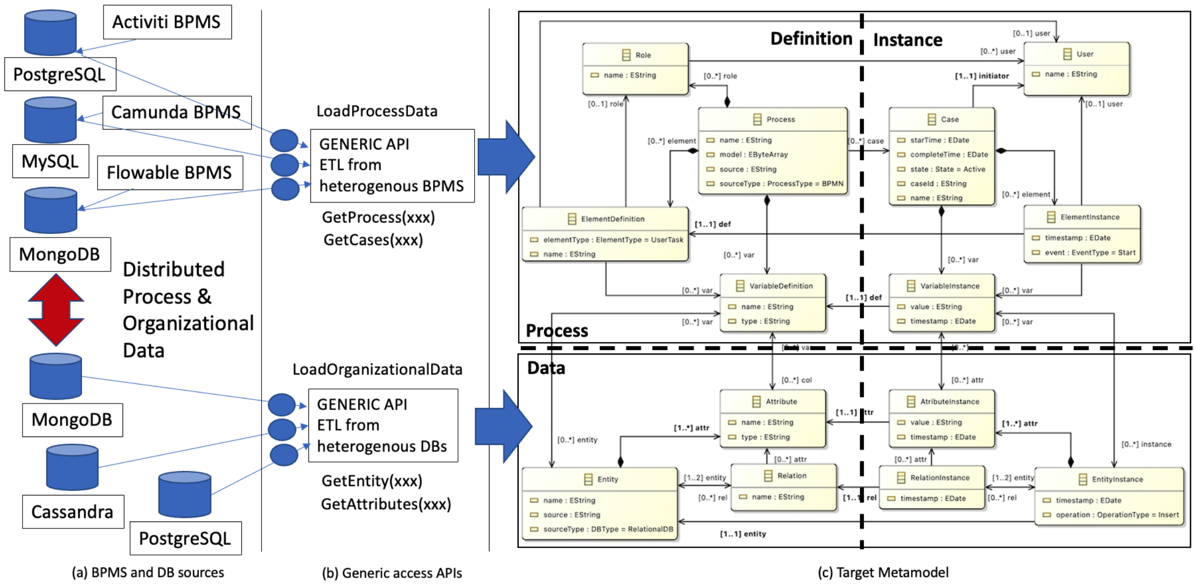
et al., 2019), the authors propose a comprehensive integration of process and data information in a consistent and unified format through the definition of a metamodel. This work has some aspects in common with our proposal in (Delgado and Calegari, 2020). Moreover, in (Tsoury et al., 2018), the authors define a conceptual framework for a deep exploration of process behavior by combining information from the event log, the database, and the transaction (redo) log. Complimentary to these ideas, in (Radeschütz et al., 2008; Radeschütz et al., 2015) the authors describe concrete matching techniques between process data and operational data. As far as we know, none of these works considered the existence of a generic API for BPMS (or databases) as we defined and discussed in (Delgado et al., 2016).

4 DATA INTEGRATION PROPOSAL

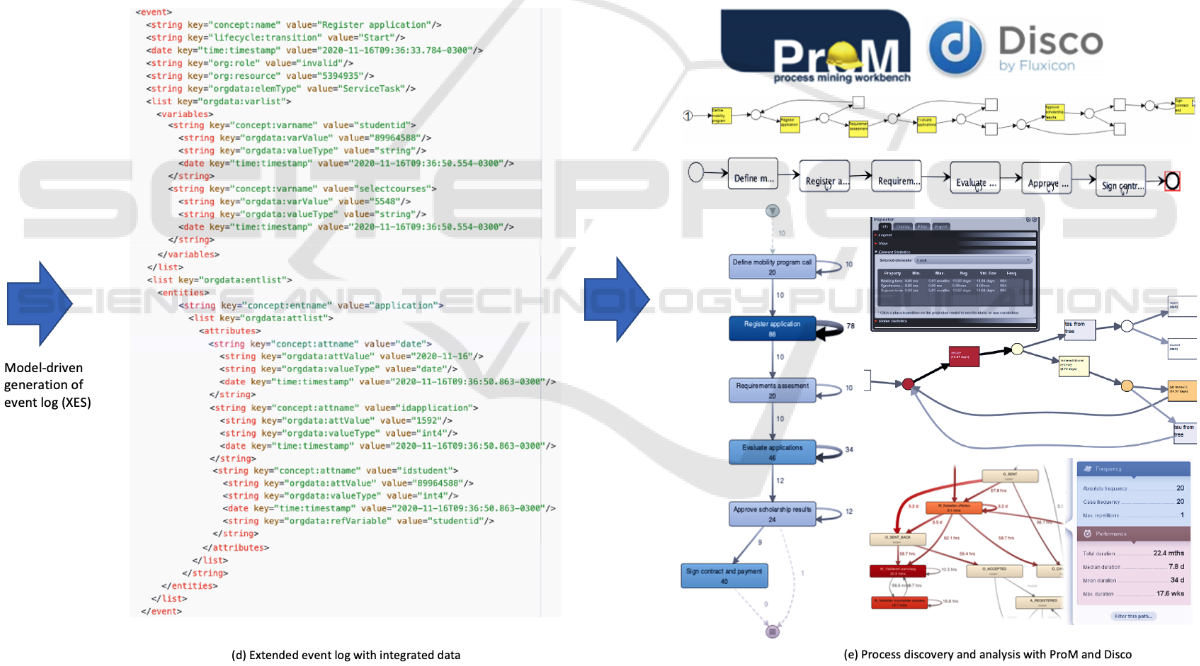
Figure 2 presents the complete integration approach we have defined. In Figure 2a the Extraction Transformation Load (ETL) process, to collect process and organizational data from heterogeneous sources. Most process mining approaches deal with implicit processes registered along with the organizational data within the execution of traditional IS in one or several distributed databases. On the contrary, we assume that processes are explicitly defined and executed within a BPMS, where process data are automatically registered in the process engine database (mostly relational, but recently there are initial implementations for NoSQL databases such as MongoDB), and organizational data are registered in a different database (which can be relational or NoSQL) within the organization, where other systems also insert and access organizational data.

In our initial proposal (Delgado and Calegari, 2020; Calegari et al., 2021) we have defined a metamodel in which process and organizational data collected from those heterogeneous sources are stored and integrated with a matching algorithm. The metamodel is divided horizontally into an upper and lower part, where the upper part corresponds to process data, and the lower one corresponds to organizational data. It is also divided vertically into a left and a right part. The left part corresponds to process and organizational definition data, and the right corresponds to process and organizational instances data. We have presented a prototype implemented in Activiti BPMS with a PostgreSQL database, and the organizational data is also registered in a PostgreSQL database. We directly extracted the corresponding data from both databases and loaded it into the metamodel, running the matching algorithm to connect the process and organizational data elements.

Although the approach is general for any BPMS and any organizational database, we realized that a significant drawback of our proposal was coupling the data extraction to a specific type of data model and a particular implementation of a database. For example, the relational model and PostgreSQL, since changing the data model or the database implementation requires a new ETL implementation to extract process and organizational data to the metamodel. Therefore,



(a) Mechanism for the ETL process and organizational data collection extended from (Delgado et al., 2021).



(b) Model-driven approach for generating the extended event log for process mining.

Figure 2: Complete approach for collection, integration and process mining of process and organizational integrated data.

in (Delgado et al., 2021) we have envisioned an extension in which a generic API is defined and used for extracting process data from BPMS. Another API is defined and used for extracting organizational data from heterogeneous databases. To include such extension for heterogeneous data models, we also extended the metamodel to better reflect different approaches.

We present here such extension, for which we integrated and extended a previously defined generic API for BPMS (Delgado et al., 2016), that we used to propose a generic user portal for BPMS that can work with different BPMS as backend. This generic API operates over a generic data model for process execution that includes several concepts and relations that are also present in the integrated metamodel, i.e., pro-

Table 1: Example of the generic API for BPMS excerpt from (Delgado et al., 2016).

Process category
GetProcessDefinitions(string name, string category, bool active):Enumerable<Process> Returns a list with existing process definition by name, category and state.
GetProcessDefinition(int processId):Process Returns the definition of the selected process.
SuspendProcess(int processId):void Suspends process with processId so no cases can be generated from it.
Cases category
CreateCase(int processId):void Allows the creation of a case from the process with id processId.
GetAllCases(string name, string creator, string user, string state):IEnumerable<Case> Returns a list of cases filtered by process name, assignee, creator or in some state.
AddCommentToCase(int caseId, string userId, string comment):void Allows to add a comment to a case by the selected user.
Tasks category
GetTasks(string name, bool assignee, string candidateUser):IEnumerable<TaskInstance> Returns a list of tasks by name, assigned or not, to be taken by the user.
TakeTask(string taskId):bool Assigns the task to the user performing the invocation.
GetTaskVariables(string taskId):IEnumerable<VariableInstance> Returns the list of variables associated with the selected task.

Table 2: Example of the generic API for heterogeneous databases.

Entity category
List <String>GetEntityNames() Returns a list with the names of existing entities.
List <Entity>GetEntityDefinitions() Returns a list of Entities, including their attributes.
Entity GetEntityDefinition(string entityName) Returns the Entity selected by name.
List<EntityInstance> GetEntityInstances(string entityName) Returns all instances of an Entity selected by name.

cess definition and instance, element definition and instance (activity, tasks), role, user. The operations defined in the API allow gathering information from the BPMS by defining a specific wrapper for each BPMS. The generic operations are translated to specific operations provided by the BPMS to obtain process instances, activities, users, etc. In Table 1 we present an example of the generic API for BPMS excerpt from (Delgado et al., 2016).

The generic API for organizational data has been defined from scratch through analyzing the data models that different NoSQL approaches defined, such as document, graphs, key-value and column-oriented, and the relational model, which was the first one considered in the proposal. In Table 2 we present an example of the generic API for organizational data for the Entity category. Based on our analysis, we have also extended the metamodel to reflect the entities better, relationships and attributes involved. We have included a few changes to the original modeling, in the first place, in the relation between entities modeling, which is now explicitly modeled by adding the notion of Relation as a concept, allowing attributes in a relation (particularly for graph databases). A Relation connects one (self-relation) or two entities and can also have attributes. Secondly, we have added attributes Source and SourceType in Entity and Process to allow traceability for the ETL process. A source can be the database connection string plus the location of the generic connector or other data to identify the origin of the data. SourceType is, for the Entity element, the type of data model, i.e., relational, document, graph, etc., and, for Process, the process type, i.e., BPMN, CMMN.

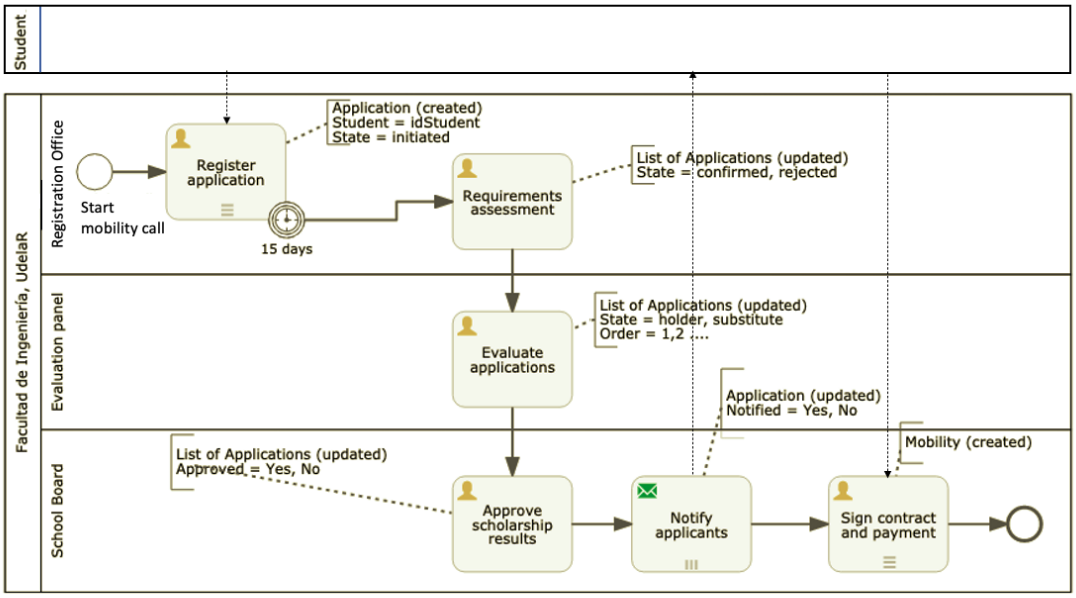
In Figure 2b we present the second part of our data integration approach. Once the metamodel is loaded with the process and organizational data, we can run the matching algorithm to reconstruct the re-

Table 3: Mapping from the integrated metamodel to the extended XES event log.

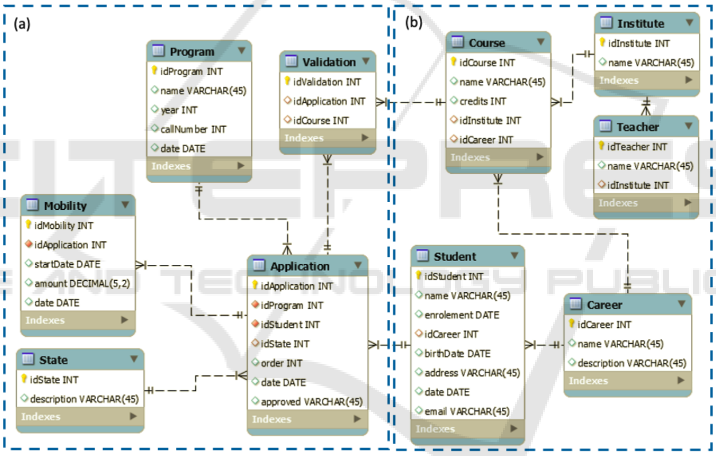
Meta-model	**XES extended**	
	Generated label	**Included in**
Process	<log></log>	
	<string key=”concept:name” value=””/ >	<log></log>
Role	<string key=”org:role” value=””/ >	<event ></event >
Element Definition	<string key=”concept:name” value=””/ >	<event ></event >
	<string key=”orgdata:elemType” value=””/ >	<event ></event >
	<list key=”orgdata:varlist”/>	<event ></event >
	<variables ></variables >	<list key=”orgdata:varlist”/>
Variable Definition	<string key=”concept:varname” value=””/ >	<variables></variables>
	<string key=”orgdata:valueType” value=””/ >	<string key=”concept:varname”/>
Case	<trace ></trace >	<log ></log >
	<date key=”startTime:timestamp” value=””/ >	<trace ></trace >
	<date key=”completTime:timestamp” value=””/>	<trace ></trace >
	<string key=”state” value=””/>	<trace ></trace >
	<string key=”id” value=””>	<trace ></trace >
User	<string key=”org:resource” value=””/ >	<event ></event >
Element Instance	<event ></event >	<trace></trace>
	<string key=”lifecycle:transition” value=””/ >	<event></event>
	<date key=”time:timestamp” value=””/ >	<event></event>
Variable Instance	<string key=”orgdata:varValue” value=””/ >	<string key=”concept:varname”/ >
	<date key=”time:timestamp” value=””/ >	<string key=”concept:varname”/ >
Entity	<list key=”orgdata:entlist”/ >	<event></event>
	<entities></entities>	<list key=”orgdata:entlist”/ >
	<string key=”concept:entname” value=””/ >	<entities></entities>
	<list key=”orgdata:attlist”/ >	<string key=”concept:entname”/>
	<attributes></attributes>	<list key=”orgdata:attlist”/ >
Attribute	<string key=”concept:attname” value=””/ >	<attributes></attributes>
	<string key=”orgdata:valueType” value=””/ >	<string key=”concept:attname”/ >
	<string key=”orgdata:refVariable” value=””/ >	<string key=”concept:attname”/ >
Entity Instance	<string key=”orgdata:operation” value=””/ >	<string key=”concept:entname”/ >
	<date key=”time:timestamp” value=””/ >	<string key=”concept:entname”/ >
Attribute Instance	<string key=”orgdata:attValue” value=””/ >	<string key=”concept:attname”/ >
	<date key=”time:timestamp” value=””/ >	<string key=”concept:attname”/ >

lations between data from process and organizational databases. From the integrated metamodel, we have defined a model-driven approach for generating the extended XES event log, which is then used as input in the ProM framework for process mining tasks. To support the automated generation of the extended event log, we have defined mappings between the metamodel concepts and the XES format tags that should be generated. In Table 3 we present the defined mappings. It is worth noting that once the metamodel is loaded with the process and organizational data for the selected process, through the ETL using the defined APIs for BPMS and organizational data, and the data is integrated using the matching algorithm, the generation of the extended XES event log from the metamodel is the same for all cases. It does not matter from which sources or type of sources (i.e., which BPMS, or which relational or NoSQL database) the data was obtained since it is expressed using concepts and relationships within the metamodel.

As shown in Figure 2b, the extended event log can be used for process discovery as input for existing algorithms. These algorithms are implemented in different plug-ins since the organizational data is treated as other attributes of the events and ignored for control flow discovery. It can also be used for process execution analysis based on traditional attributes such as timestamps for performance evaluation, as throughput time and bottlenecks. Finally, it can be used as input for a plug-in of our own (under development) to apply process mining and data mining in an integrated way. It allows a complete analysis of data by crossing process and organizational data for specific views (e.g., clustering, association rules) on data managed by different types of cases (e.g., variants) or types of cases that lead to different data results.



(a) Students Mobility BP from (Delgado and Calegari, 2020).



(b) Extended data model for the Students Mobility BP.

Figure 3: Students Mobility BP and data model from (Delgado et al., 2021).

5 EXAMPLE OF APPLICATION

We have implemented several prototypes using different technologies and defining different settings that allowed us to probe the feasibility of our proposal using the same BP and organizational data. The process we used corresponds to our university's real process, the "Student mobility" BP, where scholarships are offered to students from different interchange programs and applications are evaluated. Some are selected as holders, and other remains as the alternate. The organizational data model includes students, applications, programs, and courses. We have essay three prototype implementations, with settings combining different BPMS, different process databases, and different organizational databases. In Figure 3 we present the definition of the "Student Mobility" BP and its corresponding organizational data model from (Delgado and Calegari, 2020).

The prototype was implemented using Activiti BPMS with a PostgreSQL database for the process engine database and a PostgreSQL database for the organizational data. We implemented the ETL process directly from the databases to the metamodel. We also implemented the matching algorithm over the defined metamodel. It was presented and discussed in

```
historicTaskInstances.json
[{
  "_id": "9217bee8-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 2,
  "name": "Define mobility program call",
  "priority": 50,
  "createTime": {
    "$date": {
      "$numberLong": "1650487854232"
    }
  },
  "executionId": "9212dce4-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "taskDefinitionKey": "define_mobility",
  "formKey": "define_mobility",
  "lastUpdateTime": {
    "$date": {
      "$numberLong": "1650487854255"
    }
  },
  "startTime": {
    "$date": {
      "$numberLong": "1650487854232"
    }
  },
  "tenantId": "",
  "duration": 80665,
  "endTime": {
    "$date": {
      "$numberLong": "1650487934897"
    }
  }
},{
  "_id": "c23985fd-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 2,
  "name": "Register application",
  "priority": 50,
  "createTime": {
    "$date": {
      "$numberLong": "1650487935001"
    }
  },
  "executionId": "c22d02d3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "taskDefinitionKey": "register_application",
  "formKey": "register_application",
  "lastUpdateTime": {
    "$date": {
      "$numberLong": "1650487935004"
    }
  },
```


(a) Historic tasks as collection of documents.


```
variables.json
[{
  "_id": "94267aaa-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 1,
  "name": "teacher_list",
  "executionId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "textValue": "Lista de profesores: 1-Andrea, 8-Jorge, 2-Daniel, 3-Flavia,
  "typeName": "string"
},{
  "_id": "c1d82c0b-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 1,
  "name": "amount",
  "executionId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "longValue": 1000,
  "textValue": "1000",
  "typeName": "integer"
},{
  "_id": "c1d82c0c-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 1,
  "name": "year",
  "executionId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "longValue": 2022,
  "textValue": "2022",
  "typeName": "integer"
},{
  "_id": "c1d82c0d-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 1,
  "name": "programname",
  "executionId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "textValue": "Programa A",
  "typeName": "string"
},{
  "_id": "c1d82c0e-c0eb-11ec-ac97-e6aaeacfa8ff",
  "revision": 1,
  "name": "callnumber",
  "executionId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processDefinitionId": "mobility:1:7a138592-c0eb-11ec-ac97-e6aaeacfa8ff",
  "processInstanceId": "921267b3-c0eb-11ec-ac97-e6aaeacfa8ff",
  "longValue": 555,
  "textValue": "555",
  "typeName": "integer"
},{
```


(b) Historic variables as collection of documents.

Figure 4: Example of Flowable BPMS with Mongodb.

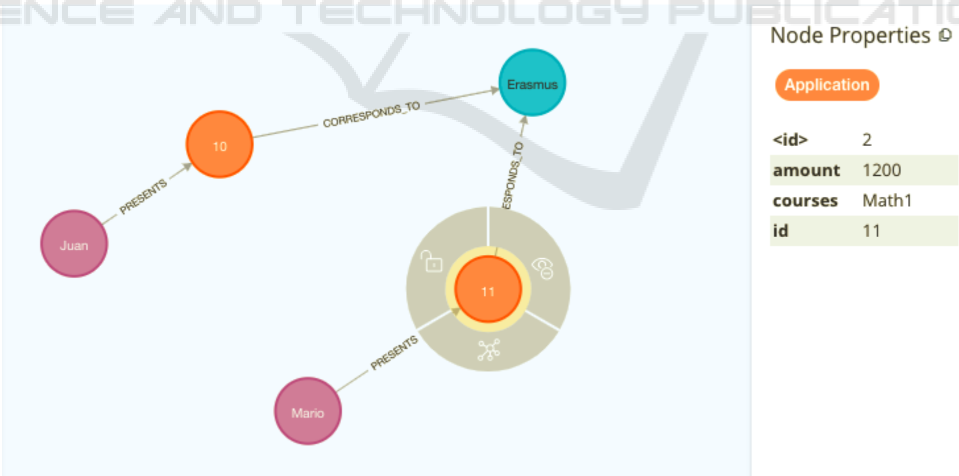


Figure 5: Example of graph modeling for the Mobility organizational data in Neo4j.

(Delgado and Calegari, 2020; Calegari et al., 2021).

The second prototype uses Flowable BPMS integrated with a MongoDB (document) database as the process engine database and a PostgreSQL database for the organizational data. MongoDB defines a collection of documents for each table in the process engine, e.g., `activityInstances`, `historicTaskInstances`, `historicActivityInstances`, `variables`, `historicVariableInstances`, etc. Each document is equivalent to a row in the table, and each field in the document is equivalent to a column.

Listing 1: Load algorithm using the generic API for organizational data.

```
db = getTargetDB()
List<Entity> entities = GetEntityDefinitions()
foreach entity in entities
    db.entity.insert(entity)
    foreach attribute in entity.Attributes
        db.attribute.insert(attribute)
    endfor
    foreach targetEntity in entity.Relationships
        db.entityEntity.insert(entity.name, targetEntity)
    endfor
    List<EntityInstance> instances = GetEntityInstances(entity.name)
    foreach instance in instances
        db.entityInstance.insert(instance)
        foreach attributeInstance in instance.attributes
            db.attributeInstance.insert(attributeInstance)
        endfor
        foreach relationshipInstance in instance.RelationshipsInstances
            foreach targetInstanceId in relationshipsInstance.TargetInstances
                db.entityInstanceEntityInstance.insert(instance.Id,
                targetInstanceId)
            endfor
        endfor
    endfor
endfor
```

Figure 4 shows the document collections for historic task instances and variables instances for the "Student Mobility" implemented in Flowable BPMS with MongoDB.

To collect the process data from Flowable, the generic API for BPMS is used, with an adapter that implements the invocation of the corresponding operations from the Flowable REST API in the same manner that we did in previous work (Delgado et al., 2016). To collect the organizational data from the PostgreSQL database, the generic API for organizational data is used in the same way, with the same select statements executed directly in the prototype.

The third prototype uses Camunda with a PostgreSQL database for the process engine and different NoSQL databases for organizational data: MongoDB and Neo4j (graph database). As before, MongoDB stores the organizational data model (Mobility database) as a collection of documents, and each document with corresponding fields, for tables such as `Student`, `Application`, and `Program`. The Neo4j database stores rows as nodes with properties for columns and labels for tables; foreign keys and join tables correspond to relationships (edges) in the graph. The Mobility database is then transformed into nodes with labels `Student`, `Application`, and `Program`, and relationships `PRESENTS` between `Student` and `Application`, and `CORRESPONDS_TO` between `Application` and `Program`. Figure 5 depicts an example of the Mobility database in Neo4j.

As before, the generic API for BPMS is used to collect process data from Camunda, with an adapter that invokes the Camunda REST API in the same manner that we did in previous work (Delgado et al., 2016). The organizational data is then collected using the generic API for organizational data, defining for each NoSQL database the corresponding adapters with specific queries to gather it. The ETL process works directly against the generic API for BPMS to load the process definition and process instance quadrant of the metamodel and against the generic API for organizational data to load the data definition and instance quadrant of the metamodel. Listing 1 presents the load algorithm for the organizational data quadrants using the operations defined in the generic API for organizational data.

Regarding the automated generation of the extended XES event log from the integrated metamodel, we have defined an M2T transformation using Acceleo[1], based on the mappings we presented in Section 4. The logs generated from the metamodel were successfully imported into both ProM and Disco tools. It confirms that the complete chain from collecting the data to loading it in the metamodel, integrating it using the matching algorithm, and generating the extended XES event log is feasible and useful for mining activities.

[1] Acceleo: https://www.eclipse.org/acceleo/

6 CONCLUSION

We have presented an extension of a proposal for process and organizational data integration from BPMS and relational/NoSQL DB sources to provide the basis for business process execution evaluation with process mining. The general proposal defines an integrated metamodel as a target of the ETL process to collect process and organizational data that is then integrated using a matching algorithm. The extension we proposed in this paper includes changes in the metamodel to adapt the data concepts to other data models, i.e., NoSQL. We also define the joint use of a generic API for BPMS and a generic API for organizational data that allows us to decouple the ETL process both from the BPMS and DB sources. In particular, we have defined a generic API for organizational data from scratch. We implemented three prototypes considering different settings of BPMS and process engine DBs, in combination with organizational DBs of different types, that allowed us to probe the feasibility of our approach.

We have defined a model-driven approach from the metamodel with the integrated process and organizational data to automatically generate an extended event log that includes the corresponding organizational data for each event (activity) of the process. This extended event log can be used as input in process mining tools. It can also be used for integrated process and data mining analysis, crossing the process view with the associated organizational data view. As future work we plan on continue applying the approach within other domains (e.g. e-Government), in heterogeneous scenarios with other BPMS and DBs.

ACKNOWLEDGEMENTS

Supported by project “Minería de procesos y datos para la mejora de procesos colaborativos aplicada a e-Government” funded by Agencia Nacional de Investigación e Innovación (ANII), Fondo María Viñas (FMV) ”Proyecto ANII N° FMV_1_2021_1_167483”, Uruguay. We would like to thank students: Alexis Artus, Andrés Borges, Santiago Sosa, Germán González, Alvaro Vallvé, Yonathan Benelli, Rafael López and Stéfano Pesamosca, for their work in the complete data integration approach prototypes.

REFERENCES

Berti, A. and van der Aalst, W. M. P. (2020). Extracting multiple viewpoint models from relational databases. *CoRR*, abs/2001.02562.

Calegari, D., Delgado, A., Artus, A., and Borges, A. (2021). Integration of business process and organizational data for evidence-based business intelligence. *CLEI Electron. J.*, 24(2).

Claes, J. and Poels, G. (2014). Merging event logs for process mining: A rule based merging method and rule suggestion algorithm. *Expert Systems and Applications*, 41(16):7291–7306.

de Murillas, E. G. L., Reijers, H. A., and van der Aalst, W. M. P. (2019). Connecting databases with process mining: a meta model and toolset. *Software and Systems Modeling*, 18(2):1209–1247.

Delgado, A. and Calegari, D. (2020). Towards a unified vision of business process and organizational data. In *XLVI Latin American Computing Conference (CLEI)*, pages 108–117. IEEE.

Delgado, A., Calegari, D., and Arrigoni, A. (2016). Towards a generic BPMS user portal definition for the execution of business processes. In *XLII Latin American Computer Conference - Selected Papers (CLEI)*, volume 329 of *ENTCS*, pages 39–59. Elsevier.

Delgado, A., Calegari, D., Marotta, A., González, L., and Tansini, L. (2021). A methodology for integrated process and data mining and analysis towards evidence-based process improvement. In *Proc. of the 16th Intl. Conf. on Software Technologies (ICSOFT)*, pages 426–437. ScitePress.

Dumas, M., Rosa, M. L., Mendling, J., and Reijers, H. A. (2018). *Fundamentals of BPM, 2nd Edition*. Springer.

Furht, B. and Villanustre, F. (2016). Introduction to big data. In Furht, B. and Villanustre, F., editors, *Big Data Technologies and Applications*, pages 3–11. Springer.

IEEE (2016). IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50.

IEEE (2020). Task Force on Data Science and Advanced Analytics. http://www.dsaa.co/.

Radeschütz, S., Mitschang, B., and Leymann, F. (2008). Matching of process data and operational data for a deep business analysis. In *Procs. 4th Int. Conf. on Interoperability for Enterprise SW and Applications, IESA*, pages 171–182. Springer.

Radeschütz, S., Schwarz, H., and Niedermann, F. (2015). Business impact analysis - a framework for a comprehensive analysis and optimization of business processes. *Comp. Sc. and Research Dev.*, 30(1):69–86.

Tsoury, A., Soffer, P., and Reinhartz-Berger, I. (2018). A conceptual framework for supporting deep exploration of business process behavior. In *Conceptual Modeling - 37th Int. Conference, ER 2018, Procs.*, volume 11157 of *LNCS*, pages 58–71. Springer.

van der Aalst, W. M. P. (2013). Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia Pacific BPM Conf. AP-BPM, Selected Papers*, volume 159 of *LNBIP*, pages 1–22. Springer.

van der Aalst, W. M. P. (2016). *Process Mining - Data Science in Action, 2nd Edition*. Springer.

Weske, M. (2019). *BPM - Concepts, Languages, Architectures, 3rd Edition*. Springer.