IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 6, 2021, accepted May 31, 2021, date of publication June 30, 2021, date of current version July 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3093356

gPROFIT: A Tool to Assist the Automatic Extraction of Business Knowledge From Legacy Information Systems

JULIAN A. GARCÍA-GARCÍA[1], C. AREVALO MALDONADO[1], AYMAN MEIDAN[1], ESTEBAN MORILLO-BARO[2], AND MARÍA JOSÉ ESCALONA[1]

[1]Department of Computer Languages and Systems, Escuela Técnica Superior de Ingeniería Informática, 41012 Sevilla, Spain
[2]Servinform, 41927 Seville, Spain

Corresponding author: Julian A. García-García (juliangg@us.es)

This research was funded by the ≪Agencia Estatal de Investigación (AEI)≫ through the Nico project (PID2019-105455GB-C31/AEI/10.13039/501100011033) of the Spanish Government's Ministry of Economy and Competitiveness. This research was also supported by the gPROFIT project, which was an R&D project carried out by the Servinform company in collaboration with our research group: Web Engineering And Early Testing group (IWT2 Group) of the University of Seville (Spain). The gPROFIT project was also funded by the Centre for the Development of Industrial Technology (CDTI) of the Spanish Government.

ABSTRACT Business digitization is a crucial strategy for business growth in the 21st century. Its benefits include improving business process automation, customer satisfaction, productivity, decision-making, turnover, and adaptation to market changes. However, digitization is not a trivial task. As a major paradigm and mindset shift, it involves a lot of effort within an organization and therefore requires commitment from employees and managers. This is especially critical in companies whose business processes are mostly reliant on legacy information systems (LIS), which are usually specialized and based on technological architectures that could be considered obsolete. The replacement of these systems by more recent, process-oriented technologies, the building up of employees' know-how and the continued use of outdated documentation are difficult, expensive tasks that hinder the initiation of continuous improvement processes in companies. This paper proposes techniques for finding and extracting process models from legacy databases. Specifically, it (*i*) lays the theoretical foundations of a model-driven framework for systematically extracting business process models (conform to standard BPMN notation) from LIS considering process time perspective, and (*ii*) proposes a technological tool called gPROFIT, which uses machine learning techniques to support that theoretical framework, facilitate its use in real environments and extract the business knowledge embedded in such legacy systems. The paper also presents proofs-of-concept showing how our proposal has been validated in several legacy systems.

INDEX TERMS Business digitization, model-driven engineering, process mining, legacy information systems, machine learning techniques.

I. INTRODUCTION

Globalization has caused a significant increase in business competition in all sectors, in both advanced and emerging economies [30]. Information and Communication Technologies (ICT) represent a key innovation for improving competitiveness and efficiency during the execution of organizations' business processes [7], [12], [19].

The Business Process Management (BPM) approach [53] is considered a strategic business advantage by many researchers because it contributes to optimize processes within organizations, as well as reduce costs and increase the quality of their services [32], [57], [60]. Furthermore, some authors have shown a significant increase in the business know-how when BPM strategies are implemented in an organization, which has also improved its internal performance [14], [29].

BPM is also applicable in software organizations, but it is necessary to use innovative techniques and solutions to

The associate editor coordinating the review of this manuscript and approving it for publication was Ricardo Colomo-Palacios.

implement these business strategies in these organizations because software processes are complex and are affected by new software development lifecycles, new technologies and large teams of multidisciplinary developers, among other. The management, control and measurement of software processes implies great costs due to these previously mentioned features [6], [31].

To achieve continuous and effective improvement, it is often necessary to integrate business processes with information systems. For example, information systems related to software production, customer management, supplier management, economic and financial management, among others. These systems can usually be process-oriented information systems [11], [27] or non-process-oriented information systems. The first one have well defined and integrated the process concept into their core software, whereas the second one are often Legacy Information Systems (LIS) that only store transaction states between running processes [3], [22], [26], [50].

The legacy system concept is associated with specialized information systems based on technological architectures that could be considered obsolete. The replacement of these systems by more recent technologies, the documentation of know-how or the initiation of continuous improvement processes are difficult and expensive tasks because of these features.

Nevertheless, all information systems have a common characteristic: their data persistence layer. This is usually represented by relational databases. In legacy systems, these databases are commonly referred by some authors [3], [39] Legacy Databases (LDB) and often store evidence related to certain business process execution perspectives [54].

Some authors have identified and defined these perspectives (flow control, information, time, organizational, operational, and cases) [47], [52], [53], [55], [56], and they have conclude that these perspectives are good datasources to extract and discover business process models.

In this context, this research paper aims to propose techniques to systematically and automatically extract and discover process models from legacy databases. Specifically, this paper (*i*) lays the theoretical foundations of a framework based on the Model-Driven Engineering (MDE) [48] paradigm to systematically extract business process models (conform to standard business process notations) from LIS considering the time perspective of the process, and (*ii*) proposes a technological tool to support our theoretical framework and facilitate its use in real environments.

On the one hand, the theoretical framework manages the time perspective of processes with different abstraction levels. First, a process definition metamodel is proposed to represent process models according to their time perspective. Later, a set of transformation rules are established to map the database metamodel to our process definition metamodel. When these transformations are executed, our framework generates process models according to BPMN-based business notations [5].

On the other hand, the supporting tool (denominated gPROFIT) has been designed to be easy to learn (in order to decrease the learning curve of users), use and understand by most people. In addition, gPROFIT includes machine learning mechanisms to achieve this goal and tries improving the user experience.

Finally, the paper is organized as follows. First, related works are presented in Section II. Later, Section III describes the methodology used to design our proposal for extracting process models from LIS using model-driven mechanisms and heuristics. Section IV details theoretical foundations of our proposal. Section V describes the technological solution of gPROFIT, which facilitates the user-friendly applicability of our model-driven theoretical framework in professional environments. Our proposal has been validated in several proofs of concept, which are presented in Section VI. Finally, Section VII presents the main conclusions and sets out some strategic considerations regarding future lines of research.

II. RELATED WORK

LIS are still the main information systems in all types of organizations. Today, a minority of LIS are Process-Aware (also called: Process-Oriented) Information Systems (PAIS) [11], [27], [47]. Process Mining [51], [52], [54], [55] may carry out business process generation from event logs existing in PAIS.

Process Mining obtains instances of processes from event log process traces. A process instance is a level-0 model (M0). Process Mining algorithms analyze many process traces and are therefore able to generate process models, which are level-1 models (M1) at a higher level of abstraction than process instances (M0). Non-PAIS lack the event log artifact, so, in this case, Process Mining is not a good approach to obtain process models. Databases hide knowledge related to processes, even in Non-PAIS LIS according to the work of Van Der Aalst [54], so they can be a source to extract dimensions of processes from these systems. This work establishes a theoretical basis that allows the use of databases to build event log records in systems that lack them, so that the generated records can feed Process Mining Techniques [51], [52], [54], [55].

In a study based on that framework [54], González López De Murillas *et al.* [10] use Redo-Logs as the source to generate an event log. The second initiative of González López De Murillas *et al.* [17] proposes a metamodel and tools to take advantage of the event log generated with Process Mining.

The Model-Driven Engineering (MDE) paradigm has become one of the most suitable approaches for the maintenance of old LIS, which are usually too complex and without updated documentation. MDE uses reverse and forward engineering techniques. OMG Model Driven Architecture (MDA) is one of the best known exponents of MDE. Architecture-Driven Modernization (ADM) [36], which is an OMG's MDA-based proposal, includes reverse and forward engineering roadmaps from an old source system to a new target one. Source LIS artifacts may be: *(i)* source

code; *(ii)* graphical user interfaces (GUI); or *(iii)* databases. Among other techniques, ADM uses Abstract Syntax Tree Metamodels (ASTM) [36], [37] and Knowledge Discovery Metamodels (KDM) [36], [41] to generate LIS models from source artifacts.

We were interested in reverse engineering roadmaps that may help us to discover processes by focusing on databases. Since ADM does not focus on process discovery field, we have needed to explore other literature regarding database reverse engineering as well as specific proposals with the aim to discover processes from Non-PAIS.

Much research has been carried out into database reverse engineering, but the works we selected were those most relevant to our own study.

The first work worthy of mention is by Cleve *et al.* [8], who propose data reverse engineering using System Dependency Graphs (SDG) that analyze Structured Query Language (SQL) embedded in application code to propose an alternative database schema. Another relevant work, despite not being oriented towards processes, was that of Cosentino and Martínez [9], who extract UML classes and OCL (Object Constraint Language) [24] rules from tables and triggers. Finally, Zanoni *et al.* [61] propose pattern detection for conceptual schema recovery in data-intensive systems. The approaches collect all types of artifacts existing in databases of Non-PAIS, although they do not face the dimensions of the processes. All these works are not oriented to processes. There are almost nonexistent experimental cases that extract processes from LIS.

Concerning with business process discovering from Non-PAIS LIS, we have selected some research works by Pérez-Castillo *et al.*: *(i)* those, which propose Modernization Approach for Recovering Business Processes from Legacy Systems (MARBLE [40], [42], [44]) as a framework that extends OMG ADM [48]; and *(ii)* Pérez-Castillo *et al.* [43], who propose recovering web services from databases. Both studies propose different steps. Each step relies on metamodeling to map artifacts between models of different levels of abstraction. The goal is to obtain approximations to the user's business processes that are composed of transaction traces that have been recorded in databases. The authors highlighted the use of relational database sentences to propose new relational database schemas, adopting ideas like those included in the aforementioned work by Cleve *et al.* [8].

The aforementioned work, regarding business process discovering from Non-PAISs, only generate some aspects of processes. Other authors' approaches, which use SDG [8], ASTM [36], [37] or KDM [36], [41], gather all kinds of Non-PAISs database artifacts, although they do not face the dimensions of the processes as a heuristic basis for generating them. There are almost non existent experimental cases that utilize these proposals to extract processes. Additionally, the results obtained do not go beyond deriving conceptual database schemes or poor approximations to real processes. Nonetheless, as results may appear to be poor in the eyes of business experts, they are not widespread.

In previous work, Arevalo *et al.* [1], Maldonado *et al.* [28] focused on the time dimension of processes to address database reverse engineering. We have defined a taxonomy of time rules, expressed by means of UML and OCL to enrich a minimal pivot process metamodel, which may act as the core that commonly may exist in many PAIS. The work [28] includes a proposal to extend BPMN with the time dimension. Arevalo Maldonado *et al.* [2] develop a framework that extracts business processes from project plans, which are included in software project management systems, such a Microsoft Project, as an example of LIS commonly used by Information Technology Companies. The pivot process metamodel, which is included in [28], was used as a baseline to transform database rules into process models of levels M0 (instances) and M1 (models), by using ASTM [37] to analyze database schemas as a source for the reverse engineering process. In this way, they propose a generic ASTM (Generic-ASTM (GASTM)), which is common to any standard SQL database management system, and a specific one (Specific-ASTM (SASTM)), in the specific case study for Microsoft SQL*Server that supports MS*Project. They apply their framework to a case study, comparing extracted processes with business processes, which are manually designed by experts.

In this paper, we aim to extend the proposal to more legacy system types. We depict the specific roadmaps developed for different case studies, taking legacy databases as source systems, and proposing metamodels and new MDA-based algorithms to transform evidences from process execution traces into their corresponding business process representation. We have also created the gProfit, based on Enterprise Architect tool, to automatically run these algorithms and analyze the results. Therefore, we hope that expert users of the case studies will appreciate the value that the research initiative brings. Our proposal faces process dimensions that are hidden in legacy databases [54]. By looking for process execution traces hidden in such databases [54], and capturing process dimensions (initially the time dimension, but eventually other dimensions such as organization, resources, data and cases), we will be able to obtain more fruitful results.

In comparison with some approaches cited above, which just use different transformation steps between ASTM and KDM, our heuristics do not use KDM and initially centers on Temporal Dimension of processes that may be scattered in databases, which otherwise would be wasted for BPM purposes. To the best of our knowledge, we have not found approaches in the literature that focus on process dimensions related metamodels as heuristic criteria for designing reverse engineering algorithms that can extract processes from legacy systems.

III. MATERIAL AND METHODS: SCIENTIFIC METHODOLOGY

After establishing the context and the motivation underlying this study, rigorous scientific methodology was applied to

produce a solution capable of answering the following hypothesis: ≪*It is possible to successfully facilitate the continuous improvement of companies and reduce their costs by combining model-driven mechanisms and machine learning techniques to extract business process models from Legacy Information Systems (LIS)*≫. The specific method used was Design Science Research Methodology (DSRM) [38]: a very widely accepted, increasingly important paradigm for research into information software systems [21]. This methodology rigorously defines five stages:

1) **Problem identification and motivation**. This phase allowed us to identify the problem our proposal was trying to solve. For this purpose, we first conducted a review of works related to our hypotheses that have been published in scientific literature: that is to say, other proposals addressing specific aspects of process model extraction from LIS. Section II identifies and discusses those related works.
2) **Objective**. As mentioned above, this paper aims to define a model-driven, tool-supported framework that facilitates business digitization. Many companies still use LIS, which, although they are not process-oriented systems, store activity or task data with time correlations. Extracting and modelling this business knowledge is essential for automation and digitization. Our proposed framework therefore aims to establish guidelines for extracting business process models from LIS. It also uses machine learning techniques to improve the automatic extraction of these models.
3) **Design and development**. This phase involved two steps: *(i)* designing a novel metamodel and its associated semantic constraints, together with the MDE-based theoretical bases of our framework; and *(ii)* developing gPROFIT, the tool which would support the application of our metamodel and automate parts of its theoretical framework.
4) **Demonstration**. This phase comprised the development of the gPROFIT tool, which effectively showed that the theoretical MDE mechanisms designed in this paper were amenable to extracting a business process model from LIS. It should be mentioned that our software solution is platform-independent, since it was developed using standards like UML (Unified Modelling Language) [45] and ISO/IEC TR:24744 [23], which include guidelines for improving consistency and uniformity in the definition of process models.
5) **Evaluation**. Section VI presents the evaluation of our proposal with several business proofs-of-concept were used to validate our proposal; more specifically, three LIS were used to extract the business model process.

IV. THEORETICAL FOUNDATIONS OF THE PROPOSAL

This section describes the theoretical foundations of our proposal for extracting business models from LIS. The theoretical model-driven framework and its mechanisms are presented in Section IV-A. Later, Section IV-B briefly describes an abstract syntax language (metamodel) that supports the time perspective definition of process models; and Section IV-C presents the transformation procedure for obtaining business process models from LIS.

A. THEORETICAL MODEL-DRIVEN FRAMEWORK

The execution of business processes typically generates organizational business knowledge. This knowledge is usually hidden and concentrated in the data structures of the organizational systems, the most stable level of such systems. These structures are Legacy DataBases (LDB) and are usually based on Legacy and Relational DataBase Models (LRDBM). Although LIS are not process-oriented systems, they usually store states resulting from the execution of organizational processes. Those states can later be reinterpreted [54], allowing business process models to be obtained from them.

There are currently many Business Process Modelling Languages (BPMLs) [15] that could be used to model business processes. Any published BPML could be considered as a target business process notation within our proposal, but there is one major handicap: the target notation must support the time semantics of processes, and this feature is not supported by all BPMLs.

To mitigate and resolve this problem, our proposal defines an intermediate metamodel as a means of obtaining greater interoperability and independence from specific LRDBMs and BPMLs. This intermediate metamodel acts as a pivot notation between LRDBMs and BPMLs: if any of them are replaced by another model, it will not be necessary to reformulate all the model-driven mechanisms (in this case, it would only be necessary to update the LRDBM/BPML transformations with the pivot metamodel).

As stated, then, this paper proposes a theoretical model-driven architecture supported by an MDA infrastructure to achieve our goal. The architecture, shown in Figure 1, considers a set of metamodels at different levels of abstraction and a two-stage transformation procedure.

The ***Platform-Specific Models (PSMs)*** at the bottom of Figure 1 define the technological features (such as the relational data scheme) of the source software platform. The relational data scheme is mainly defined in two kinds of PSM Metamodels (PSMMs) or abstract syntax languages: generic metamodels (such as the Generic Abstract Syntax Tree Metamodel or GASTM [37]), which make it possible to define the table structures of any relational database; and specific metamodels (such as the Specific Abstract Syntax Tree Metamodel; SASTM [37]), which make it possible to model complex structures within database engines (for example, data tables, constraints, and triggers based on procedural coding languages).

The ***Platform-Independent Models (PIMs)*** shown at the intermediate level of Figure 1 represent software system models that are independent of the specific technological platform used to implement them. Our pivot metamodel (previously mentioned) is located at this level and it is called the PIM UML Pivot (PIM-up) metamodel. The PIM-up

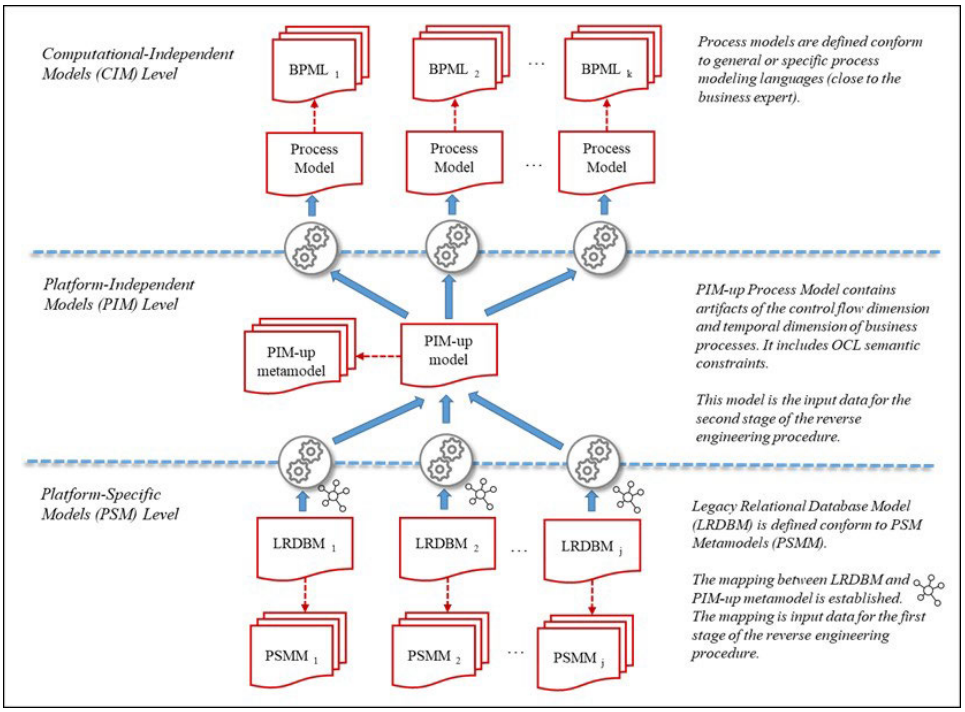


FIGURE 1. Theoretical model-driven proposal for extracting process models from legacy information systems.

metamodel, described in Section IV-B, contains the minimum concepts and relationships that are necessary to represent business processes considering the time dimension.

This minimalism may be useful when applying and validating gPROFIT and its metamodel (the PIM-up metamodel) in different business environments. Once the PIM-up metamodel is validated, it could be extended to support more semantic process dimensions [52] (such as resource, organizational, case, or data flow dimensions), since our proposal supports UML-based extension mechanisms. In fact, our research group has experience in and is currently working on transferring scientific knowledge from the University to companies in different business contexts. These opportunities and capabilities provide us with valuable feedback for extending our proposal and considering new concepts in process definition. The PIM-up metamodel also offers a simple, flexible language with several goals: *(i)* to facilitate the application of MDE-based mechanisms to gPROFIT; *(ii)* to reduce users' cognitive overload and decrease their learning curve when using gPROFIT; and *(iii)* to allow the rapid application of gPROFIT in real environments. With regard to usability, there are studies based on well-established theories [18] that have analysed the cognitive overload of several languages [16], [33], [34]. These works corroborate that complex modelling languages increase user's cognitive overload and make the defined models difficult to understand. This usability aspect has a major influence on users' efficiency when using a language.

The first stage of the transformation or reverse engineering procedure takes place at the bottom of the intermediate level in Figure 1. This first stage is based on mapping PSM level data structures with concepts belonging to our PIM-up metamodel. Section IV-C describes this stage of the reverse engineering procedure in our proposal.

Finally, the ***Computational-Independent Models (CIMs)*** at the top of Figure 1 are considered the most abstract models. These models are used to represent models close to the knowledge of the business experts. CIM are the target models in our proposal and refer to business process models that could be defined in accordance with any existing BPML [15] (for example, SPEM2.0 [49], BPMN [5] or UML Activity Diagram [13], among others).

The second stage of the reverse engineering procedure takes place at the bottom of this upper level. This second stage is based on the transformation of each concept belonging to the PIM-up metamodel to generate the homologous concept in the chosen business notation that is familiar to the business expert. This paper proposes using BPMN2.0 to instantiate business process models at the CIM level because it is a well-known general-purpose notation capable of defining business processes in any context [46].

B. METAMODEL TO SUPPORT BUSINESS PROCESS TIME PERSPECTIVE

This section introduces our extensible, highly semantic MOF-compliant metamodel, a tool which makes it possible to obtain business process models (CIM level) that consider process time dimension from the data persistence model of each LIS (PSM level). As mentioned above, we called this metamodel the PIM UML Pivot (PIM-up) metamodel and it defines the minimum concepts and relationships needed to represent the time dimension of business processes. This

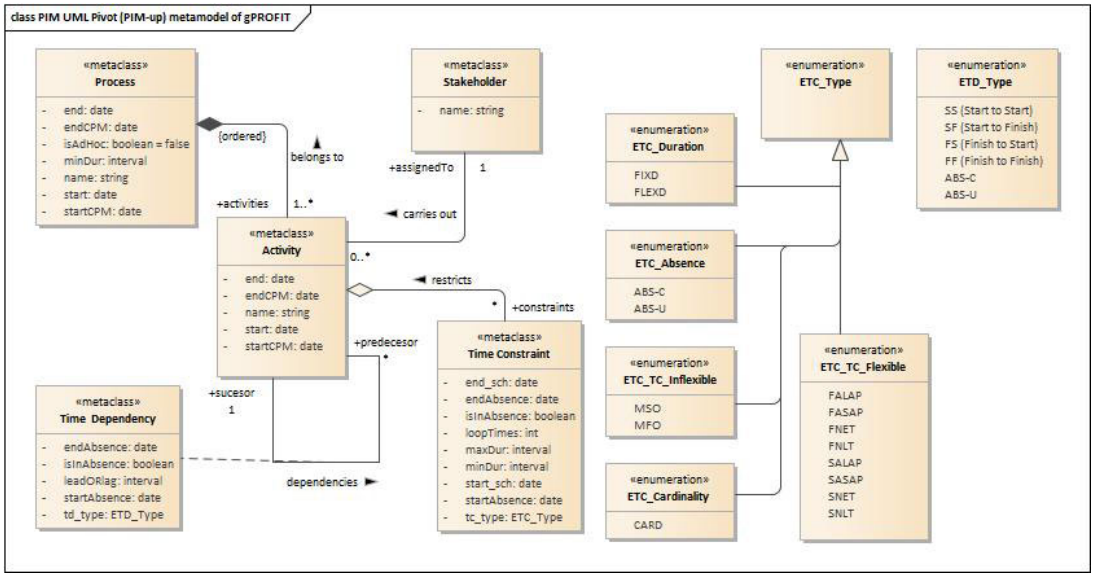


FIGURE 2. PIM UML Pivot (PIM-up) metamodel of gPROFIT to support time perspective.

dimension, which refers to the time events that occur during the execution of business processes, is analysed in detail in [1].

The PIM-up metamodel is shown in Figure 2. At this point it should be clarified that the syntax used was not enough to semantically define the PIM-up metamodel. OCL was therefore used to add formal constraints that would validateprocess models. The scope of this paper does not allow us to explain all these constraints, but they are described in [28]. Notwithstanding, and for illustrative purposes, several of them are explained in detail below.

The PIM-up metamodel constitutes the theoretical foundation of our proposal and contains the following metaclasses: ≪*Activity*≫, ≪*Process*≫, ≪*Stakeholder*≫, ≪*Time Constraint*≫ and ≪*Time Dependency*≫. Time constraints and time dependencies are also typified according to two taxonomies: ≪*ETD_Type enumeration*≫[1] and ≪*ETC_Type enumeration*≫,[2] respectively.

[1]The ≪*ETD_Type enumeration*≫ expresses the precedence relationships between the events associated with each activity belonging to the business process. These relationships are typified according to the following values: Start to Start (SS); Start to Finish (SF); Finish to Start (FS); Finish to Finish (FF). Furthermore, two kinds of absence constraints are considered: unconditional (ABS-U) and conditional in an interval (ABS-C).

[2]The ≪*ETC_Type enumeration*≫ establishes a taxonomy of time constraint based on the generalization of the following types: *(i)* ≪*ETC_TC_Inflexible enumeration*≫, which establishes fixed start and end date for the activity (''must start/finish on'' rules; MSO/MFO); *(ii)* ≪*ETC_Duration enumeration*≫, which establishes the flexible (FLEXD) or fixed (FIXD) duration; *(iii)* ≪*ETC_TC_Flexible enumeration*≫, which defines values for ''start/finish as soon/late as possible'' rules (SASAP, SALAP, FASAP, FALAP), ''start/finish no earlier/later than'' rules (SNET, SNLT, FNET, FNLT); *(iv)* ≪*ETC_Cardinality enumeration*≫, which defines the CARD value on repetitions of an activity; (v) ≪ETC_Absence≫, which defines two kinds of absence constraints (unconditional (ABS-U) and conditional in an interval (ABS-C)).

The main metaclass in the PIM-up metamodel is the ≪*Process metaclass*≫, which represents a business process aimed at achieving a specific business objective. A business process has the following attributes: name; start and end dates; and attributes calculated with the Critical Path Method (CPM) [25], i.e., ≪*startCPM*≫, ≪*endCPM*≫ and ≪*minDur*≫ (minimum duration of the process for its critical path) attributes. A process also contains a set [1..*] of ordered actions, which were modelled with the ≪*Activity metaclass*≫ in the PIM-up metamodel.

The ≪*Activity metaclass*≫ is carried out by an actor (conceptually represented by the ≪*Stakeholder metaclass*≫) and has the following attributes: name; start and end dates; and attributes calculated with the CPM (i.e., ≪*startCPM*≫, ≪*endCPM*≫ attributes).

Regarding the above mentioned concepts, it is possible to establish a lifecycle associated with activities and processes, and, more specifically, with the critical path and the time dimension of those two elements. This lifecycle has four states (≪*Inactive*≫, ≪*Scheduled*≫, ≪*Active*≫, and ≪*Closed*≫), the transitions of which are formalized using a state machine (c.f. Figure 3). The first state means that the time attributes of the activity or process instance (i.e., ≪*startCPM*≫, ≪*endCPM*≫, ≪*minDur*≫ – for processes –, ≪*start*≫ and ≪*end*≫) are not initialized. When these instances are scheduled, their attributes related to the CPM method are initialized. At this point, the instances are in the ≪Scheduled≫ state and will remain in this state until they receive their startup token. When the token is received and its input data is available, the activity or process is activated to carry out its objective (≪*Active*≫ state). At this time, the ≪*start*≫ (real start of the task) and ≪*startCPM*≫ attributes are equal. Subsequently, two transitions are possible: *(i)* to carry out a new planning phase affecting the process and its

set of activities; or *(ii)* to finalize the instance (which involves updating its current state to ≪*Closed*≫ and equalizing the ≪*end*≫ and ≪*endCPM*≫ attributes.

The syntax used to model the ≪*Process metaclass*≫ and the ≪*Activity metaclass*≫ was not enough to define the semantics of the transitions of the lifecycle shown in Figure 3. OCL was therefore used to define the semantic constraints of these transitions and control the instantiation of these metaclasses. Algorithm 1 shows some OCL constraints associated with the ≪*Process metaclass*≫.

Algorithm 1 OCL Constraints Associated With the Lifecycle of the ≪*Process Metaclass*≫

```
1 context Process inv: // inactive
   startCPM.oclIsUndefined() AND
   endCPM.oclIsUndefined() AND
   start.oclIsUndefined() AND
   end.oclIsUndefined() AND minDur.oclIsUndefined()

2 context Process inv: // scheduled
   startCPM.oclIsUndefined() AND
   endCPM.oclIsUndefined() AND
   minDur.oclIsUndefined() AND
   start.oclIsUndefined() AND end.oclIsUndefined()

3 context Process inv: // active
   startCPM.oclIsUndefined() AND
   endCPM.oclIsUndefined() AND
   minDur.oclIsUndefined() AND
   start.oclIsUndefined() AND
   end.oclIsUndefined() AND start=startCPM

4 context Process inv: // closed
   startCPM.oclIsUndefined() AND
   endCPM.oclIsUndefined() AND
   minDur.oclIsUndefined() AND
   start.oclIsUndefined() AND
   end.oclIsUndefined() AND
   start=startCPM AND end=endCPM
```

The ≪*Activity metaclass*≫ is also related to itself to model time constraints and precedence relations between instances of this metaclass as shown in Figure 2. The two relationships are identified respectively with the ≪*restricts*≫ and ≪*dependencies*≫ composition associations in Figure 2.

Time dependencies are represented by the ≪*Time Dependency metaclass*≫ and refer to precedence relations between two activities (≪*predecessor role*≫ and ≪*successor role*≫ in Figure 2). The main attributes of this metaclass allow its typology to be established according to ≪*ETD_Type enumeration*≫ (i.e., the ≪*td_type*≫ attribute; c.f. Figure 2), the time interval between the successor and predecessor activities (i.e., the ≪*leadORlag*≫ attribute), and the absence dependency and its associated

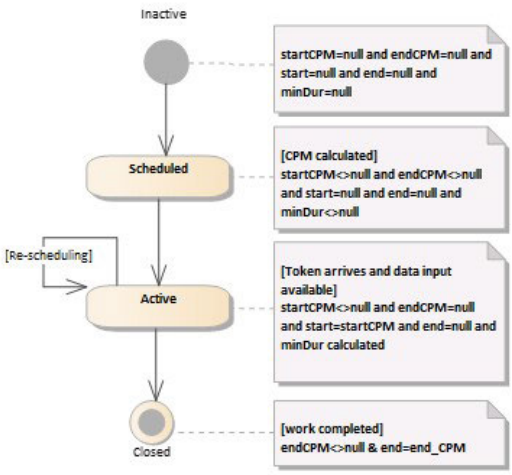


FIGURE 3. Lifecycle associated with the ≪*Process metaclass*≫ and the ≪*Activity metaclass*≫.

time interval (i.e., the ≪*isInAbsence*≫, ≪*startAbsence*≫ and ≪*endAbsence*≫ attributes).

The successor and predecessor roles also allow semantic invariants/constraints to be defined, which complement the syntax used to model the PIM-up metamodel [28]. These constraints can be also specified with time leads or lags (≪*leadORlag*≫ attribute) between events that synchronize the successor and predecessor activity. Due to space constraints, it is beyond the scope of this paper to explain all the OCL invariants, but Algorithm 2 shows some of them. The first invariant controls the synchronization of two activities related to a Start to Start (SS) type time dependency. The second invariant ensures that the successor activity does not receive the control flow (even if its token arrives) once the predecessor activity has been executed. This situation causes an unconditional absence state (ABS-U) in the successor activity. The third invariant is a conditioned absence (ABS-C), meaning that the successor activity can only receive the control flow within a set time interval *[startAbsence, endAbsence]*.

Time constraints are represented by the ≪*Time Constraint metaclass*≫, whose main attributes are: *(i)* time ranges of scheduled dates (≪*start_sch*≫ and ≪*end_sch*≫ attributes); *(ii)* time ranges related to duration (≪*minDur*≫, ≪*maxDur*≫ and ≪*loopTimes*≫ attributes), which allow fixed (FIXD) or flexible (FLEXD) durations to be specified for the execution of the activity; *(iii)* ≪*loopTimes*≫, which limits the maximum number of instances of an activity that is executed in a loop; *(iv)* ≪*isInAbsence*≫, ≪*startAbsence*≫ and ≪*endAbsence*≫, referring to the absence dependency (ABS-U) and the definition of its time interval (ABS-C), respectively; and *(v)* ≪*tc_type*≫, which means the type of time constraint in accordance with ≪*ETC_Type enumeration*≫ (c.f. Figure 2).

In this case, the syntax used to model the relationships between ≪*Activity metaclass*≫ and ≪*Time Constraint*

Algorithm 2 OCL Constraints Associated With the ≪*Time Dependency Metaclass*≫

```
1  Context Temporal_Dependency inv: // td_type: SS (Start to Start)
2    self.td_type = 'SS' implies self.predecessor → select
3      (P: Activity | self.successor.startCPM <= P.startCPM + leadORlag) → notEmpty()

4  Context Temporal_Dependency inv: // td_type: ABS-U
5    (self.td_type = 'ABS-U' AND self.predecessor.startCPM.oclIsUndefined()) implies
6      (self.startAbsence.oclIsUndefined() AND self.endAbsence.oclIsUndefined() and self.isInAbsence = true)

7  Context Temporal_Dependency inv: // td_type: ABS-C
8    (self.td_type = 'ABS-C' AND self.predecessor.startCPM.oclIsUndefined() AND
9      self.sucessor.startCPM.oclIsUndefined() AND self.sucessor.endCPM.oclIsUndefined())
10   implies
11     (self.startAbsence.oclIsUndefined() AND self.endAbsence.oclIsUndefined() AND
12       ((self.sucessor.startCPM + self.leadORlag >= self.startAbsence) OR
13         (self.sucessor.endCPM + self.leadORlag <= self.endtAbsence)) implies self.isInAbsence = true)
```

metaclass≫ in the gPROFIT metamodel was not enough to fully represent all their semantics. OCL invariants were therefore defined to complement this syntax, considering each typology defined by the ≪*ETC_Type enumeration*≫ (Figure 2). Due to space constraints, it is not possible to explain all these OCL invariants in this paper, but some of them are shown in Algorithm 3. The first invariant in Algorithm 3 refers to the fixed duration of the activity. It ensures that the activity's duration coincides with the scheduled duration (*minDur* = *maxDur*), which must also coincide with the activity's execution interval. The second semantic invariant is similar to the first, but this expression sets a flexible duration within a time interval. The third invariant defines two situations related to the type of cardinality (CARD). This invariant ensures that the time duration of an activity's loop execution (the ≪*loopTimes*≫ attribute) is within the activity's global execution interval (considering real and scheduled magnitudes).

C. PROCESS EXTRACTION FROM LEGACY SYSTEMS

As mentioned above, Section IV-A proposes a MDE architecture for the reverse engineering process for transforming the time dimension of a legacy database into a business model close to the business expert.

However, before running this model extraction procedure, the process engineer or/and business expert have to manually study and analyze the LIS data model to understand its structure and identify the relationships (mapping) between metaclasses/meta-attributes of the PIM-upl metamodel (c.f. Figure 2) and the appropriate entities belonging to the LDBM. This mapping consists of: (*i*) establishing the mapping of metaclasses (activities and processes) and their groupings (for example, each activity belongs to a process or sub-process); (*ii*) assigning the specific properties (meta-attributes) of each metaclass; (*iii*) detecting time dependencies between activities according to the rules taxonomy (c.f. Figure 2); and (*iv*) assigning ad-hoc processes to group together isolated activities or activities without dependencies.

In this context, two types of transformation rules were proposed to achieve the objective mentioned at the beginning of this section. Space restrictions make impossible to explain both transformations in detail here, but these ones are briefly explained below.

1) **Transformation rule to extract PIM-upl Model from Legacy DataBase Model (PSM → PIM transformation)**. This transformation makes it possible to capture the time dimension of a legacy database (PSM level) and obtain PIM models (in accordance with the PIM-upl metamodel and OCL time rules). Algorithm 4 describes this transformation using pseudocode. To simplify this notation, parameters are provided as references in each function.
On the one hand, the input data for this transformation are scheduled entities (≪*setProjects*≫ *parameter*) stored in the LDBM (c.f. Algorithm 4, line 2). The data type of this parameter has been represented as a set of tuples of the entity of the LDBM that is mapped with ≪*Process metaclass*≫ of the PIM-upl metamodel (c.f. Figure 2). On the other hand, the output data is a ≪*Process instance*≫ (≪*BP*≫ *parameter*) of the PIM-upl metamodel (c.f. Algorithm 4, line 3).
First, the algorithm generates a ≪*Process instance*≫ per project (c.f. Algorithm 4, line 11), in accordance with the ≪*Process metaclass*≫ of the PIM-upl metamodel.
Once this instance has been generated, the algorithm generates the activities belonging to the business process (c.f. Algorithm 4, lines 18-32). It does this by analysing each data tuple in the LDBM that is mapped with the ≪*Activity metaclass*≫ of the PIM-upl metamodel. After conducting this analysis, the ≪*Activity instance*≫ in accordance with the PIM-upl metamodel is generated and associated

Algorithm 3 OCL Constraints Associated With the ≪*Time Constraint Metaclass*≫

```
1  Context Temporal_constraint inv: // tc_type: FIXD
2    self.tc_type = 'FIXD' implies self.tc → select
3      ((self.endCPM - self.startCPM) = (self.end_sch - self.start_sch) AND
4       (self.endCPM - self.startCPM) = self.minDur AND (self.endCPM - self.startCPM) = self.maxDur) → notEmpty()

5  Context Temporal_constraint inv: // tc_type: FLEXD
6    self.tc_type = 'FLEXD' implies self.tc → select
7      ((self.endCPM - self.startCPM) >= self.minDur AND (self.endCPM - self.startCPM) <= self.maxDur AND
8       (self.end_sch - self.start_sch) >= self.minDur AND (self.end_sch - self.start_sch) <= self.maxDur) → notEmpty()

9  Context Temporal_constraint inv: // tc_type: CARD (1)
10   (self.tc_type = 'CARD') implies self.tc → select
11     ((self.startCPM + self.loopTimes * self.minDur <= endCPM) AND
12      (self.endCPM <= self.startCPM + self.loopTimes - self.maxDur)) → notEmpty()

13 Context Temporal_constraint inv: // tc_type: CARD (2)
14   (self.tc_type = 'CARD' AND self.start_sch → notEmpty() AND self.end_sch → notEmpty()) implies
15     (self.start_sch + self.loopTimes * self.minDur <= self.end_sch) AND
16      (self.end_sch <= self.start_sch + self.loopTimes * self.maxDur)) → notEmpty()
```

with the previously generated ≪*Process instance*≫ (c.f. Algorithm 4, lines 20-21).
Moreover, time constraints and time dependencies are generated for each activity.
For time constraints, a ≪*Time_Constraint instance*≫ is first generated to establish the (fixed or flexible) duration of the activity according to its scheduled duration in the LDBM (c.f. Algorithm 4, line 22). Later, ≪*Time_Constraint instances*≫ are then generated to establish start and end events for each activity, considering whether its duration is fixed (MSON or MFON) or flexible (ASAP, ALAP, NET or NLT) (c.f. Algorithm 4, lines 23-27).
For time dependencies, the algorithm analyses the precedence relationships of each ≪*Activity instance*≫ and generates a ≪*Time_Dependency instance*≫ (in accordance with the PIM-upl metamodel) for each predecessor task. This time dependency is associated with ≪*Activity instance*≫ and its type is established according to the time dependency taxonomy (FF, FS, SF or SS) (c.f. Algorithm 4, lines 28-32).
Finally, Algorithm 4 finishes merging all the instances of the business process together in a single model (c.f. line 34).

2) **Transformation rule to extract BPMN model from PIM-upl Model (PIM → CIM transformation)**. This transformation makes possible to obtain CIM models that are close to the business experts. For this purpose, this paper proposes obtaining BPMN models. Algorithm 5 describes the transformation using pseudocode. The input and output data of this transformation are, respectively, a process model in accordance with the PIM-upl metamodel and a process model in accordance with BPMN2.0. The first step in the transformation is to generate a ≪*BPMN::Process instance*≫ from the ≪*PIM-upl::Process instance*≫ (c.f. Algorithm 5, line 13). A Pool (≪*BPMN::laneSet metaclass*≫) is also generated and associated with the ≪*BPMN::Process instance*≫ (c.f. Algorithm 5, line 14).
Once both instances have been generated, each activity in the PIM-upl model is transformed into a homologous activity in accordance with BPMN2.0. More specifically, one ≪*BPMN::Activity instance*≫ per ≪*PIM-upl::Activity instance*≫ is generated (c.f. Algorithm 5, lines 16-40). If the ≪*PIM-upl::Activity instance*≫ is linked to a ≪*PIM-upl::Stakeholder instance*≫, this one is transformed into a ≪*BPMN::Lane instance*≫ (c.f. Algorithm 5, lines 20-27) which is also associated with the ≪*BPMN::Activity instance*≫ and the ≪*BPMN::Pool instance*≫ (both of which were generated previously).
Finally, the time constraints and time dependencies of the ≪*PIM-upl::Activity instance*≫ are transformed into instances of ≪*BPMN::Time_Constraint metaclass*≫ and ≪*BPMN::Time_Dependency metaclass*≫, respectively (c.f. Algorithm 5, line 29-39). These instances are also linked to the ≪*BPMN:: Activity instance*≫.

V. SUPPORTING TOOL AND TECHNOLOGICAL SOLUTION

This section describes the technological solution gPROFIT, which facilitates the user-friendly applicability of our theoretical model-driven framework in professional environments. Section V-A describes the gPROFIT technological

Algorithm 4 Transformation Rule to Extract PIM-Upl Model From Legacy DataBase Model (PSM → PIM Transformation) Considering the Time Dimension

```
1  function transformationPSM2PIM
   (Set(LDBM::ProjectTuple) setProjects) return
   PIM-uplM::Process
2  {
3    Set setBPs = {};
4    PIM-uplM::Process bp_i;
5    PIM-uplM::Activity actv_i;
6    PIM-uplM::TimeConstraint tc_i;
7    PIM-uplM::TimeDependency td_i;
8    LDBM::ProjectTuple project_i;
9    LDBM::TaskTuple T_i;
10   LDBM::TaskTuple PreviousTask_i;
11   LDBM::ConstraintTuple constraint_i;
12
13   for (project_i ∈ setProjects)
14   {
15     bp_i = createProcess (project_i)
16
17     for (T_i ∈ project_i)
18     {
19       actv_i = createActivity (bp_i, T_i);
20       includeActivity (bp_i, actv_i);
21       setDurationConstraint (actv_i);
22
23       for (Constraint_i ∈ T_i)
24       {
25         tc_i = createTimeConstraint (Constraint_i);
26         associateTimeConstraint (tc_i, A_i);
27       }
28
29       for (PreviousTask_i ∈ getPreviousTasks (T_i))
30       {
31         createActivity (BP_i, PreviousTask_i, A_j);
32         createTimeDependency (actv_i, A_j);
33       }
34     }
35   }
36   return mergeInstances (setBPs);
37 }
```

Algorithm 5 Transformation Rule to Extract BPMN Model From PIM-Upl Model (PIM → CIM Transformation) Considering the Time Dimension

```
1  function transformationPIM2CIM (PIM-uplM::Process
   bp) return BPMN::Process
2  {
3    PIM-uplM::Activity actv;
4    PIM-uplM::TimeConstraint tc;
5    PIM-uplM::TimeDependency td;
6    BPMN::Process bp_bpmn;
7    BPMN::Activity actv_bpmn;
8    BPMN::Lane lane;
9    BPMN::Pool pool;
10   BPMN::TimeConstraint tc_bpmn;
11   BPMN::TimeDependency td_bpmn;
12
13   bp_bpmn = createProcessBPMN (bp);
14   pool = createPoolBPMN (bp_bpmn);
15
16   for (actv ∈ bp.activities)
17   {
18     actv_bpmn = createActivityBPMN (actv);
19     associateActivity2Process (actv_bpmn, bp_bpmn);
20     lane = getAssociatedLane (actv.stakeholder);
21
22     if (not existLane (lane))
23     {
24       lane = createLaneBPMN (actv.stakeholder);
25       associateLane2Pool (lane, pool);
26     }
27     associateActivity2Lane (actv_bpmn, lane);
28
29     for (tc ∈ actv.constraints)
30     {
31       tc_bpmn = createTCBPMN (tc);
32       associateTC2Activity (tc_bpmn, actv_bpmn);
33     }
34
35     for (td ∈ actv.dependencies)
36     {
37       td_bpmn = createTDBPMN (td);
38       associateTD2Activity (td_bpmn, actv_bpmn);
39     }
40   }
41   return bp_bpmn;
42 }
```

architecture in detail, and Section V-B explains how gPROFIT is used from a user perspective.

A. TECHNOLOGICAL ARCHITECTURE

The gPROFIT tool supports the theoretical model-driven architecture (described in Section IV-A), making our proposal easier to apply in professional contexts and facilitating the extraction of process models from LIS using model-driven mechanisms, heuristics, and machine learning techniques. gPROFIT was designed using the following design patterns: (*i*) the Separation of Concerns (SoC) pattern [4], which refers to the ability to identify, encapsulate and separate a computer system into separate sections that manage and encapsulate a specific set of information and functionalities; and (*ii*) the component-based design pattern [20], which emphasizes the separation of interconnected concerns to build the final software system. The advantages of these patterns have been widely discussed in the scientific literature [59], but, in short, they make it possible to reduce lead time, increase quality,

improve maintenance of component-based applications, and leverage costs by developing individual components.

The gPROFIT technological architecture is organized into two modules: the ≪***gPROFIT plugin module***≫ and the ≪***gPROFIT web module***≫. Both ones are shown in Figure 4.

The ≪***gPROFIT plugin module***≫ contains software components associated with process modelling, transformation mechanisms, and reverse engineering mechanisms. This module and all its software components were designed as plugins in Enterprise Architect (EA) and encoded with Microsoft technologies (e.g.,.NET Framework and C#). EA was selected as the base technology on which to develop our technological proposal because it is a well-known modelling tool in the professional context, it supports UML extension mechanisms, it allows the integration of new user-friendly GUIs (Graphical User Interfaces), and it provides model-driven mechanisms, among other features.

This ≪*gPROFIT plugin module*≫ contains the following main software components:

1) **Modelling component**. This aims to provide mechanisms for business process modelling in accordance with our time dimension metamodel. To achieve this objective, a Domain-Specific Language (DSL) was defined using UML inheritance mechanisms (specifically, UML-Profile). Space restrictions make impossible to show the complete UML-Profile of gPROFIT, but a fragment is briefly illustrated in Figure 5.
   Later, this UML-Profile was instantiated within Enterprise Architect (EA) using its Model Driven Generation (MDG[3]) technology, which allows to generate XMI-based UML-Profiles that can be integrated into the EA interface as toolbox panels (c.f. Figure 6).
2) **Semantic constraints control component**. This aims to guarantee the integrity of the process models since it makes it possible to automatically verify the compliance of all OCL semantic constraints in our theoretical proposal. Verification is carried out at runtime and in the background by gPROFIT when users instance our metamodel with the gPROFIT UML-Profile.
3) **MDE component**. This contains a library of methods encoded with C# in the gPROFIT plugin for EA. These methods develop each model-to-model transformation rule (formally and systematically defined in the previous section), and guarantee traceability between the process models.
4) **Reverse engineering component**. This component contains a specialized parser for database models and schemas. Its objective is to analyse the models and manage their mapping with our metamodel. After establishing the mapping, it invokes the MDE component to extract and build the business model in accordance with our metamodel, considering the real data stored in the LDBM. Once the business model is built, this component transforms it to BPMN [5].

The ≪*gPROFIT web module*≫ groups together software components associated with machine learning algorithms, which aim to predict the associations between elements of relational entities (entities and columns) and elements of the PIM-up metamodel (c.f. Section IV-B). For this purpose, supervised machine learning algorithms (aimed at direct prediction) were designed, with the support of unsupervised machine learning algorithms (aimed at searching for similarities and associations).

The ≪*gPROFIT web module*≫ was designed with a client-server architecture (c.f. Figure 7), which encapsulates the machine learning algorithms and exposes a REST API that can be consumed by any client (e.g., the ≪*gPROFIT plugin module*≫). This architecture makes it possible to obtain training data and increase the gPROFIT knowledge database from all the users of the gPROFIT platform. The gPROFIT algorithm improves its prediction accuracy by regularly training its algorithm.

Once the production environment dataset (SQL files) is obtained, this one is preprocessed and normalized before carrying out the training and execution of algorithms for predicting similarities between entities and associations. After executing these algorithms, the tool proposes an initial mapping between legacy relational entities and elements of the PIM-up metamodel.

B. gPROFIT FROM USER PERSPECTIVE

As mentioned earlier, gPROFIT was designed and developed using different technologies to support its modules and software components. Specifically, the gPROFIT module related to the definition and management of process models is supported by desktop tools (based on a plugin in Enterprise Architect; EA), whereas the machine learning module is supported by web technologies.

But since gPROFIT was designed to reduce cognitive load and theoretical complexity, these technologies are transparent from the user perspective, making the tool easier to apply in real contexts. The working method for using the tool is based on the following stages:

1) **LDBM analysis**. At this stage, the user (process engineer or business expert) manually studies and analyses the LIS data model to understand its structure and identify the relationships (mapping) between meta-attributes (of the PIM-upl metamodel) and LDBM.
2) **Configuration of mapping between the PIM-upl metamodel and the LDBM**. At this stage, the user configures the mapping of each metaclass and meta-attribute of the PIM-upl metamodel with the appropriate entities belonging to the LDBM.

[3]MDG Technologies allow to extend Enterprise Architect's modeling capabilities (toolboxes, UML-Profiles, patterns, templates and other modeling resources) to specific domains and notations. Available at: https://sparxsystems.com/resources/mdg_tech/

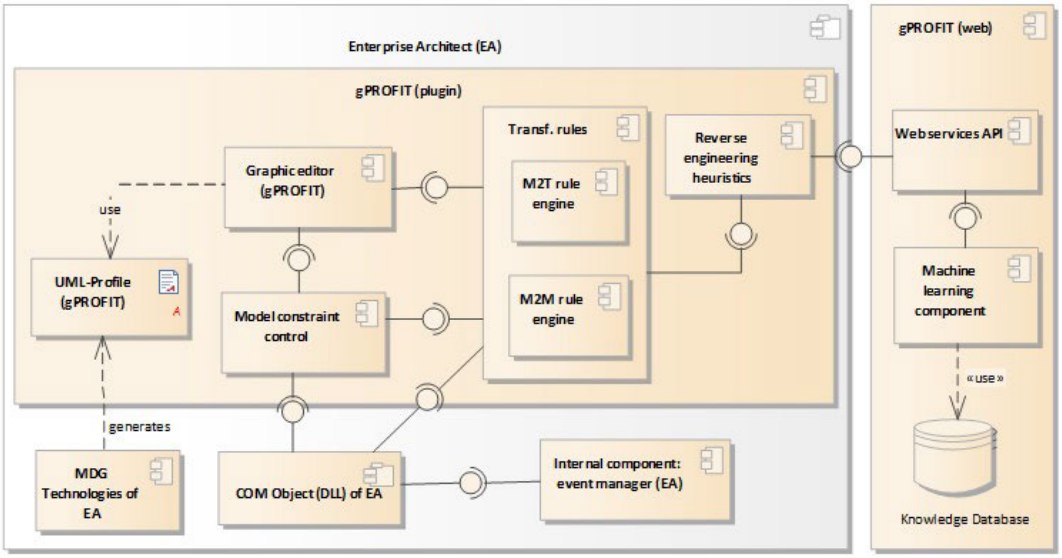


FIGURE 4. gPROFIT architecture and its software components.

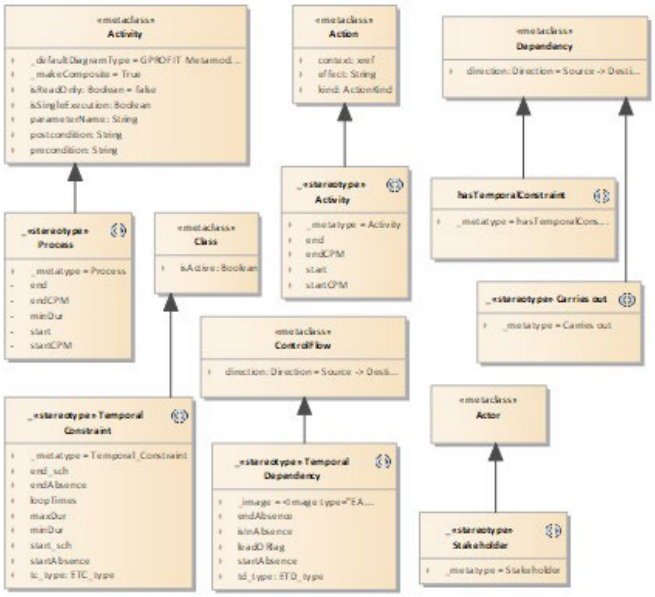


FIGURE 5. Fragment of the UML-Profile of gPROFIT.

3) **Application of machine learning techniques**. This stage is not mandatory, but users can use this gPROFIT utility to get a configuration baseline. This baseline can then be updated until the best result is achieved.
4) **Extraction of the business model**. Once the mapping has been established, the user can automatically run transformation rules to extract the business model from the legacy system data model.

All the stages except for the first manual stage are automatic and supported in gPROFIT. Space restrictions make it impossible to show many images of our tool in this paper, but it is briefly illustrated below.

1) MAIN INTERFACE AND gPROFIT WORKSPACE

The gPROFIT plugin for EA provides user-oriented GUIs to facilitate the definition, transformation, and management of process models in professional environments. This is possible thanks to the fact that EA is a well-known context modelling tool. Figure 8 shows the gPROFIT workspace and how it was integrated into EA. For ease of explanation, the figure has three labelled areas:

- **Area A**. gPROFIT's main menu shows two main options: create a new project and open an existing project. Process engineers or business experts can start using gPROFIT by indicating the project name and the path to the LDB.

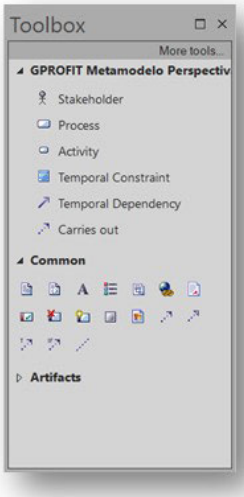


FIGURE 6. UML-Profile of gPROFIT implemented as toolbox of Enterprise Architect.

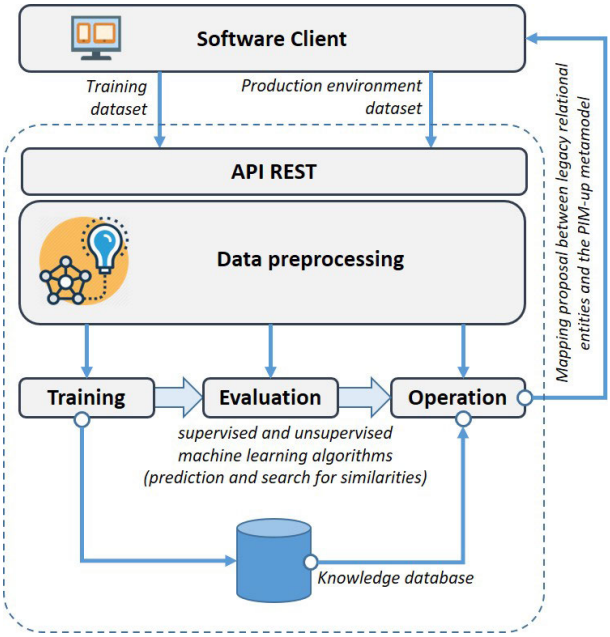


FIGURE 7. Client-server architecture of the ≪*gPROFIT web module*≫ (associated with machine learning algorithms).

- **Area B**. This area shows the user workspace. As can be seen in the figure, models are built visually and graphically by dragging and dropping elements from the toolbox. The toolbox is based on EA's MDG technology and was obtained after implementing our UML-Profile using this technology. Figure 8 shows part of the workflow associated with a clinical practice guideline.
- **Area C**. This area shows the structure and organization of models in the EA project browser. A package (stereotyped as ≪*VersiongPROFIT*≫) is created automatically when the user (process engineer or business expert) creates a new project in gPROFIT. The job of this package is to store a specific mapping between the concepts of our metamodel and the LDBM. This mapping can also be calibrated and adjusted by users as many times as they want before executing our reverse-engineered, model-driven mechanisms to obtain a BPMN business model from the LDBM. When a mapping is modified, a new version is created automatically (c.f. Figure 8, for example, shows three package versions). Area C of Figure 8 also shows two process models. In the lower zone (c.f. area C.2), gPROFIT automatically generates a process model according to our metamodel (Section IV-B) and, after obtaining this model, gPROFIT transforms it to BPMN notation (c.f. area C.1).

2) CONFIGURATION OF MAPPING BETWEEN METAMODEL AND LDBM

Regarding the metamodel-LDB mapping, gPROFIT also provides user-friendly wizards to establish this mapping per metaclass and meta-attribute of our metamodel (c.f. Section IV-B). For instance, Figure 9 shows the mapping configuration for the ≪*Activity metaclass*≫ and its meta-attributes.

The gPROFIT wizard shown in Figure 9 is similar for all metaclasses and contains the following information fields:

1) **Main Entity associated with the Metaclass (MEM)**. This field identifies and maps each metaclass with an entity of the LDBM. For instance, Figure 9 shows the mapping of the ≪*Activity metaclass*≫ with the ≪issue entity≫ of the LDBM. This information is referenced in the gPROFIT wizard as ≪*Main Entity field*≫.
2) **Entity and property of the LDBM per meta-attribute**. This information is established using the ≪*Table ER field*≫ and the ≪*Column field*≫ shown in Figure 9 (c.f. area A). For example, the ≪*name meta-attribute*≫ is mapped to the ≪*subject property*≫ of the ≪*issue entity*≫.

Before establishing the mapping of each meta-attribute, it is necessary to identify how each one is represented in the LDBM. It is acceptable to consider that the entity associated with the meta-attribute usually matches MEM. Due to the great variability of LIS designs, however, more casuistry has been identified. The casuistry considered by gPROFIT is explained in detail below.

The **first casuistry, denominated** ≪***entity method***≫ is usually the standard, basic mapping. It means the meta-attribute could be mapped to a property of the MEM. The entity and property information are indicated in ≪*Table ER field*≫ and ≪*Column field*≫, respectively (c.f. Figure 9; area A).

The **second casuistry, denominated** ≪***foreign method***≫, means the meta-attribute could be mapped to a property of a secondary entity of the LDBM. The relationship between this secondary entity and MEM is usually designed into LIS by means of a foreign key in the MEM, which stores the primary key of the secondary entity. This information is configured in

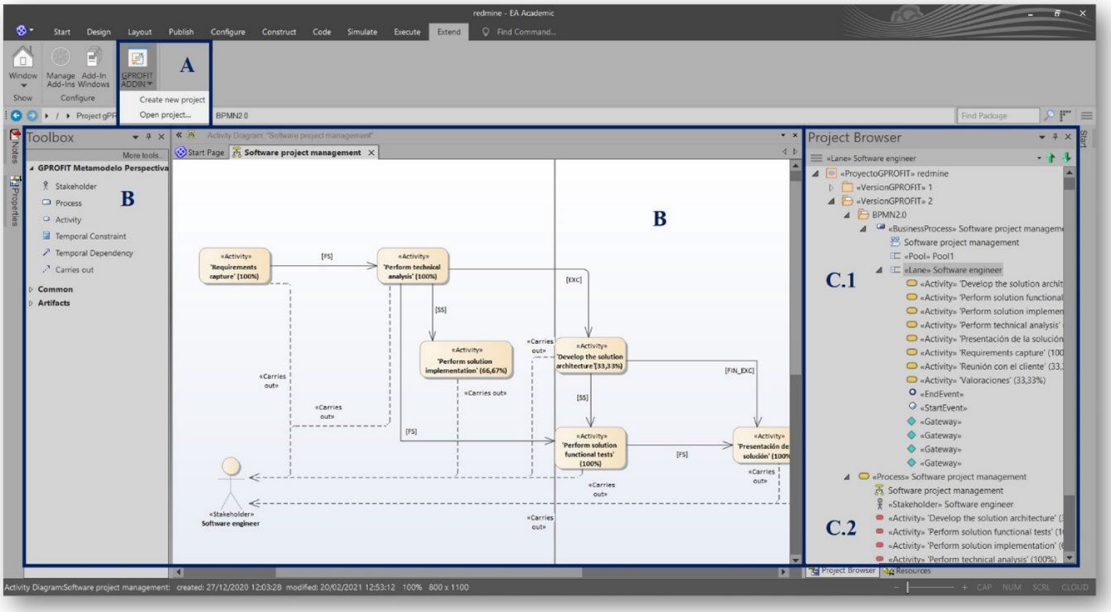


FIGURE 8. gPROFIT workspace with direct access to its functionalities.

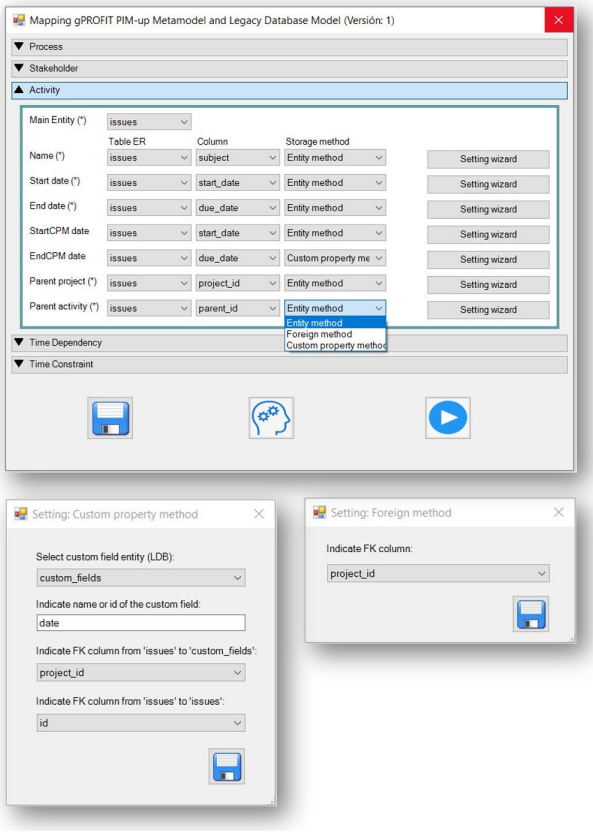


FIGURE 9. gPROFIT configuration wizard, whose objective is to map the legacy database model with the PIM-up metamodel (c.f. Section IV-B).

the gPROFIT wizard as follows: the information about the secondary entity and its property is configured in ≪*Table ER field*≫ and ≪*Column field*≫ (c.f. Figure 9; area A), while ≪*Foreign Key field*≫ (c.f. Figure 9; area B) allows the mapping of the foreign key (in the MEM), which stores the primary key of the secondary entity.

The **third casuistry is denominated** ≪***custom property method***≫. A common practice for extending a LIS data model is to use custom properties associated with specific data entities of the LDBM. Here, three entities are involved: (*i*) the first is an entity that stores the definition of each custom property into LIS; (*ii*) the second is MEM (mentioned earlier); and (*iii*) the third aims to relate the previous entities (through their foreign keys) with the value itself of the meta-attribute.

The setting of this third casuistry is configured in the following two gPROFIT wizards. The wizard shown in Figure 9 (c.f. area A) allows the mapping of the entity and its property where the value of the meta-attribute is stored into the LDBM. The fields used to achieve this objective are ≪*Table ER field*≫ and ≪*Column field*≫, respectively. The wizard shown in Figure 9 (c.f. area C) allows (*i*) the mapping of the entity where the custom properties are defined into LIS (≪*Custom fields table field*≫ and ≪*Custom fields name field*≫); and (*ii*) the mapping of the entity where the foreign keys are stored to relate the two previous entities (the custom properties and MEM entities) using the ≪*Foreign Key* 1 *field*≫ and the ≪*Foreign Key* 2 *field*≫. ≪*Foreign Key* 1 *field*≫ relates the value entity with the custom properties entity, whereas ≪*Foreign Key* 2 *field*≫ relates MEM with the custom properties entity.

3) SUPPORT FOR MACHINE LEARNING ALGORITHMS

The previous section explained how users can manually establish mapping between the PIM-up metamodel (c.f. Section IV-B) and the LDBM. Before configuring this

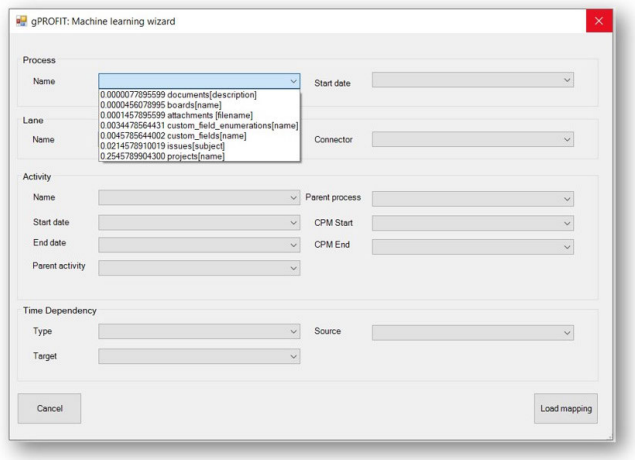


FIGURE 10. gPROFIT wizard with data returned by the gPROFIT machine learning service. Note: This service offers a set of mappings (with probability according to knowledge base similarity) between meta-attributes of the PIM-up metamodel (c.f. Section IV-B) and columns of the LDBM.

mapping, however, the user can run assistive utilities (based on machine learning techniques) to obtain an initial mapping.

In gPROFIT these utilities can be invoked using the central bottom button of Figure 9. The result is shown in Figure 10. Specifically, the machine learning algorithm analyses the names and relationships of the entities and properties that make up the schema of the LDBM provided by the user. Once the schema has been analysed, gPROFIT proposes several mappings for each meta-attribute. Each mapping is also assigned a probability, calculated after considering the gPROFIT knowledge base.

4) AUTOMATIC OBTAINING OF THE BUSINESS MODEL

Finally, after establishing the mapping between our meta-model and the LDBM (either manually or with machine learning support), the process engineer or business expert can automatically execute the reverse engineering algorithms by clicking on the ≪*play button*≫ (c.f. Figure 9).

These algorithms analyse the LDBM and its data to identify, merge, and unify each time instance of the business process in a single business model in accordance with our metamodel. Once the algorithm has been run, gPROFIT returns a complete process model considering the time perspective of all instances. This process model is then transformed into BPMN, a notation familiar to business experts and process engineers.

VI. VALIDATION: PROOFS OF CONCEPT

The purpose of this section is not to provide an in-depth, comprehensive description of an experiment, but to present insights and ideas for a statistically significant validation of the proposal. In this context, the applicability of the gPROFIT proposal was validated on three proofs of concept in the technological solutions department of the Servinform company,[4] a Spanish technology company that offers services in the full lifecycle of marketing, web, sales, contact, point of sale, back office, communications and logistics.

The proofs of concept involved obtaining BPMN models after applying the gPROFIT proposal to the databases of three of Servinform's Legacy Information System (LIS) instances. The main objective is to obtain a process model from a data entity planned in time with its activities, time rules and time dependencies. If the source LIS provides a well-defined data entity (considering its time perspective), then, the process model obtained should be a close approximation to the organization's process.

The proofs of concept have been carried out on the instance of three LIS, which simulate three real use cases: (*i*) project management area, whose operations are based on the Redmine system[5]; (*ii*) sales area, whose operation is based on the Aras PLM system,[6] which includes a specific sales management module developed for the Servinform's customer; and (*iii*) back office area whose management is carried out on the Atlassian Jira system.[7] The datasource for our proposal is a relational database backup (that is, an SQL dump file) each legacy system (Redmine, Aras PLM and Atlassian Jira). Each SQL file contains records of the execution trace of each business process previously mentioned (project management process, sales management process, and back office process).

Moreover, as mentioned at the beginning of the Section V-B, before running the business model extraction procedure (c.f. Section IV-C), first, it is necessary to analyze the database model of each source LIS and establish a mapping between its legacy database model and the gPROFIT PIM-upl metamodel (c.f. Section IV-B). In this sense, Table 1 shows the possible mapping of the three previously mentioned LIS and its legacy database with the gPROFIT metamodel.

Below, the associated mapping of each proof-of-concept is explained. This mapping consists of relating each metaclass and meta-attribute of the PIM-upl metamodel with the appropriate entities belonging to the LDBM (Redmine, Aras PLM and Atlassian Jira).

Regarding the Redmine system, after analyzing its LDBM, it was possible to map each metaclass of the PIM-upl metamodel with entities of the Redmine's LDBM, as shown in Table 1. The left area of the Figure 11 shows the result of applying gPROFIT on the Redmine system.

On the one hand, the ≪*Process*≫, ≪*Activity*≫ and ≪*Time Dependency metaclasses*≫ were mapped with the ≪*projects*≫, ≪*issue*≫ and ≪*issue_relations entities*≫ of the Redmine's LDBM, respectively. In this sense, in addition,

[4]Servinform company. Available at: https://www.servinform.es/soluciones-tecnologicas/

[5]Redmine database schema. Available at: https://www.redmine.org/projects/redmine/wiki/DatabaseModel

[6]Aras PLM. Available at: https://www.aras.com/en

[7]Atlassian Jira database schema. Available at: https://developer.atlassian.com/server/jira/platform/database-schema/

TABLE 1. Mapping between the gPROFIT PIM-upl metamodel and legacy database models of three legacy information systems (Redmine, Aras PLM and Atlassian Jira). Acronyms associated with mapping casuistics (c.f. Section V-B2): ≪*entity method*≫ (EM); ≪*foreign method*≫ (FM); and ≪*custom property method*≫ (CPM).

gPROFIT PIM-upl metamodel		Redmine		Aras PLM		Atlassian Jira	
Metaclass	**Meta-attribute**	**Table**	**Column**	**Table**	**Column**	**Table**	**Column**
Process	name	projects	name (EM)	t_invoice	concept (EM)	project	pname (EM)
	start		start_date (EM)		creation_date (EM)	-	-
	end		due_date (EM)		deadline_date (EM)	-	-
	startCPM		start_date (EM)	-	-	-	-
	endCPM		due_date (EM)	-	-	-	-
	minDur	-	-	-	-	-	-
Stakeholder	name	users	login (EM)	t_user	first_name (EM)	cwd_user	user_name (EM)
	assignedTo	issues	assigned_to_id (FM)	t_item	created_by_id (FM)	jiraissue	assignee (FM)
Activity	name	issues	subject (EM)	t_item	item_state (EM)	jiraissue	summary (EM)
	start		start_date (EM)		created_on (EM)	customfieldvalue customfield customfieldvalue (FK1) customfieldvalue (FK2)	datavalue (CPM) id: '10107' id (column) issue (column)
	end		due_date (EM)		modified_on (EM)	jiraissue	duedate (EM)
	startCPM		start_date (EM)		created_on (EM)		updated (EM)
	endCPM		due_date (EM)		modified_on (EM)		duedate (EM)
	parentProject		project_id (EM)		invoice_id (EM)		project (EM)
	parentID		parent_id (EM)		item_id (EM)	issuelink	source
Time Dependency	td_type	issue_relations	relation_type (EM)	t_history	action (EM)	issuelink	linktype (EM)
	origen		issue_from_id (EM)		source_id (EM)		source (EM)
	destino		issue_to_id (EM)		target_id (EM)		destination (EM)
	startdate		-		created_on (EM)		-

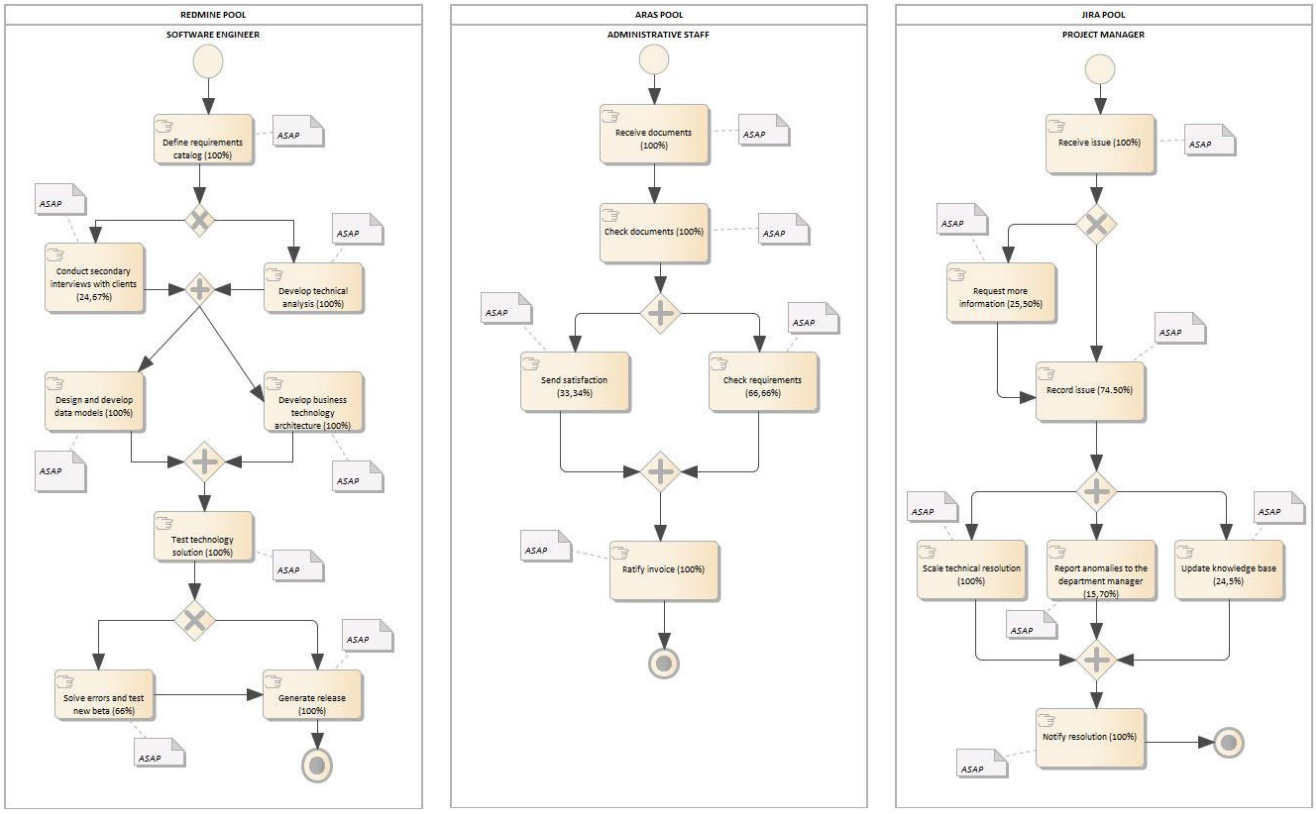


FIGURE 11. Results of gPROFIT proofs of concept carried out on three legacy information systems (Redmine, Aras PLM and Atlassian Jira).

each meta-attribute was directly mapped with columns of the mentioned entities using the basic mapping method (denominated ≪*entity method*≫; c.f. Section V-B2).

On the other hand, the mapping of the ≪*Stakeholder metaclass*≫ and the Redmine's ≪*users entity*≫ was also established with the ≪*entity method*≫. However, it was necessary to use the ≪*foreign method*≫ (c.f. Section V-B2) to map the relationship between the ≪*Stakeholder metaclass*≫ and the ≪*Activity metaclass*≫. This method means the meta-attribute is mapped to a property of a secondary entity of

the LDBM. Specifically, the ≪*assignedTo meta-attribute*≫ (belonging to the ≪*Stakeholder metaclass*≫) was mapped to the ≪*assigned_to_id column*≫ of the Redmine's ≪*issues entity*≫.

Regarding the proof of concept related to Aras PLM, it is interesting to remember that this proof of concept is based on a invoicing module deployed on Aras PLM. This module was developed by Servinform for one of its customers. In this context, after analyzing the Aras's LDBM, it was possible to map each metaclass of the PIM-upl metamodel with entities of the Aras as shown in Table 1. The middle area of the Figure 11 shows the result of applying gPROFIT on the Aras system.

On the one hand, the ≪*Process*≫ was mapped with the Aras ≪*t_invoice entity*≫, whereas ≪*Activity*≫ and ≪*Time Dependency metaclasses*≫ was mapped with the Aras ≪*t_item entity*≫ and ≪*t_history entity*≫, respectively. This last entity stored the history of transitions associated with each invoice object. These transitions establish the workflow associated with the invoicing process carried out by the organization's sales department. As shown Table 1, the previously mentioned metaclasses were directly mapped with columns of the mentioned entities using the ≪*entity method*≫ (c.f. Section V-B2).

On the other hand, the mapping of the ≪*Stakeholder metaclass*≫ and the Aras's ≪*t_user entity*≫ was also established with the ≪*entity method*≫ (c.f. Table 1). In addition, it is possible to observe the same mapping configuration of the ≪*Stakeholder metaclass*≫ and its relationship with ≪*Activity metaclass*≫ than the Redmine mapping settings. Specifically, the ≪*foreign method*≫ was also used in the gPROFIT configuration for the Aras case. In addition, the ≪*assignedTo meta-attribute*≫ of the ≪*Stakeholder metaclass*≫ was mapped to the ≪*created_by_id column*≫ of the Aras ≪*t_item entity*≫.

Regarding the Atlassian Jira system, after analyzing its LDBM, it was possible to map each metaclass of the PIM-upl metamodel with entities of the Jira LDBM, as shown in Table 1. The right area of the Figure 11 shows the result of applying gPROFIT on the Jira system.

On the one hand, the ≪*Process*≫, ≪*Activity*≫ and ≪*Time Dependency metaclasses*≫ were mapped with the ≪*project*≫, ≪*jiraissue*≫ and ≪*issuelink entities*≫ of the Jira LDBM, respectively. In addition, each meta-attribute was directly mapped with columns of the previous entities using the ≪*entity method*≫ (c.f. Section V-B2). However, there is a meta-attribute that could not be directly mapped to the entity associated with its metaclass. Specifically, it is the ≪*start meta-attribute*≫ of the ≪*Activity metaclass*≫. This metaclass was mapped to ≪*jiraissue entity*≫ as mentioned above, but this entity does not contain by default the start date. To overcome this limitation, the organization defined a custom field to store the start date in ≪*jiraissue entity*≫.

Before continuing, it should be mentioned that any custom field can be defined in the Jira ≪*customfield entity*≫. In our case, the organization needed to define the startdate custom field associated with the Jira ≪*jiraissue entity*≫. This association was established between the ≪*jiraissue and customfield entities*≫ using foreign keys. In addition, a third entity is involved in this relationship: the Jira ≪*customfieldvalue entity*≫, which stores the value that the custom field contains.

In this context, the mapping of the ≪*start meta-attribute*≫ (≪*Activity metaclass*≫) was configured using the ≪*custom property method*≫ (c.f. Section V-B2). As Table 1 shows, the following configuration was established:

- ≪*Table ER field*≫: ≪*customfieldvalue entity*≫. It refers to the Jira entity where the value of the custom field is stored.
- ≪*Column field*≫: ≪*ID column*≫. It refers to the column in the previous table where the value of the custom field is stored.
- ≪*Custom fields table field*≫: ≪*customfield entity*≫. It refers to the Jira entity where custom fields are defined in the system.
- ≪*Custom fields name field*≫: '10107'. This value is the internal name created by Jira for the stardate custom field.
- ≪*Foreign Key* 1 *field*≫: ≪*ID column*≫. It refers to the foreign key column from ≪*customfieldvalue entity*≫ to ≪*customfield entity*≫.
- ≪*Foreign Key* 2 *field*≫: ≪*issue column*≫. It refers to theforeign key column from ≪*customfieldvalue entity*≫ to ≪*jiraissue entity*≫.

On the other hand, the mapping of the ≪*Stakeholder metaclass*≫ and the Jira's ≪*cwd_user entity*≫ was also established with the ≪*entity method*≫. However, it was necessary to use the ≪*foreign method*≫ (c.f. Section V-B2) to map the relationship between the ≪*Stakeholder metaclass*≫ and the ≪*Activity metaclass*≫. Specifically, the ≪*assignedTo meta-attribute*≫ (belonging to the ≪*Stakeholder metaclass*≫) was mapped to the ≪*assignee column*≫ of the Jira's ≪*jiraissue entity*≫.

VII. CONCLUSION AND FUTURE WORKS

The BPM approach is being used increasingly by business experts, increasing the competitiveness of all types of organizations in a globalized world. As the leading vehicle of business process discovery, process mining techniques [51], [52], [54], [55] are a good way to obtain processes from PAIS event logs, but not from non-process-aware systems lacking in these artifacts. Anyway, Non-PAIS hide a lot of knowledge [54] about the execution of business logic, which experts can leverage to gain insight into organizational processes, which would otherwise be wasted in legacy databases.

We propose a MDE-based framework allowing different roadmaps to transform LIS's databases into knowledge about business processes. Database states conform to database schema or models, which are mapped into processes instances, which conform to generated process models.

The reverse engineering process uses algorithms that are focused on: *(i)* metamodelling, and *(ii)* time dimension of business processes. We have developed gPROFIT (an is automated tool), which implements our MDE-based framework. Then, gPROFIT has been applied to three Non-PAIS that the Servinform Information Technology Company uses with different customers. Servinform and their customers, as business experts, appreciate the added value of generated process models, which allow them to evaluate the generated process models. In this way, the processes obtained please the experts, assuring in many cases that they had not been formally defined until now. They agree in appreciating the results, which are generated as processes close to the real ones, although they must be analyzed and completed in a BPM [53] life cycle of continuous improvement.

We know that this study is just one more step towards validating the approach, and the following future works are planned to improve our proposal.

On the one hand, gPROFIT is at present a tool that assists in the automatic generation of processes from legacy databases. We plan to carry out more statistical experiments to adequately refute the framework in more cases, in different organizations and with different types of LIS.

On the other hand, another line of research should handle other dimensions of the processes, such as Resources and Cases. Reverse engineering should consider different legacy source databases, merging knowledge drawn from different perspectives of the same business logic into a single process model.

Finally, we also think it would be interesting to generate the event log from the states of the inherited database, for example, generating a standard format such as XES [35], [58], which could be used with conventional process mining techniques [53] and analyze the process models that they generate.

REFERENCES

[1] C. Arevalo, M. J. Escalona, I. Ramos, and M. Domínguez-Muñoz, "A metamodel to integrate business processes time perspective in BPMN 2.0," *Inf. Softw. Technol.*, vol. 77, pp. 17–33, Sep. 2016.

[2] C. Arevalo, I. Ramos, J. Gutiérrez, and M. Cruz, "Practical experiences in the use of pattern-recognition strategies to transform software project plans into software business processes of information technology companies," *Sci. Program.*, vol. 2019, May 2019, Art. no. 7973289.

[3] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, "Legacy information systems: Issues and directions," *IEEE Softw.*, vol. 16, no. 5, pp. 103–111, Sep. 1999.

[4] J. Brichau, M. Glandrup, S. Clarke, and L. Bergmans, "Advanced separation of concerns," in *Proc. Eur. Conf. Object-Oriented Program.* Hungary: Springer, 2001, pp. 107–130.

[5] P. Briol, *BPMN, the Business Process Modeling Notation Pocket Handbook*. USA: Lulu, 2008.

[6] G. Canfora, "Software process: Characteristics, technology and environments," Italy, Tech. Rep., 2004, pp. 6–10, vol. 5, no. 5.

[7] M. Cimbaljević, U. Stankov, and V. Pavluković, "Going beyond the traditional destination competitiveness–reflections on a smart destination in the current research," *Current Issues Tourism*, vol. 22, no. 20, pp. 2472–2477, Dec. 2019.

[8] A. Cleve, J. Henrard, and J.-L. Hainaut, "Data reverse engineering using system dependency graphs," in *Proc. 13th Work. Conf. Reverse Eng.*, 2006, pp. 157–166.

[9] V. Cosentino and S. Martinez, "Extracting UML/OCL integrity constraints and derived types from relational databases," in *Proc. 13th Int. Workshop OCL, Model Constraint Query Lang.*, 2013, pp. 43–52.

[10] E. G. L. de Murillas, W. M. van der Aalst, and H. A. Reijers, "Process mining on databases: Unearthing historical data from redo logs," in *Proc. Int. Conf. Bus. Process Manage.* Austria: Springer, 2016, pp. 367–385.

[11] M. Dumas, W. van der Aalst, and A. Ter Hofstede, *Process Aware Information Systems*, vol. 1. Hoboken, NJ, USA: Wiley, 2005.

[12] L. A. Enríquez, H. Cuevas-Vargas, and M. G. Adame, "The impact of information and communication technologies on the competitiveness: Evidence of manufacturing SMEs in Aguascalientes, Mexico," *Int. Rev. Manage. Bus. Res.*, vol. 4, no. 3, p. 758, 2015.

[13] H. Eshuis, *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*. The Netherlands: Centre for Telematics and Information Technology (CTIT), Univ. Twente, 2002.

[14] R. Gabryelczyk, "Exploring BPM adoption factors: Insights into literature and experts knowledge," *Inf. Technol. Manage., Emerg. Res. Appl.*, vol. 1, pp. 155–175, Sep. 2018.

[15] L. García-Borgoñón, M. A. Barcelona, J. A. García-García, M. Alba, and M. J. Escalona, "Software process modeling languages: A systematic literature review," *Inf. Softw. Technol.*, vol. 56, no. 2, pp. 103–116, Feb. 2014.

[16] N. Genon, P. Heymans, and D. Amyot, "Analysing the cognitive effectiveness of the BPMN 2.0 visual notation," in *Proc. Int. Conf. Softw. Lang. Eng.* The Netherlands: Springer, 2010, pp. 377–396.

[17] E. G. L. de Murillas, H. A. Reijers, and W. M. van der Aalst, "Connecting databases with process mining: A meta model and toolset," *Softw. Syst. Model.*, vol. 2018, no. 2, pp. 1–39, 2018.

[18] N. Goodman, *Languages of Art: An Approach to a Theory of Symbols*. New York, IN, USA: The Bobbs-Merrill, 1968.

[19] A. Haseeb, E. Xia, S. Saud, A. Ahmad, and H. Khurshid, "Does information and communication technologies improve environmental quality in the era of globalization? An empirical analysis," *Environ. Sci. Pollut. Res.*, vol. 26, no. 9, pp. 8594–8608, Mar. 2019.

[20] G. T. Heineman and W. T. Councill, "Component-based software engineering," in *Putting Pieces Together*. Boston, MA, USA: Addison-Westley, 2001, p. 16.

[21] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quart.*, vol. 28, pp. 75–105, Mar. 2004.

[22] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, "Legacy information systems: Issues and directions," *IEEE Softw.*, vol. 16, no. 5, pp. 103–111, 1999.

[23] *Software and Systems Engineering—Life Cycle Management—Guidelines for Process Description*, Standard ISO/IEC TR 24774:2007, 2007.

[24] *Information Technology—Object Management Group Object Constraint Language (OCL)*, Standard ISO/IEC 19507:2012, 2012.

[25] J. E. Kelley and M. R. Walker, "Critical-path planning and scheduling," in *Proc. IRE-AIEE-ACM Comput. Conf. IRE-AIEE-ACM (Eastern)*, 1959, pp. 160–173.

[26] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, "How Do professionals perceive legacy systems and software modernization?" in *Proc. 36th Int. Conf. Softw. Eng.*, May 2014, pp. 36–47.

[27] M. Dumas, W. M. Van der Aalst, and A. H. T. Hofstede, *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Hoboken, NJ, USA: Wiley, 2005.

[28] C. A. Maldonado, D. M. J. E. Cuaresma, and D. I. R. Román, "Una propuesta basada en el paradigma dirigido por modelos para la extracción de procesos del software desde sistemas heredados.: Utilizando la perspectiva temporal," Ph.D. dissertation, Dept. Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Seville, Spain, 2016.

[29] M. Malinova and J. Mendling, "A qualitative research perspective on BPM adoption and the pitfalls of business process modeling," in *Proc. Int. Conf. Bus. Process Manage.* Estonia: Springer, 2012, pp. 77–88.

[30] D. McAleese, *Economics for Business: Competition, Macro-Stability, and Globalisation*. London, U.K.: Pearson, 2004.

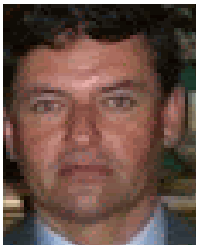
[31] G. G. Miller, "The characteristics of agile software processes," in *Proc. Technol. Object-Oriented Lang., Int. Conf.*, vol. 1. Washington, DC, USA: IEEE Computer Society, 2001, p. 0385.

[32] S. Mohapatra, *Usiness Process Reengineering: Automation Decision Points in Process Reengineering*. USA: Springer, 2012.

[33] D. Moody and J. van Hillegersberg, "Evaluating the visual syntax of UML: An analysis of the cognitive effectiveness of the UML family of diagrams," in *Proc. Int. Conf. Softw. Lang. Eng.* France: Springer, 2008, pp. 16–34.

[34] D. L. Moody, P. Heymans, and R. Matulevicius, "Improving the effectiveness of visual representations in requirements engineering: An evaluation of i* visual syntax," in *Proc. 17th IEEE Int. Requirements Eng. Conf.*, Aug. 2009, pp. 171–180.
[35] M. Bangalore Shankara Narayana, H. Khalifa, and W. van der Aalst, "JXES: JSON support for the XES event log standard," 2020, *arXiv:2009.06363*. [Online]. Available: http://arxiv.org/abs/2009.06363
[36] P. Newcomb, "Architecture-driven modernization (ADM)," in *Proc. 12th Work. Conf. Reverse Eng. (WCRE)*, 2005, p. 237.
[37] OMG, "Abstract syntax tree metamodel 1.0," Object Manage. Group, Model. Softw. Modernization, Needham, MA, USA, Tech. Rep. formal/01-01-11, 2011.
[38] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.
[39] J. Pérez, I. Ramos, V. Anaya, J. M. Cubel, F. Domínguez, A. Boronat, and J. A. Carsí, "Data reverse engineering of legacy databases to object oriented conceptual schemas," *Electron. Notes Theor. Comput. Sci.*, vol. 72, no. 4, pp. 7–19, Mar. 2003.
[40] R. Pérez-Castillo, I. G.-R. de Guzmán, and M. Piattini, "Business process archeology using Marble," *Inf. Softw. Technol.*, vol. 53, no. 10, pp. 1023–1044, Oct. 2011.
[41] R. Pérez-Castillo, I. G.-R. de Guzmán, and M. Piattini, "Knowledge discovery metamodel-ISO/IEC 19506: A standard to modernize legacy systems," *Comput. Standards Interfaces*, vol. 33, no. 6, pp. 519–532, Nov. 2011.
[42] R. Perez-Castillo, M. Fernandez-Ropero, I. G.-R.-D. Guzman, and M. Piattini, "MARBLE. A business process archeology tool," in *Proc. 27th IEEE Int. Conf. Softw. Maintenance (ICSM)*, Sep. 2011, pp. 578–581.
[43] R. Pérez-Castillo, I. G.-R. de Guzmán, I. Caballero, and M. Piattini, "Software modernization by recovering Web services from legacy databases," *J. Softw., Evol. Process*, vol. 25, no. 5, pp. 507–533, May 2013.
[44] R. Pérez-Castillo, I. G.-R. de Guzmán, M. Piattini, and Á. S. Places, "A case study on business process recovery using an e-government system," *Softw., Pract. Exper.*, vol. 42, no. 2, pp. 159–189, Feb. 2012.
[45] T. Quatrani and U. Evangelist, "Introduction to the unified modeling language," *Tech. Discuss. UML*, vol. 6, no. 11, p. 3, 2003.
[46] J. Recker, "BPMN modeling-who, where, how and why," *BPTrends*, pp. 1–8, May 2008.
[47] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Austria: Springer, 2012.
[48] D. C. Schmidt, "Model-driven engineering," *Comput.-IEEE Comput. Soc.*, vol. 39, no. 2, p. 25, 2006.
[49] R. Schuppenies and S. Steinhauer, "Software process engineering metamodel," OMG Group, Needham, MA, USA, Tech. Rep. formal/02-11-14, Nov. 2002.
[50] S. Stavru, I. Krasteva, and S. Ilieva, "Challenges of model-driven modernization—An agile perspective," in *Proc. MODELSWARD*, 2013, pp. 219–230.
[51] W. van der Aalst, "Academic view: Development of the process mining discipline," in *Process Mining in Action*. USA: Springer, 2020, pp. 181–196.
[52] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, and A. Burattin, "Process mining manifesto," in *Proc. Int. Conf. Bus. Process Manage.* Springer, 2011, pp. 169–194.
[53] W. M. P. van der Aalst, "Business process management: A personal view," *Bus. Process Manage. J.*, vol. 10, no. 2, Apr. 2004.
[54] W. M. Van der Aalst, "Extracting event data from databases to unleash process mining," in *BPM-Driving Innovation in a Digital World*. USA: Springer, 2015, pp. 105–128.
[55] W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. A. de Medeiros, M. Song, and H. M. W. Verbeek, "Business process mining: An industrial application," *Inf. Syst.*, vol. 32, no. 5, pp. 713–732, Jul. 2007.
[56] W. M. P. van der Aalst, M. Weske, and D. Grünbauer, "Case handling: A new paradigm for business process support," *Data Knowl. Eng.*, vol. 53, no. 2, pp. 129–162, May 2005.
[57] A. Van Looy, "A quantitative and qualitative study of the link between business process management and digital innovation," *Inf. Manage.*, vol. 58, no. 2, Mar. 2021, Art. no. 103413.
[58] H. Verbeek, J. C. Buijs, B. F. Van Dongen, and W. M. Van Der Aalst, "Xes, xesame, and prom 6," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Tunisia: Springer, 2010, pp. 60–75.
[59] P. Vitharana, "Risks and challenges of component-based software development," *Commun. ACM*, vol. 46, no. 8, pp. 67–72, Aug. 2003.
[60] M. Von Rosing, H. Von Scheel, and A.-W. Scheer, *The Complete Business Process Handbook: Body of Knowledge From Process Modeling to BPM*, vol. 1. San Mateo, CA, USA: Morgan Kaufmann, 2014.
[61] M. Zanoni, F. Perin, F. A. Fontana, and G. Viscusi, "Pattern detection for conceptual schema recovery in data-intensive systems," *J. Softw., Evol. Process*, vol. 26, no. 12, pp. 1172–1192, Dec. 2014.

JULIAN A. GARCÍA-GARCÍA received the Ph.D. degree in computer science from the University of Seville, Spain. He is currently a Professor and a Researcher with the University of Seville, where he has been participating in research and development projects, as a Researcher, since 2008. His current research interests include the areas of software engineering, business process management (BPM), model-driven engineering, and quality assurance. He manages several technological transfer projects with companies, and he participates as a committee member of several international conferences and journals.



C. AREVALO MALDONADO received the M.Eng. degree in industrial engineering and the Ph.D. degree in computer science from the University of Seville, Spain. He is currently a Professor and a Researcher with the University of Seville. He has 45 years of experience in multiple IT projects in industry and government and 35 years in education with the university. He has participated in research and development projects, as a Researcher in software engineering, business process management, engineering directed to models, and databases.

AYMAN MEIDAN received the Ph.D. degree in computer science from the University of Seville, Spain, in 2019. He is currently a Researcher with the University of Seville. His current research interests include the areas of software engineering, business and software process management, model-driven engineering, and quality assurance. Since 2014, he has been participating as a Researcher in several technological transfer and research and development projects.

ESTEBAN MORILLO-BARO is currently a Telecommunications Engineer with the University of Seville, Spain. His professional career has been linked to ICT consulting, since 2000. Specifically, he developed techniques to improve process-oriented information systems in any organization. He is currently specialized in the design and development of RPA techniques to improve the efficiency of business processes considering the full process life-cycle: methodological, technological, and development.

MARÍA JOSÉ ESCALONA received the Ph.D. degree in computer science from the University of Seville, Spain, in 2004. She is currently a Full Professor with the Department of Computer Languages and Systems, University of Seville. She manages the web engineering and early testing research group. Her current research interests include the areas of requirement engineering, web system development, model-driven engineering, early testing, and quality assurance. She also collaborates with public companies like the Andalusian Regional Ministry of Culture and Andalusian Health Service in quality assurance issues.