Integration of business process data and organizational data for evidence-based business intelligence

Andrea Delgado, Daniel Calegari, Alexis Artus, Andrés Borges
Universidad de la República, Facultad de Ingeniería,
Montevideo, Uruguay, 11300,
{*adelgado,dcalegar,alexis.artus,juan.borges*}*@fing.edu.uy*

Abstract

Organizations require a unified view of business processes data and organizational data to improve their daily operations. However, it is infrequent for both kinds of data to be consistently unified. Organizational data (e.g., clients, orders, and payments) are usually stored in many different data sources. Process data (e.g., cases, activity instances, and variables) are generally handled manually or implicitly in information systems and coupled with organizational data without clear separation. It impairs the combined application of process mining and data mining techniques for a complete evaluation of an organization's business process execution. In this paper, we deal with the integration of both kinds of data into a unified view. First, we analyze data integration scenarios and data matching problems considering intra-organizational and inter-organizational collaborative business processes. We also propose a model-driven approach to integrate several data sources, generating a unified model for evidence-based business intelligence.

Keywords: Business processes, process mining, data mining, process and organizational data, organizational improvement, business intelligence, model-driven

1 Introduction

In the last decades, Information Systems (IS) supporting Business Processes (BPs) in organizations have grown in complexity boosted by new technologies and new capabilities to allow internal and external interactions between participants and systems, increasing the volume and types of data they must handle. As a result, organizations face many challenges in managing the increasing complexity of these systems and the large volumes of data they produce, which is key to obtaining value.

Distributed infrastructures with heterogeneous technologies and connected devices add several entry points of data that have to be managed within this technical environment. Although most organizations agree that their BPs are the basis of their business, most BPs are handled manually or implicitly in their IS, and their data coupled with organizational data. Even if they are supported by a Process-Aware Information System (PAIS) [1], the link between process data and its associated organizational data is not easy to discover.

Although Business Process Management (BPM) [2, 3] helps organizations in managing their BPs from modeling and implementation to execution, evaluation, and analysis, providing support for the automation and improvement of processes, changing the focus of an organization to be a process-based one is not an easy task. In most cases, new versions of the BPs are modeled, implemented, and executed within Business Process Management Systems (BPMS) [4], which are a kind of PAIS, coexisting with older IS which continue providing support for other tasks, and managing process data and organizational data as well. In inter-organizational collaborative business processes, several systems from different participants, databases, BPMS, middleware, messages, and services exchanges are involved, increasing the complexity of gathering the data to get a complete view of their execution.

In this context, a compartmentalized vision of process data 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. On the contrary, it is required a unified view capturing all the pieces of data consistently for applying both process mining and data mining techniques to get a complete understanding of the business process execution. Therefore, while other proposals (e.g., [5, 6]) focus mainly on the process mining view for intra-organizational BPs, we focus on both the process and data mining views as well as on intra- and inter-organizational BPs, providing such unified view [7].

In previous work, we have dealt with BPs execution concepts, including data related to BP. We have defined a generic unified model and API for process execution [8, 9], and process and services execution evaluation and improvement [10]. In this paper, we propose integrating process data and organizational data as a basis for providing organizations with elements to allow a complete evaluation of their business process execution. Our contributions are twofold: the analysis of scenarios for data integration problems and data matching to link process data and organizational data, considering both intra-organizational BPs and inter-organizational collaborative BPs; and a metamodel-based proposal for the integration of several data sources containing process data and organizational data, generating a unified model to be used as the input for mining process data and organizational data altogether.

This paper is a substantially extended and thoroughly revised version of [11]. We extend the work mentioned above by providing:

- An extension of the scenarios for inter-organizational collaborative business processes (Section 2).
- The definition of an algorithm that exploits data values and timestamps to match process data and organizational data (Section 3.2), and its use through the application example (Section 4.3).
- An extension of the application example showing a generic strategy for populating a model, conforming to the integrated metamodel, from a concrete BPMS and an organizational DB (Section 4.2).

The rest of this paper is organized as follows. In Section 2, we specify the problem and the scenarios we are dealing with to integrate process data and organizational data. In Section 3 we present our metamodel-based proposal to integrate process data and organizational data, and in Section 4 we present an application example. In Section 5 we discuss related work. Finally, in Section 6 we present some conclusions and future work.

2 Problem definition

Most organizations support their business with many different systems, which are also implemented and executed in various technologies and platforms. For years, the problem was integrating such systems, providing solutions based on, e.g., centralized databases and direct connections between them, and web services to be invoked from other systems (later micro-services), and using middleware platforms [3]. This integrated view of systems has been mostly operational. Although the data and the system function can be seen as a whole by the organization and the users, the business processes they support have continued to be implicit, embedded, and spread across several systems as before. However, many advances have been made for business process execution's conceptual and technological support in the last decade. Business Process Management Systems (BPMS) [4] based on the BPMN 2.0 standard [12], e.g., Activiti[1], supports the complete BP lifecycle, from modeling and implementation to execution, evaluation, and improvement.

As depicted in Figure 1, the integration of process data and organizational data needs to consider at least two (internal) data sources: the process engine database and the organizational database. Traces of process execution elements (e.g., user and service tasks, business rules, users involved, timestamps) are registered in the process engine database and (partially) with the organizational data. However, complete organizational data (i.e., data objects) is often handled within activities and registered directly into the organizational database without a corresponding registry in the process engine. The organizational database in Figure 1 represents potentially many databases populated from other IS, i.e., they do not have to be specific for the BPs within the BPMS. They can also contain master data for the organization, not even connected with one particular IS. Service activities of a BP and the IS can access them through services. Process engine databases and ways of configuring and implementing processes vary depending on each BPMS [13]. Although it is possible to provide a unified data model for process execution for those BPMS [8], process engine databases and organizational databases are not automatically connected. Thus, the problem of relating process data and organizational data managed outside the process engine remains the same for BPMS.

The first challenge for having an integrated view is linking registers in the process engine database (event log) with data objects in the organizational database (organizational data). The following cases relate process activity with the organizational data that it is involved with:

1. A user task requires a form to be filled, and form fields are connected with process variables. Not every variable should be mapped to a DB object: they could be local variables that are only considered in the process but not interesting for the whole organization (e.g., a variable considered in a gateway). They could also be information returned by services consumed and shown to a user (e.g., document templates stored in a content management system).

[1] https://www.activiti.org/

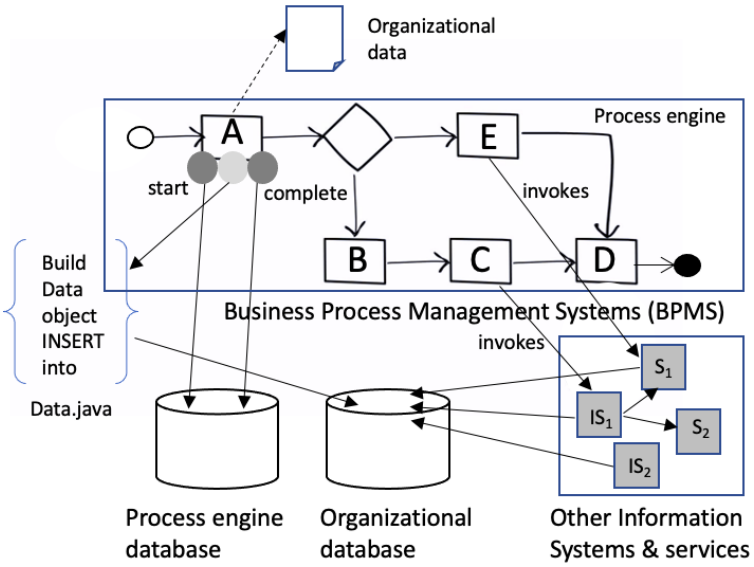


Figure 1: Linking process data and organizational data

2. A service task consumes information from other systems and stores it within a variable. This information could be later connected with the organizational data or not, depending on the case.

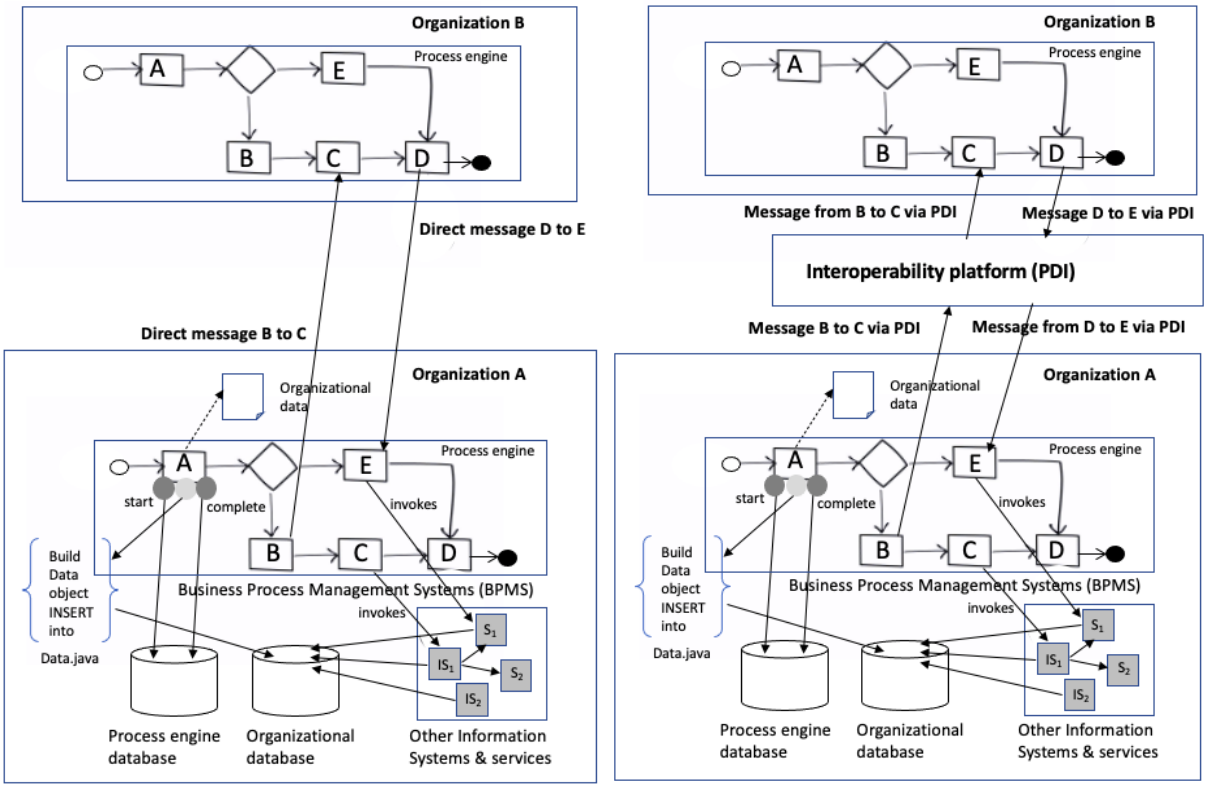
3. A business or script task uses variables to make a decision. These variables are mapped with DB objects, local variables, and external information. As a complementary aspect, also in the other scenarios, we need to consider that the data objects that a process manipulates do not necessarily have the same structure as the DB objects, e.g., the business task uses partial information from many DB objects, so from the process perspective, this data object is not considered as it is within the DB.

4. A send or receive message task provides a way for process interactions with other processes. As before, the same considerations about the information they exchange apply. Moreover, some of this external information can be connected with other organizational DBs, which must be considered in a broader scenario with many BPMS and organizational DBs.

When dealing with inter-organizational collaborative business processes, execution traces can be spread across several organizations. As a result, the corresponding data to discover the process model could be harder to get. Depending on the type and the architecture defined for the interactions between participants, traces of interactions can be centralized by providing a service-based interoperability platform enforcing all interactions to pass by it (e.g., the e-government domain [14, 15]), which can help in getting the logs. Apart from the two (internal) data sources mentioned before, external sources should also be considered (i.e., messages, activities performed by other participants, centralized registries within an interoperability platform, etc.). However, as before, these sources are not automatically connected, so linking all the elements involved makes the integration problem more complex for inter-organizational collaborative processes.

In these cases, we identified mainly two interaction scenarios between participants: (i) closed, in which interactions between organizations are explicitly defined and agreed upon as collaborative BPs (e.g., e-government domain); or (ii) open, in which organizations offer capabilities for integration, not explicitly agreeing on their BPs but mainly on the contract of the capabilities they provide or require to be able to participate in the collaboration. Figure 2 illustrates the scenarios for inter-organizational collaborative BPs.

Within these scenarios, two organizations (A and B) interact with each other via message exchange. This interaction can be implemented in several ways (web services, message queue, etc.) and registered in different ways and locations. For instance, in scenario a), Organization A registers to send a message to Organization B (message from activity B in organization A to activity C in organization B). Organization A also registers to receive a message from Organization B (message from activity D in organization B to activity E in organization A). At the same time, Organization B registers the same events for the messages it sends and receives. In this case, the choreography between them can be reconstructed from each organization's logs (if we can obtain them) and integrating with each internal process. Thus, the collaborative BP can be discovered, but the organizational data involved could be complete or not within these interactions. In scenario b), these interactions are registered within the integration platform (PDI), so the choreography can be reconstructed from these logs. Still, it must also be integrated with each internal process to discover the collaborative BP, and the organizational data involved could also be complete or not within these interactions.



(a) With direct interaction between participants

(b) With interaction via PDI between participants

Figure 2: Scenarios for inter-organizational collaborative BPs interactions

Also, depending on the type of collaborative BP and interactions (closed, open), the data from each internal process (orchestration) could be available or not, or could be necessary or not. For example, when the point of view corresponds to a single organization (e.g., organization A), it could be interested in discovering only the interactions with other participant organizations, but not these organizations' internal processes. On the other hand, when the point of view corresponds to a global vision of the collaborative BP (e.g., e-government agency), the interest could be only in discovering the choreography (without any data from the participant orchestrations) or the complete inter-organizational collaborative BP (with all data from the participants).

As presented before, the problem of integrating process data and organizational data is present within both intra- and inter-organizational collaborative BPs, being the latter a far more complex problem, since in some cases, the complete data could not be available due to restrictions from the participants. In what follows, we summarize the existent scenarios of BPs enacted within a BPMS:

1. Intra-organizational BPs: relating process data with organizational data and services execution within a single organization.

2. Inter-organizational collaborative BPs: relating process data with organizational data, messages, and services, directly interacting with participants from several organizations.

3. Inter-organizational collaborative BPs: relating process data with organizational data, messages, and services, with interactions between participants via a centralized interoperability platform.

In the following sections, we deal with the first scenario (intra-organizational BPs), which also provides the basis for the others proposed as future work.

3 Towards a unified data vision

From a general perspective, we are concerned with an integrated framework for organizational data science that involves process and data mining techniques and algorithms, the integration of process data and organizational data, data quality and process compliance assessments, as well as methodologies, guides, and tools to support all of the above [7]. This framework is meant to improve organizations based on evidence,

reducing the effort to identify and apply techniques, methodologies, and tools in isolation. As a first step, in this paper, we are dealing with integrating process data and organizational data. Other aspects of the framework and their corresponding methodologies, tools, and integration are part of future work.

We are neither focused on process discovery from database information nor data model inference from process data, as other works reviewed in Section 5. Instead, we assume the existence of processes enacted in a BPMS that access an organizational database. We are interested in exploiting such connections in both ways, i.e., from processes to data and vice versa. The proposal, depicted in Figure 3, involves three perspectives: raw data, model, and data warehouse (DW).

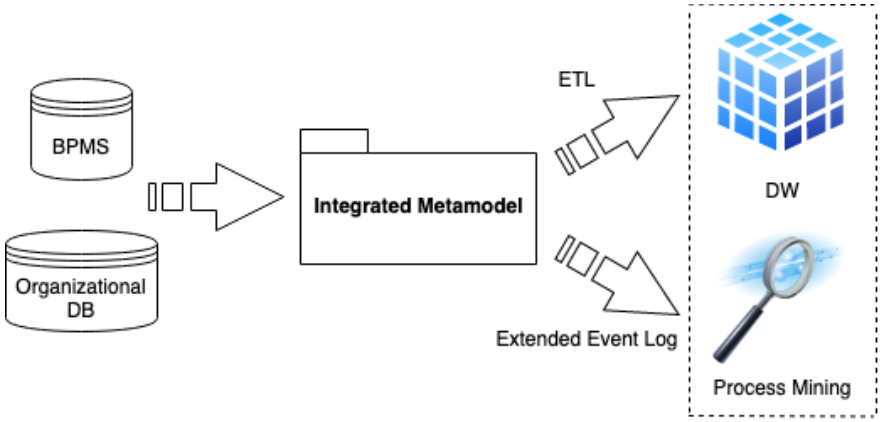


Figure 3: Raw data, model and DW perspectives

The raw data perspective involves a BPMS managing process and an organizational database with its corresponding change log to relate its data with BPMS events through time. As introduced in Section 2, there can be many organizational databases and BPMS, although we provide an initial basic scenario. We focus on integrating and processing such information and not on their governance, which is managed outside our framework. We care about every process event, not only those related to read/write action on a database, and about every data object in the organizational database linked with process activities. These two information sources are closely related but not contained in each other (neither a traditional event log nor a database contains all the information). Thus, separately they are incomplete for organizational data science. Since we are working with inter-organizational processes, this perspective can be extended with more than one of these elements and those from the integration platform.

The model perspective involves the definition of an integrated metamodel for process data and organizational data. A high-level representation of the metamodel is depicted in Figure 4 and further developed in Section 3.1. It is composed of a process view and a data view, both with two levels of information: the definition of elements and their instances.

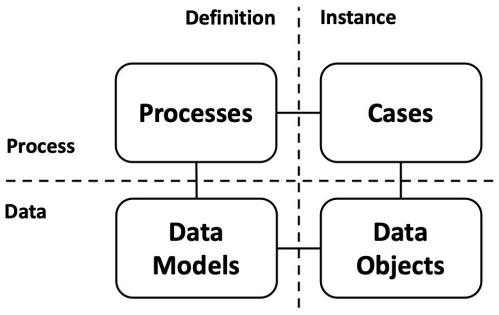


Figure 4: High-level definition of the metamodel

This metamodel is populated using data extracted from the raw data perspective, which involves an Extract, Transform, and Load (ETL) process considering data quality aspects. We are integrating a data quality framework already defined. Moreover, this data phase deals with non-functional requirements, such as confidentiality. For example, within an e-gov context, personal data protection laws require anonymizing data to be considered for a mining phase. Both data aspects are out of the scope of this paper.

Once the process data and organizational data are loaded into a model, or even during this ETL process, the relations between the process and data views need to be discovered, which is one of the main challenges discussed in Section 3.2.

The metamodel can be extended with other perspectives of interest in the context of inter-organizational collaborative BPs, for which many works can be considered (analyzed in Section 5). Furthermore, considering the massive amount of information that these models can have, this metamodel can be defined using a relational approach as in [5], or using NoSQL databases, as in [16], which allows storing large models in a NoSQL backend and efficiently use model-driven technologies to exploit such information.

Finally, the DW perspective proposes to structure the information provided by the integrated metamodel into a DW. This DW contains specific dimensions to describe facts and measures to provide meaningful answers to business questions in this context. We propose to use Pentaho[2] for the construction of the DW and the implementation of the ETL process required for building the DW. Figure 3 shows that it is possible to extract many extended event logs from the integrated metamodel used for process mining interests. In this sense, both process mining and data mining techniques can be applied from a common information source. We propose to use ProM[3] as a standard interface for integrating the whole process and access to the selected DW dimensions.

3.1 Integrated metamodel

The metamodel, depicted in Figure 5, is based on three existent proposals tailored in a common one. As a basis, we use the metamodel defined in [9], which provides an integrated vision of processes definition and instance concepts, along with their relationships as a basis for process execution.

The **process-definition quadrant** describes the relationships between processes, their corresponding elements (e.g., tasks, as defined in Figure 6), user roles and variables definition in design time. There is a direct relationship with the **process-instance quadrant** since it represents a process execution, i.e., concrete process cases (in different states, as defined in Figure 6) which have element instances, some of them pre-assigned and performed by users, that manipulate specific instances of variables. In a few words, we collect data related to the execution of a BP similar to an event log. In this sense, we discard other static process information, e.g., the order of elements and gateways. We consider the history of events corresponding to elements in a process, as defined with the `EventType` datatype (in Figure 6) within an `ElementInstance`, and the history of variables through time (`VariableInstance`), both with a timestamp.

The above part of the metamodel considers data from the process perspective, i.e., data within variables and locally or globally defined for process execution, but does not consider organizational data stored in the organizational DB. We took a more general approach to consider other kinds of structured and semi-structured sources, not only relational databases. However, further evaluation and possibly some adaptations are needed to support different semi-structured data types (e.g., NoSQL).

We define a **data-definition quadrant** describing a data model with entities and their corresponding attributes (e.g., in a database: a table and its columns), which can have references between them (e.g., in a database: primary and foreign keys). Elements in this quadrant are related to a **data-instance quadrant**, describing instances of these elements containing specific values that evolve through time. We consider the operation kind (i.e., insert, update, or delete) that changes a given entity instance and the history of attribute values and changes through time (with timestamps).

Both data and process views can be connected since variable definitions can be related to entities or attributes from a static perspective. In addition, variable instances can be related to data that evolves in the organizational database. These relations, depicted in Figure 5 as the ones crossing the vertical line separating the process and data views, must be discovered by the matching mechanism.

3.2 Matching data

The matching mechanism must deal with discovering the existing connections between process data and organizational data. Local or external information contained within variables but not connected to DB objects will be considered part of the process view but disconnected from the data view. The opposite also applies since organizational data not directly used by any process will be present in the integrated model but disconnected from process data. Nevertheless, this data must not be discarded since it allows expanding the knowledge contained in each view.

For the definition of a matching mechanism, we need to explore data matching techniques [17] which provide strategies for identifying, matching, and merging records of the same entities from several databases. These techniques consider the context in which variables are used, the attributes that identify the objects in the DB (e.g., primary keys), the type of variables, and so on. We have to deal with similar problems when applying process mining techniques for process discovery, e.g., low data quality, lack of unique entity identifiers, and computation complexity.

[2]Pentaho Platform: https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho-platform.html
[3]ProM: http://www.promtools.org

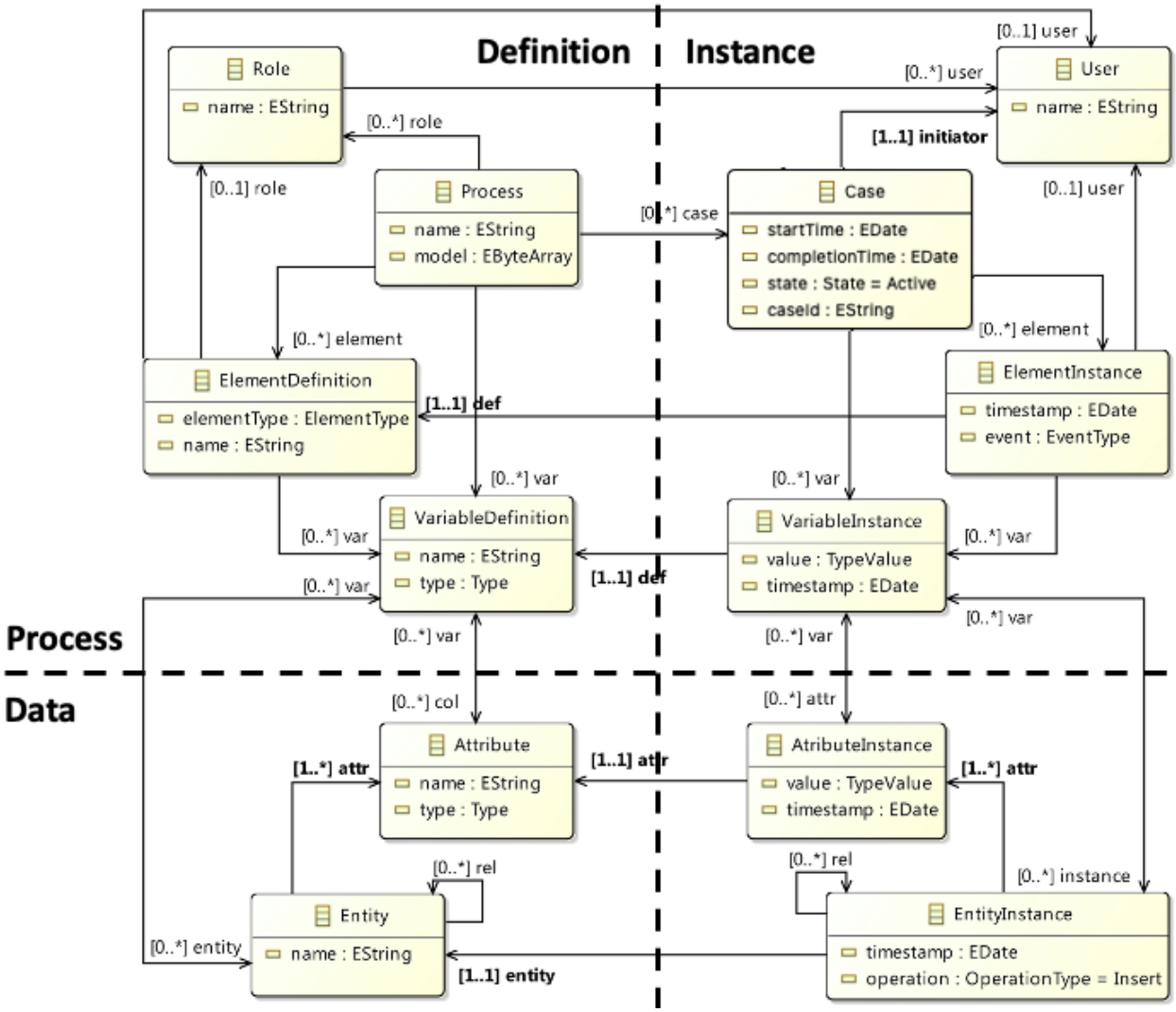


Figure 5: Integrated process data and organizational data model

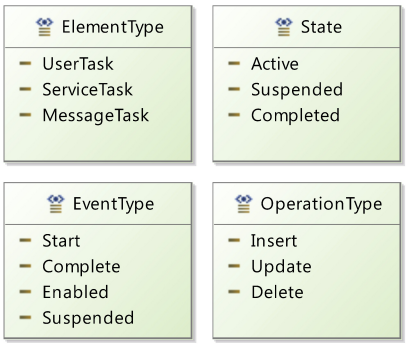


Figure 6: Datatypes of the metamodel

One of the strategies that have been addressed as current work is based on the use of timestamps. As already depicted in Figure 1, a BPMS has two main points of interaction with the process DB and the organizational DB: when an activity starts and completes. These two events are represented with the `ElementInstance` concept and potentially involves a set of `VariableInstance`. If some DB operation takes place, its information is also registered using an `EntityInstance` and its corresponding `AttributeInstance`. Thus, matches between variables and attribute instances can be discovered by searching similar values within a period near the start and complete events timestamps. We have implemented this algorithm in Java over a SQL database representing the metamodel, as detailed for the application example in Section 4. Its pseudo-code is depicted in Listing 1.

There are two conditions for finding a match:

- Match by value: the value of a process variable must exactly match the value of the organizational database attribute.
- Match by timestamp: the values of the variables and attributes must have been updated (created or modified) within a specific configurable time range.

Listing 1: Matching algorithm

```
1  //Define time tolerance
2  def confTime = x milliseconds;
3  void match() {
4      //Get every attribute instance
5      listAttInstance = getListAttributeInstance();
6
7      foreach attribute in listAttInstance {
8          //Find match in model
9          matchVar =  SELECT   vi.idVar
10                     FROM     Variableinstance vi
11                     WHERE    vi.value = attribute.value() AND
12                              vi.timestamp BETWEEN (attribute.timestamp() - confTime
13                                                    AND attribute.timestamp() + confTime);
14
15         //Set references in model
16         setReferences(matchVar, attribute);
17     }
18 }
```

The algorithm takes a configurable time tolerance (line 2) and for each `AttributeInstance` registered in the model (line 7) tries to find a matching `VariableInstance`. In the pseudo-code, the query is expressed as a SQL query (lines 9 to 13). It selects the `VariableInstance` that meets the "match by value" condition (line 11) and also the "match by timestamp" condition (lines 12 and 13). In the last case, the condition considers a range around the variable's timestamp and tolerance. Finally, if a `VariableInstance` is found, the algorithm sets the corresponding references between model elements based on the metamodel relations that cross the process and data views, i.e.,

- relates the `AttributeInstance` with the `VariableInstance` found;
- relates the `Attribute` defining the `AttributeInstance` with the corresponding `VariableDefinition` defining the `VariableInstance`;
- relates the above `VariableDefinition` with the `Entity` containing the `Attribute`;
- relates the `VariableInstance` with the concrete `EntityInstance` containing the `AttributeInstance`.

Although the DB updates depend directly on the BPMS that manages the process, values may not have to be instantly updated to the DBs. They may also have been modified before the update, i.e., there is no corresponding register in the organizational DB. Thus, there may be considerable differences in the registration times of the values in the two DBs. For this reason, the algorithm requires an adjustment phase to determine the time tolerance that achieves the best matching results.

The algorithm assumes that there will be just one attribute matching a variable. However, the metamodel accepts many possible matches that must be resolved, e.g., by considering a broader context for the variables. In this context, we are not yet considering more advanced strategies using the type of variables, the other variables that can be updated within the same `ElementInstance`, and so on. Furthermore, we are neither considering data quality problems, as mentioned before. These are reasonable assumptions in an intra-organizational scenario driven by a BPMS, like the one we are experimenting with.

In more complex scenarios, e.g., processes with parallel tasks or distributed DBs, clock synchronization imposes challenges to the timestamp-based approach, which needs to be addressed as future work. In inter-organizational collaborative BPs, we also need to consider the latency in communications and clocks with time differences in the different servers and participants.

3.3 Exploiting process data and organizational data

The integrated view allows exploiting its information in both directions: from the process view, we can relate cases and element instances to the data they managed, and from the data view, we can relate data instances (i.e., creation of orders) to the cases and elements that managed such data.

The first point of exploitation is the integrated metamodel that captures all the pieces of information. Once a model is populated from a concrete BPMS and an organizational DB, the model can be exploited by performing process data and organizational data queries. The metamodel provides us with some basic possibilities, such as obtaining a snapshot of an execution trace and its associated data at a time.

Moreover, we can define a model to text transformations for building many different event logs to apply process mining techniques. We can also use an ETL process to populate the DW dimensions to analyze the integrated data from the two points of view mentioned earlier: process data and organizational data. This is the second point of exploitation.

We envision defining dimensions related to the two levels of abstraction in the metamodel: i) process definition, element definition, and variables definition; ii) process case, element instance, variable instance, as shown in Figure 5 left and right, respectively. By drilling up and down in the defined dimensions, we will be able to analyze the integrated data to identify, for example, from the process point of view, cases that present the same behavior concerning a DB object and from the data point of view, DB objects that are managed by the same type of cases. We can also enhance this type of cross-information by adding service and message interactions between participants of inter-organizational processes, such as which organizations consume the data that a process manages or which changes are made in the organizational DB not managed by a process (potential data leaks).

4 Application example

In this section, we present an example of integrating data through the defined metamodel. Although it is simple, it allows presenting the main elements of the proposal introduced in Section 3: registering of data (raw perspective) and the ETL process and matching algorithm for integrating such data (model perspective). The exploitation of process data and organizational data (DW perspective) is left for future work.

4.1 Students mobility process and data model

The "Students Mobility" business process is a real process from our university describing the evaluation of mobility programs for students. Calls are opened each year, students present their applications, and scholarships are assigned after some controls and evaluations. A simplified version of this process is presented in Figure 7, modeled in BPMN 2.0 using the Activiti Kickstart App web application.

The process is supported by the organizational data model presented in Figure 8. For the sake of simplicity, this data model is an excerpt of a broader organizational database since it only considers data objects related to this specific process: general information of a mobility program (`Program`) and the student's applications (`Application`) with the corresponding data of the students that present the application (`Student`), the state of each application (`State`) and the scholarships assigned (`Mobility`).

The process starts with an open mobility program call. Within the "Register application" task, the Registration Office receives students' applications (for 15 days). For each one of them, a new `Application` is registered in the organizational database in the state "Initiated", also linking it with the corresponding open `Program` (i.e., Erasmus 2020 first semester) and `Student`. Then, within the "Requirements assessment" task, every application can be downloaded and checked (all at once). As a result, the applications are updated in the organizational database by changing their states to "Confirmed" or "Rejected". Those rejected applications are no longer considered within the process. An evaluation panel carries out the "Evaluate applications" task. The list of applicants is evaluated and ranked (ordered from 1 to n). Potential holders and substitutes are defined depending on the number of available places for the program call. As a result, the applications are updated within the organizational database, changing their states to "Holder" or "Substitute" and adding the order assigned within the applicants' list. After that, the School Board evaluates the resulting list of applications within the "Approve scholarship results" task. The organizational database is updated for setting the holder's and substitute's applications' approved status to true. Within the "Notify applicants" task, each applicant is notified of the results; also those rejected during requirements assessment. This task is updated with the notified status of the application. Finally, scholarship holders sign their contracts within the "Sign contract and payment" task and receive their payment. The organizational database is updated by creating a `Mobility` for each holder application.

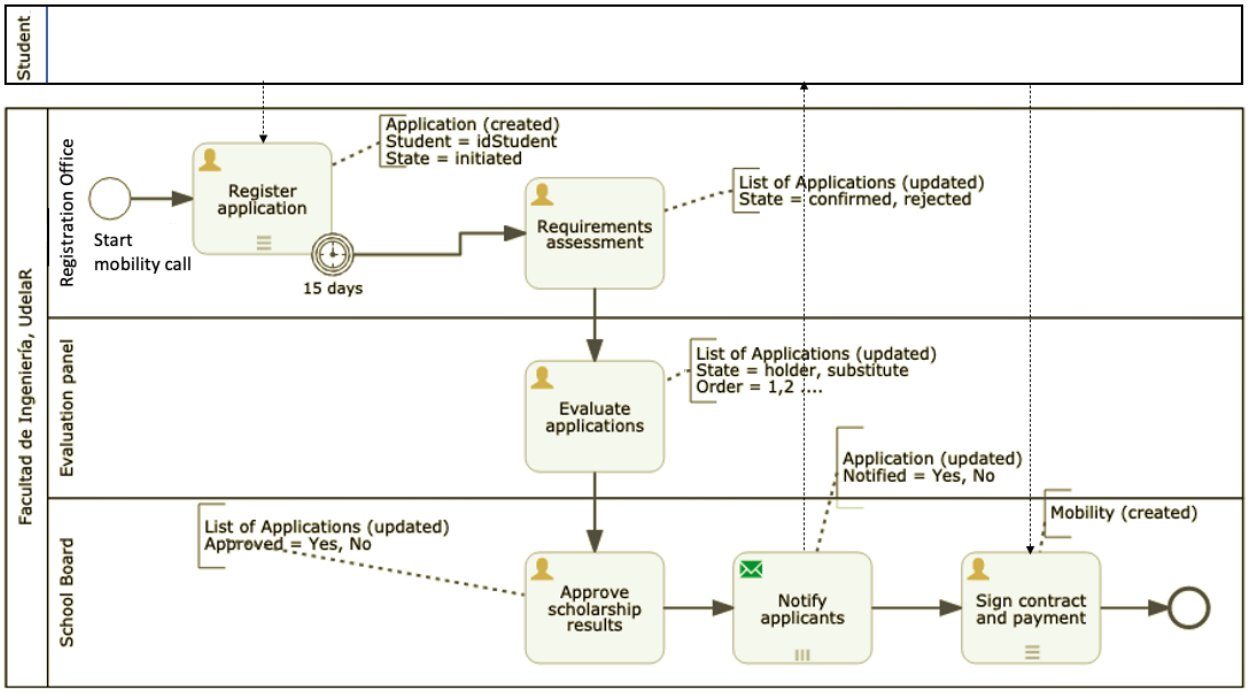


Figure 7: Students Mobility business process

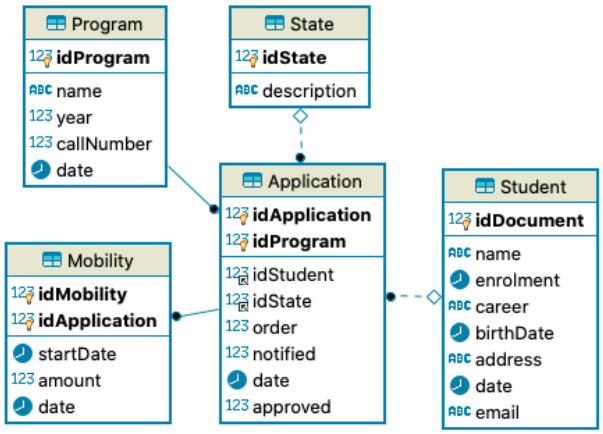


Figure 8: Students Mobility organizational data model

We implemented the case study using Activiti 6.0 BPMS [4] with a MySQL 8.0.20 database [5] (both community editions) where we defined two schemas: the "activiti6ui" schema (as defined by Activiti BPMS) for the process engine database, and the "mobility" schema for the organizational database.

Figure 9 depicts the Activiti BPM Task App with the execution of the "Register application" task in which students submit their applications to the mobility call. The form defines three variables: the program described as "Erasmus" for this case, student name, and identification document. Figure 10 shows the data registered for the process engine and the organizational database regarding the execution of this task.

It can be seen that two student applications were received for the Program "Erasmus" within the case *2515*: one from *Jane Doe* with identification document *12223334*, and the other from *John Smith* with identification document *23334445*. Both students were already stored in the organizational database, and are referenced from the `Application` table to the `Student` table, as shown in Figure 8.

4.2 ETL from process and organizational databases to the metamodel

As a first step towards using the integrated metamodel, we defined a new database schema with tables reflecting the metamodel concepts. We populated it with the data from the process engine and the organizational

[4] https://www.activiti.org/
[5] https://www.mysql.com/products/community/

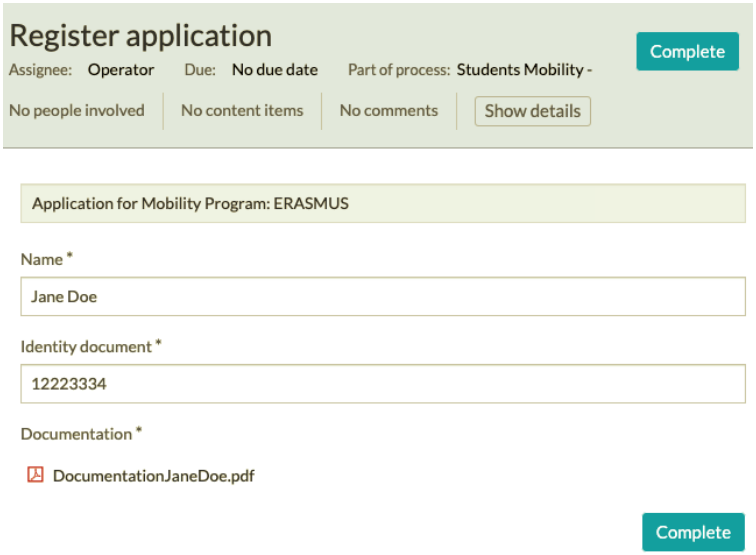


Figure 9: Example of execution of the task "Register application" in Activiti

id_	PROC_DEF_ID_	TASK_DEF_KEY_	PROC_INST_ID_	NAME_	ASSIGNEE_
2528	sm:5:80	task1	2515	Register application	oper
2534	sm:5:80	task1	2515	Register application	oper
2541	sm:5:80	task2	2515	Requirements assesment	NULL

START_TIME_	END_TIME_	NAME_	VERSION_
2020-05-20 16:25:01.440	2020-05-20 16:26:21.815	Students Mobility	5
2020-05-20 16:26:21.822	2020-05-20 16:27:50.353	Students Mobility	5
2020-05-20 16:30:02.981	NULL	Students Mobility	5

(a) Process data

Application	State	Program	StudentDoc	StudentName	timestamp
1	Initiated	Erasmus	12223334	Jane Doe	2020-05-20 16:26:25
2	Initiated	Erasmus	23334445	John Smith	2020-05-20 16:27:55

(b) Organizational data

Figure 10: Example of data registration from "Register application" task execution

databases. In Figure 11 we present the data model for the metamodel implementation in PostgreSQL.

Several queries were performed to get the data extracted and loaded into the metamodel, following an Extract, Transform, and Load (ETL) process. In the following, we present the step-by-step ETL process we carried out within this example, noting that although conceptually it would be the same for other sources and metamodel implementations, specific elements would need to be changed.

4.2.1 ETL for Process Definition and Instance quadrants

To populate the two superior quadrants of the metamodel, i.e., Process Definition and Process Instance (Cases), we use the process engine database, in this case, Activiti [18]. Figure 12 shows an excerpt of Activiti database tables with those used within this application example. Its database defines over 30 tables covering three different objectives: tables for deployment artifacts and runtime execution of process instances and their activities (in Figure 12(a)), tables for user and group management purposes (in Figure 12(b)), and tables used to register the history of the execution of process instances and their activities (in Figure 12(c)).

We had to define a flow for the data extraction and the metamodel population to avoid problems due to foreign key restrictions between tables in the metamodel. In the first place, users and roles are loaded, extracting the data from the `act_id_membership` table shown in Figure 12(a), which relates Role in the process definition quadrant with User in the process instance quadrant. With this data, Role and User tables are populated along with the relationship between User and Role. After this, the process definition can be loaded by querying the data from the `act_re_procdef` and `act_ge_bytearray` tables shown in Figure 12. These tables store static process data (e.g., name, version, description, and deployment id) and the process model XML file. With this data, we populate the `Process` table.

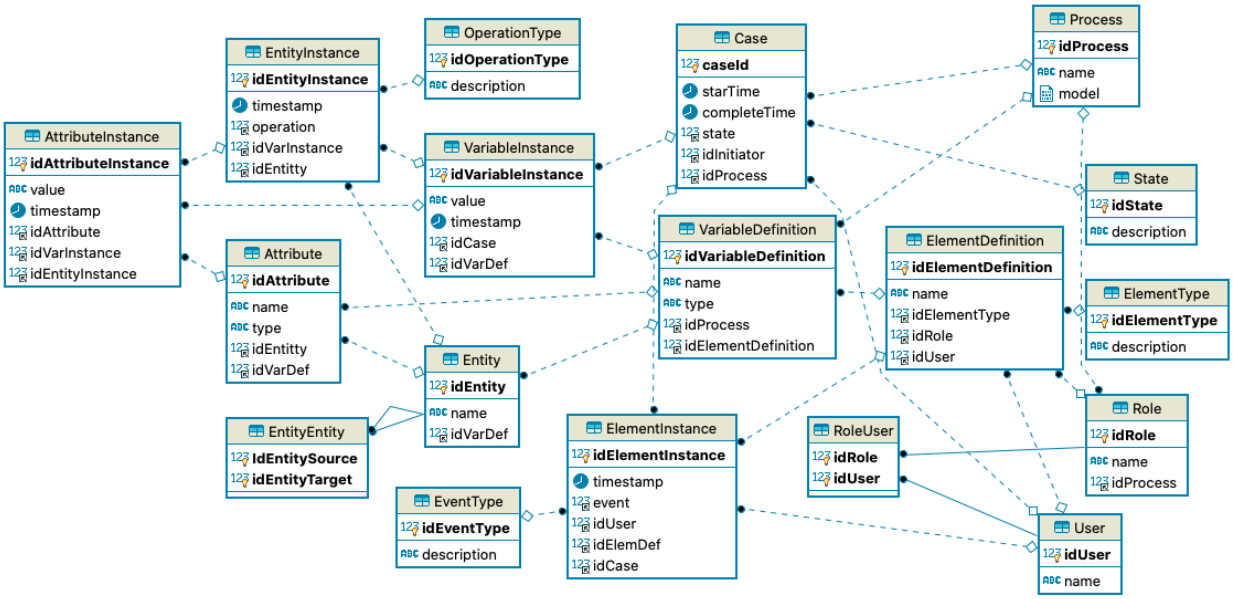


Figure 11: Data model of the metamodel implementation in PostgreSQL database

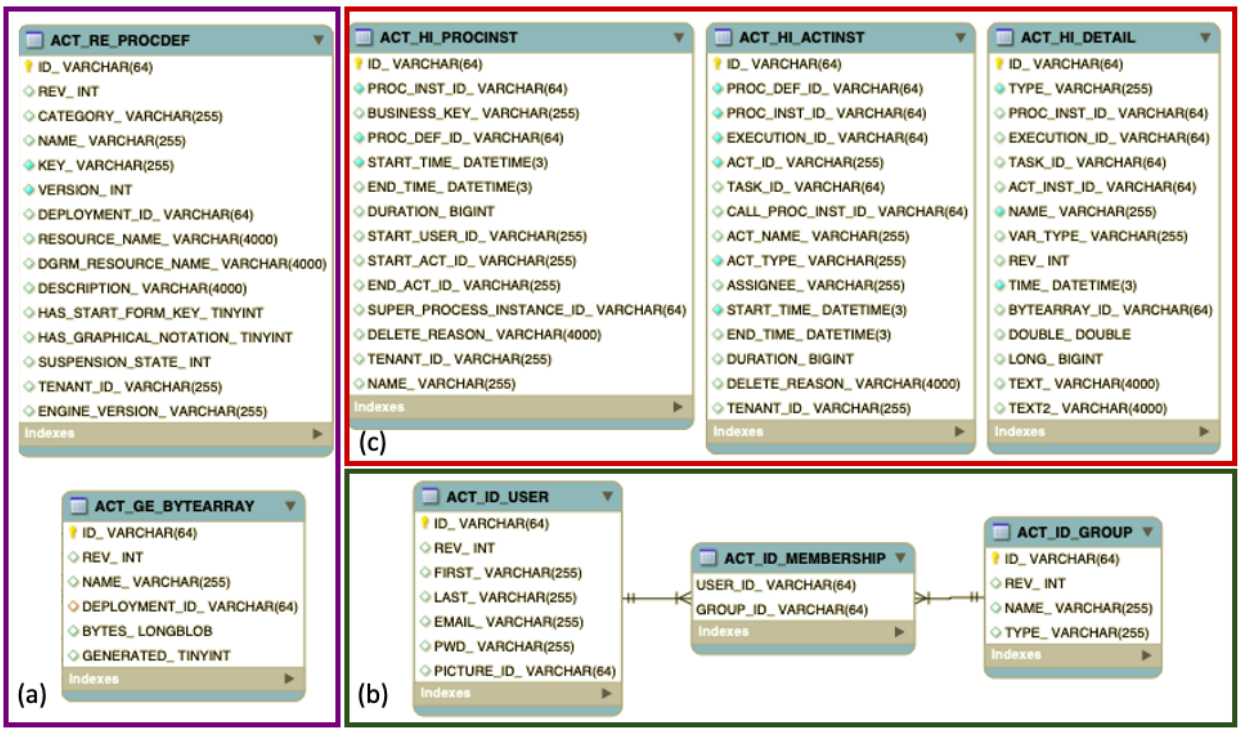


Figure 12: Excerpt of Activiti database tables

Then, process instances can be loaded into the `Case` table, which references the process definition we loaded before. To do so, we queried the data in table `act_hi_procinst`, also shown in Figure 12 (c). This table stores the history of cases (process instances) for the selected process and their data (e.g., process instance id, start time, duration, and initiator user). Figure 13 shows both the query and excerpt of the results, where selected data are presented. It is worth noting that although the name of the table seems to refer only to activities, every element defined is registered in this table.

The `delete_reason_`, `start_time_`, and `end_time_` attributes show the case state. The first one indicates if the process has been suspended if it is null. If the `end_time_` attribute is present, the case has been completed. If it is also null, the case is active. With this data, the `Case` table is populated along with the relationships with tables User and Process definition.

```
SELECT id_, proc_def_id_, start_time_, end_time_, start_user_id_, delete_reason_
FROM act_hi_procinst;
```

id_	proc_def_id_	start_time_	end_time_	start_user_id_	delete_reason_
765328	mobility:5:732510	2020-10-14 22:37:18.215000	2020-10-14 22:38:14.119000	admin	<null>
765491	mobility:5:732510	2020-10-14 22:38:14.401000	2020-10-14 22:39:06.114000	admin	Cancelled by Admin
765686	mobility:5:732510	2020-10-14 22:39:06.491000	2020-10-14 22:39:51.920000	admin	<null>
765851	mobility:5:732510	2020-10-14 22:39:52.238000	2020-10-14 22:41:10.062000	admin	<null>
766141	mobility:5:732510	2020-10-14 22:41:10.241000	2020-10-14 22:42:45.511000	admin	<null>

Figure 13: Example of querying data from Activiti's process engine database for history of cases

After this, we load the definition of process elements. This is a particular case since the process engine database does not register the elements within a process in database tables but only in the XML process model. From the XML, it is easy to obtain the elements defined, e.g., user tasks and the roles that can perform them, going from there to corresponding users. With this data, the `ElementDefinition` table is populated. The `ElementInstance` table is populated from table `act_hi_actinst`, also shown in Figure 12. This table stores the historic of element instances within cases, and their associated data (e.g., activity id, name, type, and start and end time).

Finally, process variables are loaded into the metamodel, including definitions and instances. Since there is no single table in the Activiti data model where variables definition is registered, this query is more complex. It involves table `act_hi_actinst` to obtain the information of activity instances and table `act_hi_detail` (shown in Figure 12) to obtain the historical information of the variables on each activity instance (e.g., name and type). Also, a *group by* is needed to get variables only once. With this data, the `VariableDefinition` table and its relationships with `ElementDefinition` and `Process` tables are populated. Figure 14 shows both the query and excerpt of the results, where the selected data are presented.

```
SELECT aha.proc_def_id_, act_id_, name_, var_type_
FROM act_hi_actinst aha
     JOIN act_hi_detail ahd on aha.id_ = ahd.act_inst_id
WHERE var_type_ IS NOT NULL
GROUP BY (aha.proc_def_id_, act_id_, name_, var_type_);
```

proc_def_id_	act_id_	name_	var_type_
mobility:5:732510	evaluate_applications	app	long
mobility:5:732510	define_mobility	programname	string
mobility:5:732510	define_mobility	program_id	long
mobility:5:732510	sign_contract	sign	string
mobility:5:732510	register_application	courses_list	string
mobility:5:732510	define_mobility	callnumber	string

Figure 14: Example of querying data from Activiti's process engine database for definition of process variables

Getting data from variable instances presents an even more complex scenario. As explained before, a variable's definition does not have an id in Activiti, so relating variable instances with their corresponding definition is challenging. The first step is to get variable instances data from tables `act_hi_actinst` and `act_hi_detail` (in Figure 12(c)), which results in getting all attributes but the corresponding variable definition. To get that reference, we work with the already loaded data in the metamodel, relating the variable name with the `Process` id. The `ElementDefinition` id obtains a unique tuple that corresponds to the variable definition. The reason for doing this is that variable names can be the same within different processes and tasks. The `VariableInstance` table is populated with this data along with its relationships with `ElementInstance`, `VariableDefinition`, and `Case` tables.

4.2.2 ETL for Data Definition and Instance quadrants

We use the organizational database to populate the metamodel's two inferior quadrants, i.e., Data Definition and Data Instance. In this case, we used PostgreSQL. We use the database log to match process and organizational data. We extract the current value of the organizational data in the database and the history of each variable's values to relate each value within the process. The database log must present the adequate granularity level, i.e., log data from the write operations (insert, update and delete) to populate

the metamodel. Figure 15 shows an example of a database log from PostgreSQL. Within the log, there is one line for each operation with its corresponding timestamp, and in the following line, its parameter values. Using the log implies parsing a text file, including the elements mentioned before, and loading them into a list of nodes (`ParserLog` objects), one for each of the two lines described, including all data from the entity, timestamp, operation, parameters, and corresponding attribute's values.

```
2020-10-14 22:37:45.378 LOG: insert into validation (idapplicatoin, idcourse, idvalidation) values ($1, $2, $3)
2020-10-14 22:37:45.378 DETAIL: parameters: $1 = '1536', $2 = '8', $3 = '8861'
2020-10-14 22:37:45.381 LOG: insert into validation (idapplicatoin, idcourse, idvalidation) values ($1, $2, $3)
2020-10-14 22:37:45.381 DETAIL: parameters: $1 = '1536', $2 = '10', $3 = '8862'
2020-10-14 22:37:56.242 LOG: update application set amount=$1, approved=$2, date=$3, notified=$4 ...
2020-10-14 22:37:56.242 DETAIL: parameters: $1 = '854', $2 = 'NULL', $3 = '2020-10-14', $4 = 'NULL' ...
```

Figure 15: Example of database log from PostgreSQL with adequate granularity level

To load the Data definition quadrant, we populate the `Entity` and `Attribute` tables with the data from the tables defined in the organizational database, loading each `Entity` and loading each of its attributes with the corresponding reference from the entity. For the relationships between entities, we navigate each foreign key registered in the database, loading this relationship in the `entityentity` table referencing each of the related entities.

After that, for populating the Data instance quadrant, an `EntityInstance` is created from the parsed log with the corresponding operation and timestamp. Then we iterate over each entity's attributes, checking with the corresponding table definition whether an attribute is a foreign key. If it is not a foreign key, we insert the attribute with its value and timestamp. In the case that it is a foreign key, if the corresponding entity is already created, we insert the relationship in the `entityentity` table. Suppose the `EntityInstance` is not present. In that case, a recursive step is taken to get the key from the organizational database, returning to insert an EntityInstance as before, and the relationship between instances.

It is worth mentioning that the recursion has a parameterizable number of steps to run, so the number of foreign keys to find is configurable. This allows us to define how many extra organizational tables outside the specific data model support the process we want to include in the metamodel to cut out the potential load of the complete organizational database. For example, in the StudentMobility process, we need references to other existing tables, such as careers, courses, and institutes, that are already defined within other systems. Still, the process includes references to them (e.g., career FK in the Student table see Figure 8).

4.2.3 Summary of the ETL process to the metamodel

The ETL process we have presented allowed us to populate the metamodel with the data from Activiti's process engine database and from the organizational database, going step by step through the defined tables respecting the references between them. Due to the existing references in the metamodel, we have defined a specific order for loading data incrementally that must be respected to insert data in the metamodel respecting the restrictions, both for the Process and the Data quadrant. We have also automated the process using a Java application that allows us to execute the ETL process, as presented before. We use Spring Data JPA defining an entity for each concept in the Activiti database, such as process, cases, elements, variables, saving them using repository elements to the metamodel database. Listing 2 shows an example of the Java pseudo-code to populate the process quadrants from the Activiti database. In Listing 3 the getCases() method is shown as an example of querying the Activiti database to get each type of element that are invoked from Listing 2 to populate the metamodel.

Listing 2: Java pseudo-code for the ETL process populating the process quadrants from Activiti database

```
1  public void activiti(Int processId){
2      //Connect to Activiti database
3      dbActiviti.connectActiviti();
4
5      //Get and save users and roles
6      users = dbActiviti.updateUsersAndRol(processId);
7
8      //Get and save process
9      process = dbActiviti.getProcess(processId);
10     processRepository.save(process);
11
12     //Get and save cases
13     cases_list = dbActiviti.getCases(processId);
14     caseRepository.saveAll(cases_list);
15
16     //Get and save element definitions
17     element_definition_list = dbActiviti.getElementDefinition(processId);
18     elementDefRepository.saveAll(element_definition_list);
19
```

```
20      //Get and save element instances
21      element_instance_list = dbActiviti.getElementIntance(processId);
22      elementInstanceRepository.saveAll(element_instance_list);
23
24      //Get and save element variable definitions
25      var_def_list = dbActiviti.getVariableDefinition(processId);
26      variableDefRepository.saveAll(var_def_list);
27
28      //Get and save element variable instances
29      var_instance_list = dbActiviti.getVariableIntance(processId);
30      variableInstanceRepository.saveAll(var_instance_list);
31  }
```

Listing 3: Java code for the `getCases()` method to get all cases from the Activiti database

```
1   public List<Case> getCases(Int processId) throws SQLException, ParseException {
2       //Connect to Activiti database
3       Statement statement = this.connectionActiviti.createStatement();
4
5       //Query for all cases in Activiti database from the table act_hi_procinst
6       String query = "SELECT id_, proc_def_id_, start_time_, end_time_,"
7                       + "start_user_id_, delete_reason_"
8                       + "FROM act_hi_procinst WHERE proc_def_id_ = processId";
9
10      //Execute query and prepare the list of cases to return
11      ResultSet resultSet = statement.executeQuery(query);
12      List<Case> ret = new ArrayList<Case>();
13      SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss.SSS");
14
15      //Get and set each case with its attributes in the cases list
16      while (resultSet.next()) {
17          Case case_ = new Case();
18          case_.setCaseId(resultSet.getLong(1));
19          case_.setProcess(activityUtils.idProcessToLong(resultSet.getString(2)));
20          case_.setStarTime(resultSet.getTimestamp(3));
21
22          //Get and set the timestamp when end_time is not null
23          if (resultSet.getString(4) != null) {
24              case_.setCompleteTime(resultSet.getTimestamp(4));
25          }
26
27          //Get and set the user
28          case_.setUser(userRepository.findByName(resultSet.getString(5)).getIdUser());
29
30          //Get and set the state of the case if there is a delete reason
31          //or if there is end_time timestamp
32          if(resultSet.getString(6)!=null){
33              case_.setState(Constants.case_state_suspended);
34          }else {
35              if(resultSet.getString(4)!=null){
36                  case_.setState(Constants.case_state_completed);
37              }else {
38                  case_.setState(Constants.case_state_active);
39              }
40          }
41          ret.add(case_);
42      }
43      return ret;
44  }
```

4.3 Integrated data and analysis

Figure 16 shows an excerpt of the data populated from the process engine and the organizational database (tables are joined for its visualization), similar to the data presented in Figure 10: (a) two instances of the "Register application" task, each one with the three variables `idVarDef` defined in the form, and a registry of the complete event (to which the `idVarInst` is associated); and (b) the corresponding applications as two `EntityInstance` (`idEntityInstance` 5 and 6) of `Entity` (`idEntity` 2) and values for their `AttributeInstance` (`idAttributeInstance` from 37 to 52) for the defined `Attribute` (`idAttribute` from 6 to 13). The `idVarInstance` column only has "null" values since there are no references yet from the organizational data to the process variables.

As presented in Section 3.2, we aim to discover the existing relationships between process and data concepts, i.e., `Attribute` and `Entity` (data-definition quadrant) with `VariableDefinition` (process-definition quadrant), and the corresponding ones for instances: `AttributeInstance` and `EntityInstance` (data-instance quadrant) with `VariableInstance` (process-instance quadrant). For doing this, we envision at least two approaches: discovering the relationships before populating the metamodel and populate all data together, including the relationships, and populating the metamodel with the raw data from both databases first, and find the relationships later. Although we are working on both approaches, we present here a simple example of the second one.

idElemInst	name	timestamp	Event	idCase	name	idVarInst	value	idVarDef	VarName	type
2	Register application	2020-05-20 16:26:22	Complete	2515	Students Mobility	1	Jane Doe	1	Name	string
2	Register application	2020-05-20 16:26:22	Complete	2515	Students Mobility	2	12223334	2	DocumentId	integer
2	Register application	2020-05-20 16:26:22	Complete	2515	Students Mobility	5	Erasmus	3	Program	string
4	Register application	2020-05-20 16:27:50	Complete	2515	Students Mobility	3	John Smith	1	Name	string
4	Register application	2020-05-20 16:27:50	Complete	2515	Students Mobility	4	23334445	2	DocumentId	integer
4	Register application	2020-05-20 16:27:50	Complete	2515	Students Mobility	6	Erasmus	3	Program	string

(a) Process data

idEntityInstance	timestamp	idEntity	name	idAttribute	name	type	idAttributeInstan...	value	idVarInstance
5	2020-05-20 16:26:25	2	Application	6	idApplication	INT	37	1	NULL
5	2020-05-20 16:26:25	2	Application	7	idProgram	INT	38	1	NULL
5	2020-05-20 16:26:25	2	Application	8	idStudent	INT	39	12223334	NULL
5	2020-05-20 16:26:25	2	Application	9	idState	INT	40	1	NULL
5	2020-05-20 16:26:25	2	Application	10	order	INT	41	NULL	NULL
5	2020-05-20 16:26:25	2	Application	11	notified	INT	42	NULL	NULL
5	2020-05-20 16:26:25	2	Application	12	date	DATETIME	43	2020-05-20 16:26:25	NULL
5	2020-05-20 16:26:25	2	Application	13	approved	INT	44	NULL	NULL
6	2020-05-20 16:27:55	2	Application	6	idApplication	INT	45	2	NULL
6	2020-05-20 16:27:55	2	Application	7	idProgram	INT	46	1	NULL
6	2020-05-20 16:27:55	2	Application	8	idStudent	INT	47	23334445	NULL
6	2020-05-20 16:27:55	2	Application	9	idState	INT	48	1	NULL
6	2020-05-20 16:27:55	2	Application	10	order	INT	49	NULL	NULL
6	2020-05-20 16:27:55	2	Application	11	notified	INT	50	NULL	NULL
6	2020-05-20 16:27:55	2	Application	12	date	DATETIME	51	2020-05-20 16:27:55	NULL
6	2020-05-20 16:27:55	2	Application	13	approved	INT	52	NULL	NULL

(b) Organizational data

Figure 16: Example of data of the integrated metamodel, as shown in Figure 10

idAttributeInstan...	value	timestamp	idAttribute	idEntityInstance	idVariableInsta...	idVarDef	timestamp	idCase
12	Jane Doe	2020-05-19 12:39:58	15	3	1	1	2020-05-20 16:26:22	2515
11	12223334	2020-05-19 12:39:58	14	3	2	2	2020-05-20 16:26:22	2515
39	12223334	2020-05-20 16:26:25	8	5	2	2	2020-05-20 16:26:22	2515
20	John Smith	2020-05-19 12:44:18	15	4	3	1	2020-05-20 16:27:50	2515
19	23334445	2020-05-19 12:44:18	14	4	4	2	2020-05-20 16:27:50	2515
47	23334445	2020-05-20 16:27:55	8	6	4	2	2020-05-20 16:27:50	2515
2	Erasmus	2020-05-19 12:48:05	2	1	5	3	2020-05-20 16:26:22	2515
2	Erasmus	2020-05-19 12:48:05	2	1	6	3	2020-05-20 16:27:50	2515
54	Susan Brown	2020-05-22 16:33:35	15	12	7	1	2020-05-22 16:42:14	2543
53	34445556	2020-05-22 16:33:04	14	12	8	2	2020-05-22 16:42:14	2543
63	34445556	2020-05-22 16:42:17	8	13	8	2	2020-05-22 16:42:14	2543
2	Erasmus	2020-05-19 12:48:05	2	1	9	3	2020-05-22 16:42:14	2543

(a) Matching data based on value

idAttributeInstan...	value	timestamp	idAttribute	idEntityInstance	idVariableInsta...	idVarDef	timestamp	idCase
39	12223334	2020-05-20 16:26:25	8	5	2	2	2020-05-20 16:26:22	2515
47	23334445	2020-05-20 16:27:55	8	6	4	2	2020-05-20 16:27:50	2515
63	34445556	2020-05-22 16:42:17	8	13	8	2	2020-05-22 16:42:14	2543

(b) Filtering (a) based on timestamps (i.e., `AttributeInstance.timestamp` $\geq$ `VariableInstance.timestamp`)

Figure 17: Example of matching data based on the value of `VariableInstance` and `AttributeInstance`

As an example of matching `AttributeInstance` and `VariableInstance` concepts, consider the "value" attribute defined for both of them. In Figure 17 we present the result of querying both elements in the integrated metamodel. The results in (a) show all instances from the organizational database that present the exact value of the "value" attribute. It can be noticed that not all of them are related to instances of the variables in the process execution. In (b), we filter the results based on the timestamps. The registers present timestamps with a minimum difference since the organizational database inserts are defined within the "Register applications" task in the process execution. Therefore, comparing both timestamps, we can filter the result, reducing it to two applications for the *2515* case and one application for the *2543* case, which corresponds to the identification document of the Student.

Applying the advanced matching algorithm presented in Subsection 3.2 yields the same results as shown in Figure 17 (b). It considers a rank of timestamps for the attribute's value registers in the organizational database, in which the variable's value register from the process engine database is included. The timestamp tolerance for the rank must be adequately defined, considering communication latency, computer clocks, and other variables that affect the timestamps of each database.

After integrating all pieces of information, we can perform more advanced queries regarding the process data and organizational data altogether. e.g., if some deadline requirement is established for the evaluation process, we can search for the careers affected by delayed cases. Timing information is taken from the process view, careers from the student information in the organizational view, and specific students who participate in such cases are related through the applications stored in process variables and database objects.

5 Related work

In [19] the relation between data science and process science through process mining is explored, and in [20] a process cube is defined to analyze and explore processes interactively based on a multidimensional view on event data, taking as input logs that already contain process data and organizational data.

Some works analyze the exploitation of database events as a source of information for event logs. In this context, in [21] the authors define an algorithm to merge event logs from different databases to analyze inter-organizational processes, and in [6] the authors propose building multiple viewpoint models from databases, providing a holistic view on a process, such that it is possible to automatically generate many possible event logs (one for each viewpoint of interest) for applying process mining techniques. All these works observe the problem from process mining and not on exploiting both sources of information altogether. Instead, we are taking an integrated view of execution data in the organization, providing a complete view of the organization's processes' real behavior.

In line with our interests, in [5] the authors propose a comprehensive integration of process and data information in a consistent and unified format through the definition of a metamodel. However, they focus on the extraction of read/write event logs from a database. Thus business-level activities are hidden, and the analysis is focused on the lower level of database operations. Besides the high-level structure of its metamodel is strongly related to ours, our metamodel considers a more business-oriented perspective, e.g., it involves other kinds of events related to business activities (e.g., service invocation) and process enactment information (e.g., users), it takes events from the life-cycle of element instances (e.g., user tasks) and not abstract events, and it considers typed process variables. The authors also present several scenarios for inferring metamodel data that is not already present from existing data (e.g., schema discovery from process events), which is different from our view. For example, the scenario we describe in Section 3 does not infer data about DB objects or process events from other existing data. Instead, it assumes the existence of both processes data and organizational data and focuses on discovering (rebuild) the existing relationship between them.

In [22], the authors discuss the problem mentioned above and define a conceptual framework for an in-depth exploration of process behavior. They combine information from three sources: the event log (business-level), the database (low level), and the transaction (redo) log. They also analyze different types of operations for mapping elements. In our case, we provide a uniform way of expressing all the information. We consider extended event logs from BPMS, e.g., including process variables not mapped to a DB object. The mapping operations need to be analyzed within our proposal.

Complimentary to these ideas, in [23, 24] the authors describe concrete matching techniques (manual, semantic semi-automatic, and semantic automatic) between process data and operational data that must be considered within our proposal. These techniques are later integrated into a business impact analysis framework based on a data warehouse. Unlike them, we focus on the integrated metamodel that captures all the data pieces, as the first point of exploitation, before building the data warehouse from the integrated data.

Finally, since our metamodel can be extended in the context of inter-organizational collaborative BPs, we can consider existing metamodels for BPs interactions via an interoperability platform [15], compliance requirements [25], and process and services evaluation and improvement [10].

6 Conclusions

Providing an organization with the evidence-based business intelligence necessary to improve their daily operation requires an integrated vision of the process data and organizational data. In this work, we have explored how this integrated vision can be achieved. We first analyzed the problem by identifying several scenarios for intra- and inter-organizational collaborative processes. We then provide an initial solution proposal by defining an integrated metamodel that could be the input for applying data and process mining techniques. We also show how this works from a practical perspective through an application example.

From a general perspective, we are currently working on many other aspects related to the integrated framework. e.g., extracting BPMS information from a unified data model [8], uncoupling the ETL process from a specific process engine model, and data quality from a methodological and technical point of view. We are also working on concrete dimensions for the DW perspective and their integration into ProM. Finally, as part of future work, we will implement a case study based on the Born Alive collaborative process presented in [15] currently enacted within the Uruguayan e-gov PDI. This case study proposes new requirements for the extension of the approach to consider inter-organizational scenarios. Moreover, it allows assessment of the framework, measuring its effectiveness, benefits, and limitations in real scenarios.

Acknowledgments

This work was supported by project "Minería de procesos y datos para la mejora de procesos en las organizaciones" funded by Comisión Sectorial de Investigación Científica (CSIC), Universidad de la República (UdelaR), Uruguay.

References

[1] M. Dumas, W. M. van der Aalst, and A. H. ter Hofstede, *Process-Aware Information Systems: Bridging People and Software through Process Technology.* John Wiley & Sons, Inc., 2005.

[2] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management, Second Edition.* Springer, 2018.

[3] M. Weske, *Business Process Management - Concepts, Languages, Architectures, Third Edition.* Springer, 2019.

[4] J. Chang, *Business Process Management Systems: Strategy and Implementation.* CRC Press, 2016.

[5] E. G. L. de Murillas, H. A. Reijers, and W. M. P. van der Aalst, "Connecting databases with process mining: a meta model and toolset," *Software and Systems Modeling*, vol. 18, no. 2, pp. 1209–1247, 2019.

[6] A. Berti and W. M. P. van der Aalst, "Extracting multiple viewpoint models from relational databases," *CoRR*, vol. abs/2001.02562, 2020.

[7] A. Delgado, A. Marotta, L. González, L. Tansini, and D. Calegari, "Towards a data science framework integrating process and data mining for organizational improvement," in *15th Intl. Conf. on Soft. Tech. (ICSOFT).* ScitePress, 2020, pp. 492–500.

[8] A. Delgado, D. Calegari, and A. Arrigoni, "Towards a generic BPMS user portal definition for the execution of business processes," *Electronic Notes in Theoretical Computer Science*, vol. 329, pp. 39 – 59, 2016.

[9] A. Delgado and D. Calegari, "A generic BPMS user portal for business processes execution interoperability," in *XLV Latin American Computing Conference, CLEI.* IEEE, 2019, pp. 1–10.

[10] A. Delgado, B. Weber, F. Ruiz, I. G. R. de Guzmán, and M. Piattini, "An integrated approach based on execution measures for the continuous improvement of business processes realized by services," *Inf. Softw. Technol.*, vol. 56, no. 2, pp. 134–162, 2014.

[11] A. Delgado and D. Calegari, "Towards a unified vision of business process and organizational data," in *XLVI Latin American Computing Conference, CLEI 2020.* IEEE, 2020, pp. 108–117.

[12] OMG, "Business Process Model and Notation (BPMN) version 2.0," OMG, Tech. Rep., 2011.

[13] A. Delgado, D. Calegari, P. Milanese, R. Falcon, and E. García, "A systematic approach for evaluating BPM systems: Case studies on open source and proprietary tools," in *Open Source Systems: Adoption and Impact - 11th IFIP WG 2.13 International Conference, OSS 2015, Proceedings*, ser. IFIP Advances in Information and Communication Technology, vol. 451. Springer, 2015, pp. 81–90.

[14] A. Delgado, L. González, and D. Calegari, "Towards setting up a collaborative environment to support collaborative business processes and services with social interactions," in *Service-Oriented Computing - ICSOC 2017 Workshops and Satellite Events, Revised Selected Papers*, ser. LNCS, vol. 10797. Springer, 2017, pp. 308–320.

[15] A. Delgado, D. Calegari, L. González, A. Montarnal, and F. Bénaben, "Towards a metamodel supporting e-government collaborative business processes management within a service-based interoperability platform," in *53rd Hawaii Intl. Conf. on System Sciences, (HICSS).* ScholarSpace, 2020, pp. 1–10.

[16] G. Daniel, G. Sunyé, and J. Cabot, "Mogwaï: A framework to handle complex queries on large models," in *Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1-3, 2016.* IEEE, 2016, pp. 1–12.

[17] P. Christen, *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, ser. Data-Centric Systems and Applications. Springer, 2012.

[18] T. Rademakers, *Activiti in Action. Executable business processes in BPMN 2.0.* Manning publications, 2012.

[19] W. M. P. van der Aalst and E. Damiani, "Processes meet big data: Connecting data science with process science," *IEEE Trans. Services Computing*, vol. 8, no. 6, pp. 810–819, 2015.

[20] W. M. P. van der Aalst, "Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining," in *Asia Pacific BPM - First Asia Pacific Conference, AP-BPM 2013, Beijing, China, August 29-30, 2013. Selected Papers*, ser. LNBIP, vol. 159. Springer, 2013, pp. 1–22.

[21] J. Claes and G. Poels, "Merging event logs for process mining: A rule based merging method and rule suggestion algorithm," *Expert Syst. Appl.*, vol. 41, no. 16, pp. 7291–7306, 2014.

[22] A. Tsoury, P. Soffer, and I. Reinhartz-Berger, "A conceptual framework for supporting deep exploration of business process behavior," in *Conceptual Modeling - 37th Intl. Conf., ER 2018, Proceedings*, ser. LNCS, vol. 11157. Springer, 2018, pp. 58–71.

[23] S. Radeschütz, B. Mitschang, and F. Leymann, "Matching of process data and operational data for a deep business analysis," in *Proc. of the 4th Intl. Conf. on Interoperability for Enterprise Software and Applications, IESA*. Springer, 2008, pp. 171–182.

[24] S. Radeschütz, H. Schwarz, and F. Niedermann, "Business impact analysis - a framework for a comprehensive analysis and optimization of business processes," *Com.Sci.Res.Dev.*, vol. 30, no. 1, pp. 69–86, 2015.

[25] L. González and R. Ruggia, "A comprehensive approach to compliance management in inter-organizational service integration platforms," in *13th Intl. Conf. on Soft. Tech (ICSOFT)*. SciTePress, 2018, pp. 722–730.