Information Sciences 595 (2022) 18–37

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

SOWCompact: A federated process mining method for social workflows

Javier Rojo [b],*, Jose Garcia-Alonso [b], Javier Berrocal [b], Juan Hernández [b], Juan Manuel Murillo [b], Carlos Canal [a]

[a] ITIS Software, Universidad de Málaga, Spain
[b] University of Extremadura, Spain

A R T I C L E   I N F O

Article history:
Received 9 March 2021
Received in revised form 14 February 2022
Accepted 18 February 2022
Available online 24 February 2022

Keywords:
process mining
Pattern discovery
Social workflows
Federated process mining

A B S T R A C T

The exaggerated use of smartphones and growing informatization of the environment allows modeling people's behavior as a process, namely, a social workflow, where both individual actions and interactions with other people are captured. This modelling includes actions that are part of an individual's routine, as well as less frequent events. Although infrequent actions may provide relevant information, it is routine behaviors that characterize users. However, the extraction of this knowledge is not simple. Current process mining techniques face problems when analyzing large amounts of traces generated by many users. When very different behavioral patterns are integrated, the resulting social workflow does not clearly depict their behavior, either individually or as a group. Proposals based on frequent pattern mining aim to distinguish traces that characterize frequent behaviors from the rest. However, tools that allow grouping/filtering of users with a common behavior pattern are needed beforehand, to analyze each of these groups separately. This study presents the so-called federated process mining and an associated tool, SOWCompact, based on this concept. Its potential is validated through the case study called activities of daily living (ADL). Using federated process mining, along with current process mining techniques, more compact processes using only the social workflow's most relevant information are obtained, while allowing (event enabling) the analysis of these social workflows.

1. Introduction

The development of process mining enables access to new frontiers in the analysis of social workflows (SOW) [17]. The exaggerated use of mobile, particularly smartphones, and Internet of Things (IoT) devices is placing a huge amount of the sensors around people. Owing to this growing infrastructure unprecedented volumes of information regarding people's lives have become readily available. Representing all this information in a chronological log allows us to consider a person's life as a process execution, namely, the process of that individual's life. Subsequently, routine behaviors and patterns of conduct can be inferred. The analysis of thousands or millions of these executions pave the way for determining how individuals and groups behave; thus, providing new perspectives on SOWs [17]. Subsequently, it is possible to further analyze patterns,

* Corresponding author.
E-mail addresses: javirojo@unex.es (J. Rojo), jgaralo@unex.es (J. Garcia-Alonso), jberolm@unex.es (J. Berrocal), juanher@unex.es (J. Hernández), juanmamu@unex.es (J.M. Murillo), carloscanal@uma.es (C. Canal).

https://doi.org/10.1016/j.ins.2022.02.035
0020-0255/© 2022 Elsevier Inc. All rights reserved.

such as calculating the probability of an individual performing a given activity. The corresponding activity is defined as the next activity of an individual who previously engaged in a specific sequence of activities, or the most likely behaviors within a particular social group. Examples of systems relying on such techniques are those that analyze interactions among individuals to improve recommendation systems [36], how customers move in shopping malls to determine preferred destinations [14], or assess patient routines to establish causal relationships in healthcare applications [49].

Although applying process mining to analyze SOWs can be particularly useful, certain facts must be considered first. Process mining techniques and algorithms assume that traces in the logs correspond to instances of reality that tend to be governed by a defined process. However, when it comes to social flows, each trace corresponds to a behavior of an individual, which is not necessarily deterministic. Therefore, a behavior does not need to match that of other individuals. When it does, it is highly likely that this is a coincidence [17]. Using process mining techniques on a set of traces when there is no clearly defined process usually leads to unrealistic, abnormally complex, and unreadable [21] process models with many states and transitions [32]. These are difficult to analyze, and will not provide relevant information about any of the social groups involved.

For these reasons, it is necessary to adapt process mining techniques to the specific features of SOWs. This study presents a methodology called federated process mining (FPM), which applies process mining techniques to the analysis of complex SOWs based on the concept of software federation: A group of semi-autonomous devices operating together for a common purpose. Consequently, each member of the federation carries out their task individually, before all results are pooled. For the case under discussion, each individual's data will be processed independently. Then, the integrated data of the users that form an SOW, based on common behaviors, will be analyzed. Using the computational capabilities of contemporary smartphones, each individual behavioral process model can be discovered locally. This design decision assumes that the behavior of an individual is not erratic [16], and thus the resulting models avoid unnecessary complexity. By querying smartphones, only the relevant individuals, those that show patterns of conduct of interest, are selected, and their traces are aggregated to generate a workflow for this particular social group. This approach avoids unwanted variability, resulting in higher quality process models of SOWs.

Utilizing smartphones to perform the first phase of the proposed methodology limits the proposal to the analysis of SOWs generated by data stored on such devices. However, the exaggerated presence and constant use of these devices in people's daily lives make them perfect candidates for the analysis of SOWs.

To demostrate how the proposed method can be used and its benefits, we present SOWCompact, a tool for federated process mining that uses existing process mining tools and algorithms in a federated architecture. SOWCompact has been validated with a case study named activities of daily living (ADL) [24], using an existing dataset that contains event logs on ADL from real users.

This research suggests a way to successfully apply process mining techniques to the analysis of SOWs using a federated method to reduce the complexity of the generated models without a quality loss of models. Additionally, by leveraging mobile computing resources, this method reduces the amount of information sent to the process mining servers, which reduces both data traffic and, more importantly, computational load.

The novelty of this proposal lies in the distribution of the social workflow process mining procedure in two phases through a distributed architecture. It leverages of the computational capabilities of the smartphones in performing an initial individual process mining phase. In addition, this distributed architecture enables adopting a new filtering method based on the identification of users that meet a behavior of interest. This approach creates more compact models that represent more accurately the behavior of a specific social group.

The remainder of this paper is structured as follows. Section 2 introduces the motivations behind this study. Section 3 highlights the weaknesses of the current state-of-the-art methods both regarding process mining and social workflows, providing crucial background information on the novelty of our approach. Section 4 describes the federated process mining method and proposes a starting point architecture for tools using this method. Section 5 introduces the so-called SOWCompact, describes its main features, and demonstrates how it can be used to successfully analyze SOWs. A methodology to assess of the proposed approach in any case study is presented in Section 6. In Section 7, the results of applying this methodology to assess the validity of our proposal in the case study described in Section 6.1 are presented. This is followed by an analysis and discussion presented in Section 8. Finally, Section 9 concludes this study and discusses future research directions.

2. Motivation

Social computing (SC) is the area of information technology that deals with the interrelations between social behavior and computer systems [46]. Initially, SC mainly focused on processing and analyzing social information [38], which is defined as the fingerprint left behind by people when performing daily activities. However, the term has progressively evolved toward a wider meaning that includes using computer systems to support any type of social behavior, where humans are the main protagonists; not only as beneficiaries but also as active players [9].

The exaggerated use of smartphones among a large part of the world's population makes these devices ideal for virtually representing their users [30]. Performing SC tasks on users' smartphones allows harnessing the computing power of each user's device while preserving the privacy of the user data [18] and reducing power consumption [4]. We have also worked on the deployment of application programming interfaces (APIs) on such devices to distribute computing processes, which facilitates communication and data requests between the server and smartphones [26].

In the context of SC, social media mining is defined as the process of representing, analyzing, and extracting patterns and trends from social media data. Its objective is to generate detailed profiles of individuals and social groups [35].

Social workflows (SOWs) model describes how a group of people performs a set of activities [17]. The sequence of activities performed by each individual during the SOW is represented by a trace.

Accordingly, activities of daily living (ADL) are the daily routine activities of a person performs [24] (e.g., eating, drinking, walking, or exercising). Studying these activities can be contemplated in SC, because the process that follows a person performing their ADL can be considered an SOW.

The application of process mining techniques to ADL SOWs allows the mining of processes carried out by users in their daily activities. Although each person's daily routine can drastically differ from that of the people around them, often we can observe behaviors that are common to certain groups. Identifying people who are part of a particular group would allow us to establish, for instance, certain behaviors that these people have in common. Thus, the behavior of a particular group would constitute a specific SOW, different from that of any other group. The identification of this SOW and its associated processes may be beneficial in different fields. For example, in medicine, such an approach could ensure the effective monitoring users' health and habits, which would improve the results of preventive medicine and patients' quality of life [49]. This approach may also help identify deficiencies or anomalies in people's behavior [48].

However, while process mining techniques can positively contribute to SC, they remain far from providing satisfactory results. The reason for this is illustrated in Fig. 1, where the model generated using a heuristic algorithm for process mining is demonstrated. This model integrates the traces of the ADL SOWs of only seven different users and monitors their activities for an average period of ten days. Observe that, although the number of users and time frame are limited, trying to discover frequent patterns using that model is difficult. For example, three out of the seven users go *shopping* before preparing a meal (*meal preparation*). This simple pattern that is composed of only two activities is difficult to identify among many different patterns that intersect with it because other users' patterns also include these activities. Furthermore, it is even more difficult to identify other patterns that are met simultaneously with the one initially queried.

To solve this problem, a novel method of pattern extraction is required. Instead of processing the traces of all users, a better solution is to consider only the traces of a group of users that already possess the required pattern. To do this, we need to discover the individual model of the behavior of each user. Then, these individual models are filtered to identify which users exhibit the required pattern.

3. Related work

Before explaining the solution proposed in this study, it is necessary to review the existing literature in the field of process mining applied to social workflows and events carried out by users that correspond to activities with a certain degree of variability. Moreover, using smartphones to carry out these tasks should be discussed.

Process mining techniques are divided into different categories based on their main goals. Process discovery is one such technique that is used to generate a process model based on the behavior captured in a log [1]. This study focuses on this type of process mining to discover the processes contained in the data captured by each smartphone, so that they can be processed further later.

In the process discovery domain, several studies have attempted to leverage the models generated to perform workflow analysis. Van der Aalst et al. [45] proposed a solution based on process discovery to estimate the waiting time on a phone call. These systems have also been used for commonly in recommendation systems. Schonenberg et al. [36] proposed a recommendation service based on a log, in which they applied different methods to determine recommendations.

Recently, with the increasingly exaggerated use of mobile devices, process mining is being applied to problems involving data obtained from these devices. For instance, proposals such as that of Sztyler et al. [43] demonstrate how process mining can be applied to data collected by activity sensors to discover user processes. This approach represents the most frequent relationships between the daily activities carried out by the user and allows us to observe the acceptance and deviation from the model that should originally be followed. The activities performed are detected through sensors connected to the user's smartphone. Another similar proposal uses process mining of data obtained from healthcare sensors connected to the smartphone to self-monitor the user's health [42]. Hwang and Jang [22] used process mining to analyze the changes in customers' movement patterns in a fashion retail store following certain changes in the position of mannequins. Other studies have focused on the safety of mobile applications, for example, Bernardi et al. used process mining to detect whether an application contained malicious software [3]; if so to which malware family it belonged.

However, these studies focus on integrating data obtained from different sources in a server using many already existing tools, where all data are analyzed as a set using a process mining tool. ProM[1] is a widely used graphical interface tool that is compatible with numerous algorithms. Other tools, such as PM4PY[2], offer interface free solutions that allow integrating process

[1] https://www.promtools.org/doku.php

[2] https://pm4py.fit.fraunhofer.de/

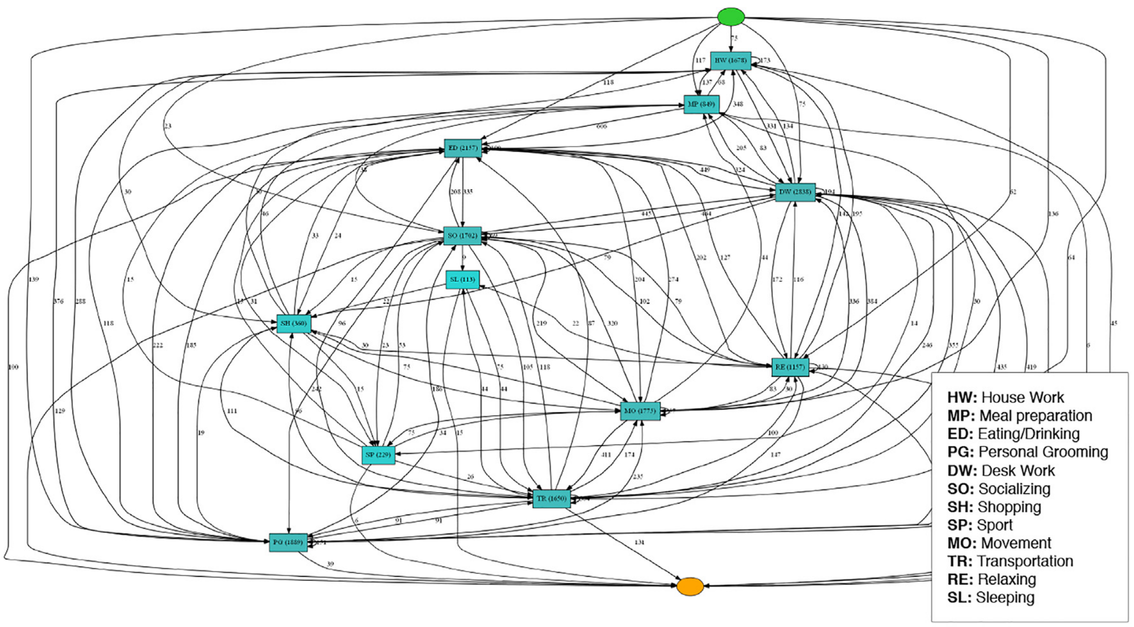


Fig. 1. Spaghetti model generated from a SOW with classic process mining techniques.

mining into other systems using Python. BAB [41] offers a cloud solution for analyzing large amounts of data using its own process analysis algorithms.

However, generating models where all data are integrated on the server implies integrating the data obtained from different users based on very different behaviors. This adds noise to the models, which is a main problem in process discovery applications [2]. In addition, if the amount of data to be integrated is large, the server used for analysis will require a sufficiently large capacity.

Solutions to such problems already exist in the literature. Regarding the noise, there are solutions that alleviate it by extracting frequent behavioral patterns from the generated model. Subsequently, the models become more understandable and of higher quality. These proposal types are encompassed under the term frequent pattern mining. This approach was adopted by Chapela-Campa et al. [8], who suggested using an algorithm over the generated models to identify the most executed parts of the models. The algorithm they proposed was based on the frequency at which the arcs of the entire model structure were executed. The resulting models only contained the most frequently observed activities and relationships. Other proposals for detecting frequent patterns were based on the frequency at which each single arc of the model was executed. These proposals used the values of the arcs computed by tools such as Disco with heat maps, or by algorithms such as Heuristic Miner [47]. Consequently, the newly generated models include only the arcs with a frequency higher than a given threshold. However, considering only the individual weights of the arcs causes problems because the causality between tasks is not considered [8].

Other studies have addressed the efficiency problems that occur in the server. For instance, the proposal of Loreti et al. aimed to solve the problem of analyzing large logs by means of computer clusters [28], using techniques such as MapReduce. Other proposals have tried to leverage the computing capacity of modern mobile devices to perform some phases of process mining. Using mobile devices as process mining tools, that serve as event collectors, was suggested by Medeiros et al. [12]. However, process mining using mobile devices has not advanced along with advances in the hardware technologies of these devices. Future research could follow up on this approach and analyze users' data kept on mobile phones, thus, generating a model for each user on their own devices. This approach of constructing personal models could perform the initial phase of analysis and, if necessary, could be integrated with other user models on a server. This methodology was applied in Kim and Kim's proposal [25], which conducted process mining on a server, using Disco tool, to analyze user behavior through mobile commerce application logs generated in Android OS. Using a hybrid architecture, each user log was analyzed on the device where it was generated. Later, on the server, users with similar shopping patterns could be selected and analyzed together, based on the existence of common patterns in their process model.

Although there are proposals that address noise in integrated processes and proposals to adopt some process mining phases in mobile devices, a full mining of individual user data separately on smartphones has not been conducted. Even though a proposal of this type addresses the two problems previously discussed, to the best of our knowledge, there are no frameworks that can perform of all tasks of process discovery in mobile devices. However, using this type of approach has several advantages. First, once a user's model is generated, it would be possible, for instance, to check which pattern each user met (if known—classification) or which different patterns existed (if not known —clustering). Next, the integration of user data with similar patterns could be performed. Subsequently, possible noise ocurring from analyzing logs containing actions of users with different patterns would be eliminated. Such a proposal could be complemented with current research on frequent pattern mining, which would improve the results offered by both. Second, by first analyzing each user data separately and only analyzing these results in aggregate, better efficiency could be achieved.

Discovering and grouping users based on the observation of their behavioral patterns can be done through process mining clustering techniques, which have commonly been used in practice [13]. For example, Song et al. proposed in [40] using clustering over a profile that characterizes each trace. Thus, the traces of all users are grouped into different clusters according to their profiles. The profile is composed of different features derived from traces, such as number of events and concrete events that appear in the trace. Other proposals, such as that of Jablonski et al. [23], are based on a similar characterization of traces. Cirne et al. [10] suggested a different approach, using clustering techniques alongside model discovery to generate new models with less noise in algorithms such as Alpha.

However, notice that the objectives of these techniques differ from those pursued in this study. If our aim were to create a method to group all existing users based on their different patterns in individual models generated in their smartphones, we would use clustering techniques. However, we aim to be able to ascertain which users meet an already known behavior in the model to select and analyze them exclusively. This is closer to the results that could be expected from the classification techniques applied to process mining than those of clustering techniques. To the best of our knowledge, there are no previous techniques or methods for filtering users based on a behavior of interest, nor for taking advantage of a distributed architecture, based on the federation of different smartphones, which is the goal of our proposal.

Thus, it is necessary to define a proposal that allows conducting an analysis focused on a group of users who share a common pre-determined behavior. Assuming that mobile devices generate most of the information that will be included in the social workflows of social groups to which the user belongs, it would be convenient to utilize the computational power of these devices to optimize the process of selecting which users are part of a social group based on whether they comply with the desired behavior.

None of the analyzed proposals performed this type of task. To achieve a more efficient analysis of social workflows using current process discovery algorithms, it is necessary to introduce a new proposal that covers all these aspects.

3.1. Pattern query languages

Another important topic for which the current the literature must be reviewed is the representation of behaviors of interest, namely Frequent Patterns, in a language that can be understood by all components of federated process mining. Different alternatives have been proposed to specify this language.

On the one hand, several proposals have considered process mining models in a textual language. Frequent patterns can be considered as fragments of the user models, hence, models themselves. Accordingly, solutions based on using declarative languages [33] have been suggested in the literature. Declarative languages represent models as a set of declarative constraints that limit the number of execution alternatives in a process. An example of this can be found in the work of Schönig et al. [37], which proposes a process discovery technique for models represented using the Multi-Perspective Declare (MP-Declare) [7] declarative language. The proposed process discovery technique follows an SQL-based process mining approach. However, after analyzing different alternatives of declarative languages, using a solution based on these languages was discarded. This was owing to their proposal being based on imperative languages [33] to represent the models because of their better expressiveness. Therefore, representing the models and frequent patterns in different language types may not be the best option. Because such applications suffer from and increased complexity when translating models from one representation to another. Other solutions have developed complete frameworks performing queries over process models. This is the case of Polyvyanyy et al. [34], who developed a framework to perform optimized queries over process repositories, where the models regarding which queries will be consulted are stored in. Yongsiriwit et al. [50] proposed a query language that allows for consulting on which fragments of models an activity appears. This language complements an approach aimed at generating a neighborhood context for each activity based on the execution order of the activities involved in the process. More specifically, it extracts the execution order of activities to build a neighborhood context for each activity.

However, there are alternatives such as that of Meddah et al. [29]. Their proposal uses a syntax similar to that of regular expressions to define sequences of events that must appear in the model and connects those events with certain operations. Accordingly, linear temporal logic (LTL) [15] provides a well-defined and widespread language that allows easy representation of event sequences in time, similar to the function of processes.

The underlying premise of regular expressions, with LTL syntax or other, to represent sequences of events is the same as that of declarative languages to represent process mining models in a textual language: A sequence of events that must be met in the model definition as a constraint. Therefore, this sequence of events can be understood as the representation of a partial model that must be contained within a user model or within a model that would be generated from a trace to consider that this user and trace comply with a behavior of interest.

In addition, solutions based on regular expressions provide some advantages over solutions based on declarative languages, that is, they are simpler in defining the rules and integrating the support of this language in the pattern query resolver. Moreover, they benefit the generated model of the individual process mining component of the framework proposed in this study. Therefore, a solution of this type is the most suitable for the proposed method.

4. Federated process mining (FPM)

The challenges of classic process mining techniques to when analyzing complex SOWs resulting from integrating data from users without a common behavior pattern have been presented above. To address these difficulties, we propose using the so-called federated process mining (FPM), where process mining is performed in two steps: First, individual mining; then, mining user groups with a common behavior. The first step allows individuals to be filtered to determine those who comply with the behavior of interest in an SOW. Once the individual data is filtered out, only the SOWs resulting from users who meet the desired behavior are analyzed. This process assumes that each time there is only one group of interest, that is, only the individuals that comply with the behavior of interest are analyzed. Data from users that do not belong to the group of interest are excluded and analyzed only when a new group is formed according to a new searched behavior.

Federated process mining uses the computational power of the user' devices to perform the first phase, which reduces the amount of data that needs to be processed to analyze the SOW. Owing to this filter, federated process mining techniques offer more compact models without loss of quality. Fig. 2 illustrates an example of the model generated for the same SOW in Fig. 1 using filtering. These figures, although difficult to read owing to their size (a higher resolution version is provided as additional material), allow us to compare the readability of the models and visually compare the models generated from the same SOW considering:

- A classic process mining approach, in which all user data are integrated into a single model (Fig. 1).
- A federated process mining approach that filters users and integrates only the traces that comply with the behavior of interest (Fig. 2).

In the second case, 33% of the arcs were reduced by filtering users and traces compared to the model generated using classic non-federated process mining. This means that the average of approximately 10 arcs per event, with 116 arcs in total, is reduced to seven when filtering users and traces, with 78 arcs in total. A higher number of arcs going in and out of an event means a low-precision model that accepts almost any execution order. The sum of the weights of these arcs is also reduced to 23% of that of the classic model. This demonstrates the disadvantage of the models generated with classic non-federated process mining, which may contain any process, while the filtered approach offers results closer to the reality of the concerned users for a given SOW.

The heuristic nets illustrated in Figs. 1 and 2 depict real models generated by applying the two different process mining approaches to the case study data, ADL event logs.

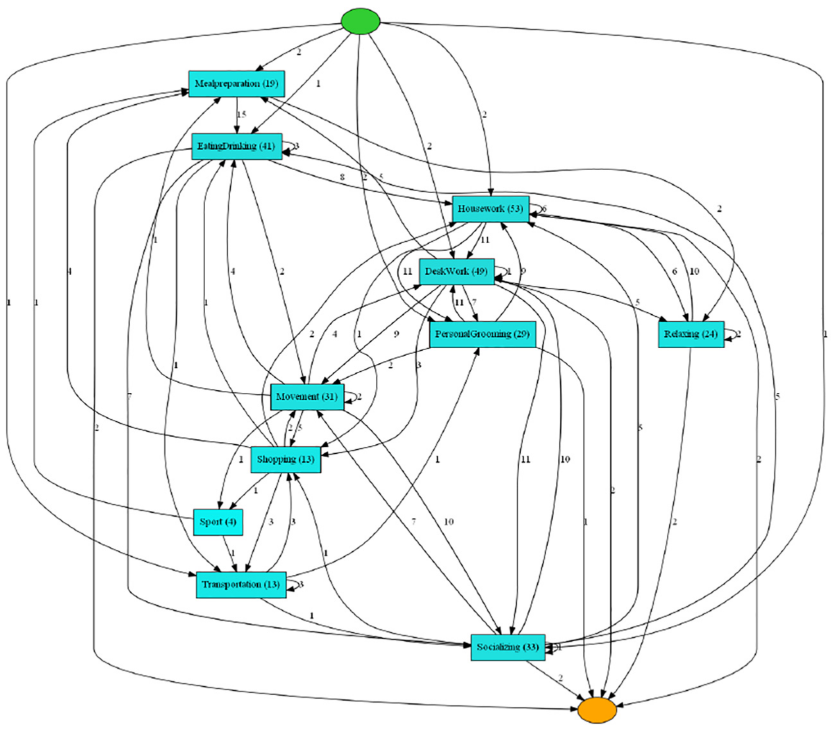


Fig. 2. SOW model discovered using a FPM approach.

This dataset, presented in more detail in Section 6.1, was gathered by Sztyler et al. [43], and stores seven different event logs that represent the actual behavior of seven different individuals for an average of ten days. This includes twelve different daily activities that are common to most people's routines.

4.1. Federated process mining architecture

To facilitate the development of FPM tools, a novel architecture is proposed in Fig. 3, based on three main components: First, the individual process mining component, which integrates the traces of an individual in her smartphone and produces the corresponding behavioral process model; Second, the social process mining component, which selects the users to be analyzed based on the presence of a certain pattern in their behavioral process models. Once it obtains the information of the users that meet the pattern, mining is performed to obtain the process model for the SOW that these users represent; Finally, the federated process mining aggregator (FPM aggregator) component supports the connection between these two components, asking smartphones for the individuals possessing the frequent pattern in their behavioral models and collecting their traces for the social process mining component. This communication is performed using a query language that can be interpreted by both components.

The workflow presented in Fig. 3 is described as follows:

1. The user's smartphone creates an event log by gathering data from the sensors or applications. Movement traces from the GPS, activity levels from the accelerometer, application usage events, or WIFI connectivity are examples of user information provided by the smartphone that can be collected during this stage. The developers of the applications where federated process mining is utilized must implement a module to generate this log.
2. With this event log, the user's smartphone applies a process mining algorithm that generates a model for the user's behavior represented in the event log.
3. Only then, the server can request data from the user event log using this model. To do this, the server defines a query with a pattern (frequent or not) that must be met by the models of the users from whom it will gather data. This query is sent to all user smartphones using the FPM aggregator.
4. When the smartphone of a user receives the query, it processes the user's model to check whether it meets the pattern. If it does, the smartphone processes the event log to select the specific traces that meet the pattern sought after. Traces are not directly processed because this would be inefficient for large event logs or if queries with different behavioral patterns were being sent frequently.

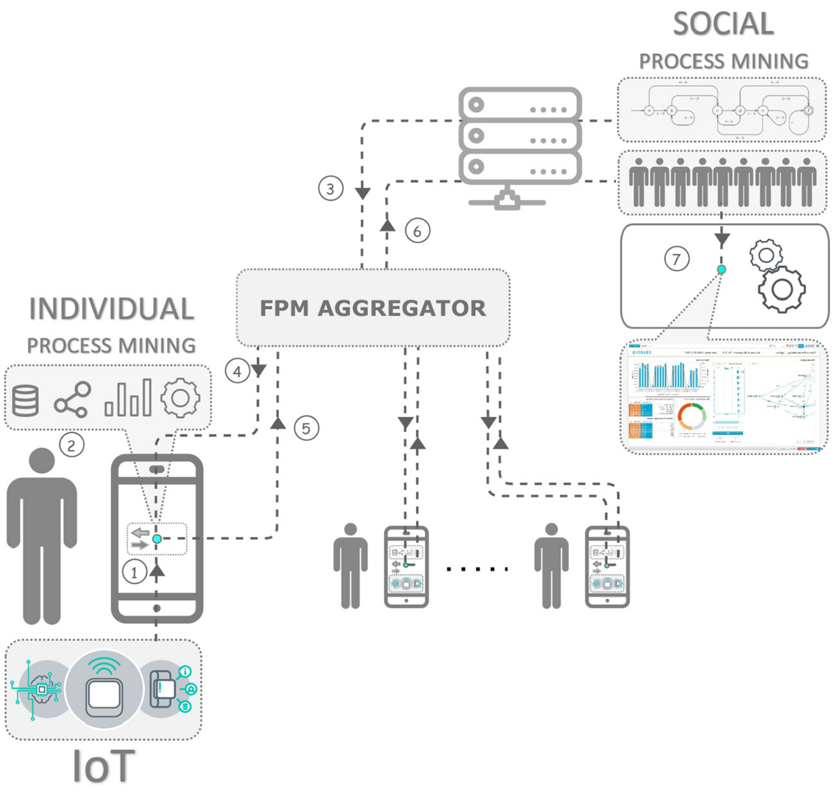


Fig. 3. Architecture for federated process mining.

5. Traces that meet the pattern are sent to the FPM aggregator, along with the query and user to which they correspond. The FPM aggregator is responsible for integrating the data of different users' traces that meet the query pattern.
6. The integrated data are stored in the server.
7. Using these integrated data, a new phase of process mining is carried out. In this new phase, a model representing the behavior of a single social group (SOW) is obtained.

By following this process, a social group is determined based on the users who meet a certain pattern defined in the query sent to the users' devices. As can be understood this process, the behavioral pattern of interest that the users must comply with, which is reflected in the query, must be known to the analyst beforehand, that is, before launching the federated process mining process. Only user data that meet the pattern of interest will be sent for further analysis and become part of the resulting SOW. Therefore, only users that comply with the requested behavioral pattern are selected. Then, the analyst is able to analyze their information and that of the remaining patterns that they have in common if relevant. The patterns that might exist between selected and not selected users are no longer relevant, from the moment that the unselected users do not meet the queried behavior. Therefore, the analysis of their information and other patterns that they might share between them or with the users who do meet the pattern of interest is not necessary.

5. SOWCompact: a federated process mining tool

The federated process mining method and FPM architecture have been described earlier. Therefore, we now introduce the so-called SOWCompact, which is a complete FPM tool. Its components are described as follows:

Individual Process Mining (Fig. 4). This component was implemented as an Android application. It runs on each user's smartphone and is responsible for generating the individual models and checking whether these users meet the patterns that the server seek. To do this, the application employs an individual process mining module, which can be added to any smartphone application that contains an event log to perform the process mining. Therefore, any developer who wants to perform federated process mining using their application only needs to import this module and link it to their own event log files. Developers must generate these event log files outside the individual process mining module, considering the relevant information. Considering that the variety of information with which these event logs can be generated is only limited the creativity of the team behind the application, SOWCompact has not implemented any module that generates an event log in a user application. The only requirement is that event logs must be generated in the XES format [19]. This format, one of the most frequently used in process mining, is an extension of XML, which employs a notation containing a set of well-known tags to represent data as traces of events. Therefore, event logs in this format are visually understandable and widely compatible with process mining tools.

To generate a model from the log file, a version of the Alpha algorithm [11], which considers loops in the traces, is used. This algorithm has been chosen because it has an existing implementation[3] in Java that can be employed in Android applications. Nevertheless, other process mining algorithms can be used instead. For replicability, the algorithm, adapted for use as a library in an Android application, can be found here [4].

Once the model is generated, it is stored in the smartphone and periodically updated to include more recent user behavior.

The other component of the individual process mining module is the pattern query resolver, which processes the server queries received through the FPM aggregator. This module checks whether the user's model meets a given pattern. In this case, it collects traces that contain the pattern and sends them to the server.

The use of a process mining model in this component ensures that all user traces only need to be processed on the smartphone if the specific user meets the desired pattern. Subsequently, traces from users that do not meet the pattern will not be processed again for each received query.

Social Process Mining. This component is deployed on the server side, and its source code can be found here [5]. It is responsible for generating the SOW models with the data integrated by the FPM aggregator using a process mining tool. Specifically, *Process Mining for Python* (PM4PY) was chosen as the mining tool because it is free and widely used in the process mining community. It also allows for automating the process of generating models using Python scripts. However, other process mining tools can be used instead. From the algorithms available in this tool, the heuristic miner algorithm [47] has been used, because it is a suitable for working with real-life data when there are not many different events [20]. According to the first phase of individual process mining, the number of different events that these traces will contain will be smaller if all data are analyzed directly on the server. As is the case with the tool, any other algorithm can be employed instead.

Likewise, this component can be replaced by any other process mining server that performs process mining on event log data. It would only be necessary to add a module to define queries based on patterns and to receive the traces of all the users that meet the requested pattern.

[3] https://github.com/atanasovskib/AlphaAlgorithm
[4] https://bitbucket.org/spilab/androidalpha
[5] https://bitbucket.org/spilab/serverapp

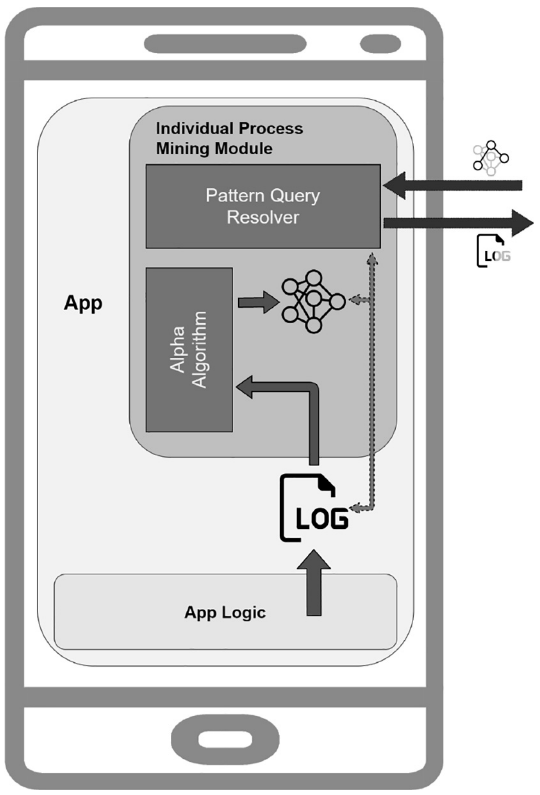


Fig. 4. Internal architecture of the individual process mining component.

FPM Aggregator. This component is deployed between the individual process mining and social process mining components, allowing communication between them. It is responsible for requesting the data that meet a pattern from the users' devices and adding the response data sent to the server from the users who meet the required pattern. The FPM aggregator was implemented using our previous proposal for the deployment of mobile APIs [26].

Pattern Query Language. This language manages the interaction between the individual process mining and the social process mining components through the FPM aggregator. Specifically, it is used to specify event patterns. For this purpose, linear temporal logic (LTL) [15] was selected.

LTL is a well-defined modal temporal logic that allows defining linear time conditions referring to future events. It also establishes a relationship with other formal languages, such as propositional logic, because it utilizes propositional variables and logical operators along with the temporal modal operators it defines. Specifically, the operators that LTL offers are:

- ◯**A** (next): event A must happen in the next state.
- ◆**A** (finally): event A must happen in some of the following state.
- □**A** (globally): event A must occur in all the following states.
- **A** $\mathscr{U}$ **B** (until): event A will continue to happen at least until event B, in a current or future state.
- **A** $\mathscr{R}$ **B** (release): event B will continue to happen at least until and including the moment when event A occurs.
- **A** $\mathscr{W}$ **B** (weak until): event A must occur at least until event B occurs; if B never happens, A must remain forever.
- **A** $\mathscr{M}$ **B** (strong release): event B must happen up to and including the point where event A occurs, which must be maintained in the current or future position.
- **Logical operators**: LTL supports propositional logic, thus, its operators can be used to write the LTL formulas. Some examples of these operators are as follows:
  - **A**∨**B** (or): either event A occurs, or event B occurs, or both
  - **A**∧**B** (and): event A and event B happen
  - ¬**A** (negation): event A does not occur
  - **A**→**B** (material implication): if event A happens, then event B happens
  - **A**↔**B** (bi-conditional):event B happens if and only if event A happens

LTL formulas are created by combining these operators. An example of Pattern Query in this language would be ◆*((A∨D)→ ∘B$\mathscr{U}$C)*. According to this expression, B must occur immediately after A or D and should continue to occur at least until C. Focusing on the ADL case study, an example of query defined with this language would be ◆*((House-*

work∨Deskwork)$\mathscr{W}$*Movement→Sport)*. This query searches for people who do their housework or study/work when they are at home. When they leave the house, they sports.

The chain formed by events and operators that constitute a query is sent from the server to the smartphone as a string. Once a smartphone receives the query, it loads it into the pattern query resolver, which decodes it and offers a series of methods to the other components to access the query data. In this method, the social process mining component allows us to specify patterns in this language to be processed later in the smartphone, accordingly, the individual process mining component understands which pattern the user's model must contain. Nevertheless, the Pattern Query language can be replaced by any other language that allows for the identification of patterns in the models.

6. Validation methodology

This section presents the proposed validation methodology, so that researchers wanting to apply the proposal in different case studies can assess whether it suits their needs. The validation focuses on three main dimensions: *The size and quality of the models discovered*. The first affects the readability of models obtained, while the second ensures that new models preserve their quality. *The amount of data sent to the server*. Because this affects the amount of user information going out of the smartphone, amount of network traffic generated, and amount of information to be subsequently processed by the server, as well as *the execution time*, and because the same hardware, software, and algorithms in the server-side have been employed in all approaches and scenarios evaluated, this metric can help us assess how the FPM-based proposal improves the workload rates with respect to a classical process mining-based approach. Different metrics are considered for each of these dimensions, following a federated process mining approach and a non-federated server-based process mining approach:

- Model size:
  -*Number of activities*: The number of activities, that is, nodes, in the model.
  -*Number of arcs*: The number of arcs, that is, relations between nodes, without considering their frequency.
  -*Sum. arc weights*: The sum of the weights of all arcs, that is, the sum of the frequencies of all relations between nodes.
- Model quality:
  -*Log fitness* [5,6]: A well-known metric in process mining to evaluate the replayability of traces.
  -*Precision* [5,6]: A well-known metric that determines whether a model is precise. A model is deemed when it does not allow paths that do not exist in the event log.
  -*Generalization* [5,6]: A well-known metric that determines whether a model filters too-specific components that are not frequently executed in the processes, if not, the generalization is considered poor.
  -*Simplicity* [5,6]: A well-known metric that determines whether a model's execution paths are clear, that is, a model is simple when the relation between the number of transactions and the number of activities is lower.
- Amount of data sent to the server:
  -*Total size of individual logs*: The amount of information, in kilobytes (KB), sent from smartphones to the server, to be stored and processed using process mining, which corresponds to the sum of the size of each individual log with the user's traces.
  -*Size of integrated log*: The size (in KB) of the log that integrates the traces of the different users into a single event log.
  -*Total size of logs*: The sum of the size (in KB) of the individual logs and integrated log.
- Execution time:
  -*Time needed to generate the integrated log*: The time, in seconds (s), necessary for the server to merge the information of the individual logs in an integrated log.
  -*Time needed to generate a process mining integrated log model*: The time (s) necessary for the server to discover processes belonging to the above-mentioned integrated log.
  -*Time needed to generate process mining individual models*: The time (s) that the server needs to execute process discovery in all individual event logs to generate individual models related to the individual behavior of each user.
  -*Average time to generate process mining individual models*: The average time (s) that the server needs to execute process discovery in each individual event log.

By evaluating and comparing these parameters in both approaches, it is possible to determine whether the proposed tool yielded processes of similar quality and smaller size while reducing space consumption and execution times. To ensure that the analysis was rigorous, a well-defined procedure was followed during validation.

Step 1: Define scenarios. The first step of the validation process is to define scenarios where social workflows are of interest. This is highly dependent on the case study. For example, in our case study, we defined 30 different scenarios, each associated with a different real situation. Moreover, the SOW was analyzed to identify people with a specific behavior, performing certain actions in a certain order during their daily routine.

Step 2: Define queries for the SOWCompact approach. To analyze these scenarios with a non-federated process mining approach, it is necessary to obtain all user data, generate a subsequent process model, and interpret the results obtained from the model. Because all scenarios considered belong to the same case study, the resulting model is the same in all scenarios. However, it can be assumed that the complexity of the model will be greater when the number of users and/or traces is high. This is owing to a highly probable increased heterogeneity of the integrated data.

The situation changes when using a federated process mining approach. In this case, an increase in users and/or traces will not automatically cause an increase in the complexity of the model. Because of the initial filtering on the users' devices, the resulting model is generated only on the SOW of people that meet certain behavior of interest for a scenario.

Once the scenarios are defined, a query that represents a particular behavior in each scenario must be defined to filter users and traces using the SOWCompact approach.

Step 3. Define the environment. The next step is to define the tools to be employed in each of the two approaches.

For the SOWCompact approach, the environment is composed of the tools described in Section 5. The PM4PY tool and heuristic miner algorithm [47] were employed on the server side. To compare the results of both approaches without them being influenced by the algorithm or process mining tool employed, we employed the same methods in the traditional non-federated process mining approach. However, other algorithms and process mining tools can be employed.

Our proposal offers a new application method for process mining tasks, not a new process mining algorithm for the discovery of processes. Therefore, it can be employed along with any classic or new process discovery algorithms.

Step 4. Setup the environment. After defining the environment, it is necessary to deploy it. The environment should be deployed on the same hardware in both approaches, and under the same conditions, for example, operating systems processes, other programs running background tasks, etc.

Using the source code available in the repositories linked in Section 5, the environment is set up as follows:

- *Social Process Mining:* The code in the repository must be downloaded and executed with any Python 3 distribution in a server.
- *Individual Process Mining:* The code in the repository must be downloaded and included as a module in any Android application, the following instructions on the repository. It is necessary to indicate the direction in which the component of social process mining is deployed.

Step 5. Interpretation of the results. The last step is the interpretation of the results obtained using both approaches in the case study and scenarios evaluated. Similar to step 1, this interpretation is highly dependent on the specific case study and should be adapted to the analyzed data. An example of appropriately analyzing the results in our case study is presented later.

Evaluation of the proposal on any case study and with any scenarios is possible if these steps are followed, allowing researchers to test the FPM in different contexts.

6.1. Case study: analysis of activity daily living (ADL) events

The case study selected for this study applies SOWCompact to mining social workflows on activity daily living (ADL).

A case study in the ADL domain was chosen because the events detected represent the day-to-day actions of users, hence, they contain high levels of variability and/or noise. However, despite this variability, there are always actions associated with user habits that are performed mostly in the same manner and order, that is, following the same process. By choosing this case study, the proposal faces the issues that make actual process mining techniques inappropriate for social workflows. Therefore, if the proposal performs correctly in this case study, it can be used for any other case study with less variable data and less noise. For example, the analysis of the process followed by users in their interactions with IoT devices or the process they follow when shopping through an app on their smartphones, etc.

Specifically, we used in this case study the dataset presented by Sztyler et al. [43] because it was generated with data from real users and it is a widely used and referenced dataset [44,39].

It contains ADL data generated by seven different real individuals, all males around 23 years of age. These data are presented as a set of event logs, one per individual, available in a public repository[6]. Each event log stores one trace for each user's day, including the processes the user has followed in his daily activities that day. The time frames of the user activities that are collected in each log are different, spanning from two days for the shortest and sixteen days for the longest. On average, the event logs captured seven users' activities for approximately ten days. They include twelve different daily activities that are common to most people's routines: *house work*, *meal preparation*, *eating and drinking*, *personal grooming*, *desk work*, *socializing*, *shopping*, *sports*, *movement*, *transportation*, *relaxing*, and *sleeping*. For each activity that is performed, an event is generated in the trace when the activity is completed, with the name of the activity that has just been completed.

The 30 different scenarios mentioned before were generated using this dataset. We detail five examples below:

- **Scenario 1:** The manager of a supermarket chain is interested in including a pre-cooked food section in their supermarkets. To do this, they need to know who would be interested in this kind of product. For example, determining if customers who usually go shopping every day before preparing a meal are usually people with little time to spare throughout the day.
- **Scenario 2:** A health science group is conducting research on the lifestyle of people who do not participate in sports. They are interested in finding out what the habits of these people are.

[6] https://sensor.informatik.uni-mannheim.de/#dataset_dailylog

- **Scenario 3:** A healthy habit recommended by health experts is not to park the car near one's home or workplace. Subsequently, the person must walk some distance when going to and leaving the car. It is necessary to analyze whether people who follow this recommendation also follow other healthy habits in their daily routines.
- **Scenario 4:** A local council is considering placing a new taxi stand in one of the local leisure areas to prevent people who meet their friends for a drink from taking their own car. To find out if this makes sense, they want to know if this is a frequent behavior among people in their locality.
- **Scenario 5:** A chain of fast-food restaurants wants to create a new menu for people who follow a healthy lifestyle, with promotions for those group visits. Accordingly, the chain wants to know the habits of those customers who usually go there in groups.

For each of these five scenarios, we established a query as follows:

- **Query Scenario 1** ($\blacklozenge$*(Shopping*$\rightarrow \circ$*Mealpreparation)*)
- **Query Scenario 2** ($\blacklozenge$*(*$\neg$*Sport)*)
- **Query Scenario 3** ($\blacklozenge$*(Movement*$\rightarrow$*Transportation)*)
- **Query Scenario 4** ($\blacklozenge$*(Socializing*$\rightarrow \circ$*EatingDrinking*$\rightarrow$*Transportation)*)
- **Query Scenario 5** ($\blacklozenge$*(*$\neg$*EatingDrinking*$\rightarrow \circ\neg$*Socializing)*)

The formulas used in these different scenarios all seek to represent behaviors that occur at some point in time. Therefore, there is a $\blacklozenge$ operator encompassing the rest of the formula, which represents the behavior sought. As detailed above, the LTL syntax used in the pattern query language supports these and other time-related scenarios.

7. Validation results

In this section we present the results of applying SOWCompact to the ADL case study following the methodology presented in the previous section. For brevity, only the results of the previously mentioned five scenarios are discussed here; the results obtained for the remaining scenarios have been submitted as a Supplementary material.

These, as well as rest of the queries for the 30 scenarios used to validate the proposal, can be found in the supplementary material, including a brief description of each scenario, in the document named *Scenarios and Queries*. These scenarios contemplate partial and total overlapping situations between different queries. The results obtained during the validation of these scenarios regarding different parameters are also included in the supplementary material.

The results mentioned in this section were obtained using the following hardware:

- The individual process mining component (only for SOWCompact) was deployed on Redmi Note 7 smartphones with Android 9, 4 GB of RAM, and a Snapdragon octa-core processor working at 2.2 GHz and OnePlus Nord smartphones with Android 10, 12 GB of RAM, and a Snapdragon octa-core processor working at 2.4 Ghz.
- The server component (in the traditional, non-federated process mining approach and SOWCompact) was deployed on a laptop running Windows 10 with 16 GB of RAM and an Intel Core i7-8550U processor with base and turbo frequencies at 1.8 GHz and 4.0 GHz, equipped with NVMe SSD technology for storage.

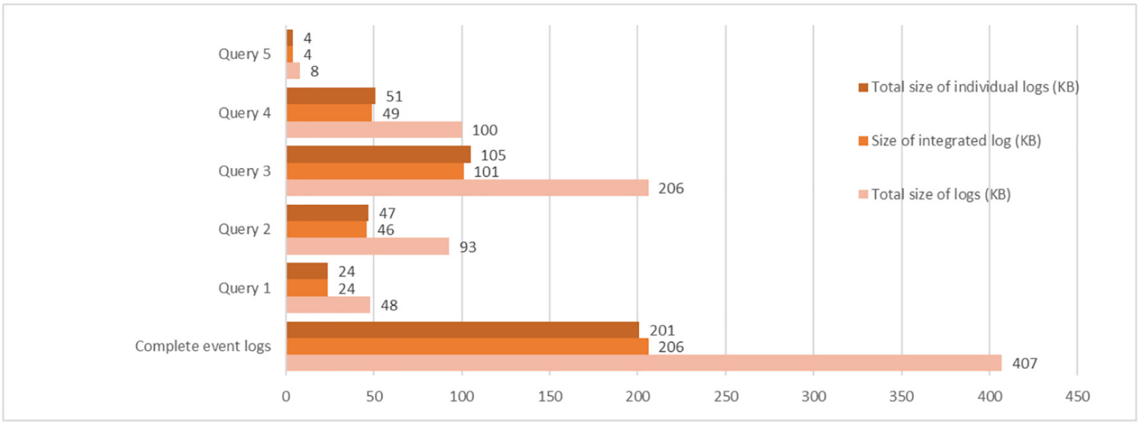
7.1. Data amount and computational load evaluation

We present an assessment of the amount of data sent to the server and the execution time on the server. The analysis of the validation results obtained in this subsection supports our main claim, that is, that we can generate more compact/smaller models without a loss of quality.

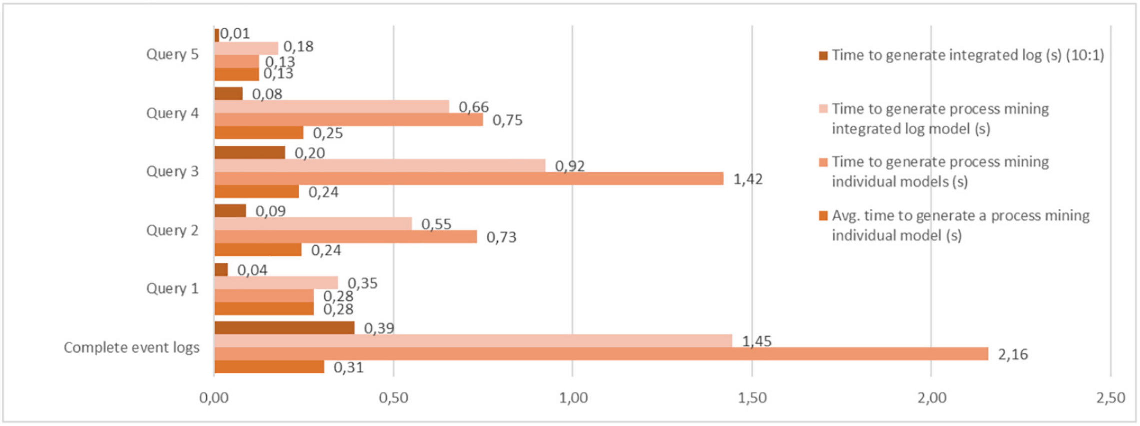
Fig. 5 shows the comparison between the results from the non-federated (*complete event log* values) and SOWCompact approach (*Query 1–5* values), both regarding the amount of data transferred from the users to the server (Fig. 5a) and execution time on the server (Fig. 5b). This comparison is made by measuring different parameters for each approach, and, as will be discussed later, demonstrates the optimization performed by the FPM approach on both data size and computational load for every scenarios. Each of these parameters was selected to cover some of the phases of the workflow described at the beginning of the validation for each of the two approaches (federated and non-federated).

On the one hand, the amount of data has been evaluated in terms of the *total size of the individual logs* sent to the server (full logs in the case of non-federated process mining and filtered logs in the case of SOWCompact), the size of the log generated in the server when the traces of these individual logs are integrated (*size of the integrated log*), and the size of all logs (individuals and integrated) in the server (*total size of logs*).

On the other hand, the computational load was evaluated in terms of the time required to process these data. Specifically, the time required to generate the integrated log in the server (*time to generate an integrated log* with 10:1 scale), time required to generate the process mining model with the integrated log (*time to generate process mining integrated log model*), and total and average time to generate models of individual logs in the server (*time to generate process mining individual mod-*



(a) Amount of data.



(b) Execution time.

Fig. 5. Evaluation of the amount of data and computational load.

els and *Avg. times to generate process mining individual models*, respectively); if this was necessary in the case study in which SOWCompact was applied. The reductions in these times translate into a reduced computational load on the server.

In the case of the non-federated approach, the results are the same for all scenarios evaluated because the users send their complete logs in all cases, regardless of the behavior pattern under consideration. This implies that the same amount of data is sent to the server and same computational load is incurred to process them in all scenarios. Therefore, the results of this approach are shown only once, under *complete event logs*. In the case of the SOWCompact approach, the results for each scenario are different, depending on how restrictive the query is or how common the behavior is. Therefore, the results for each of these scenarios are presented separately. The same applies to the evaluation of the quality and size of the models.

In the bar charts presented in Fig. 5, observe how the amount of data sent to the server is reduced (*Size total of individual logs* Fig. 5a) in all scenarios with the use of SOWCompact. This is because only the data that complies with the required pattern are sent. Therefore, filtering the amount of data sent to the server reduces data consumption on the users' smartphones, as well as the total size of logs and storage needs of the server. Likewise, the computational needs for processing these logs decrease proportionally, as depicted in Fig. 5b. This translates into a reduction in the number of resources required to compute the SOW models.

The percentage by which the amount of data is reduced depends on the frequency of a pattern. If it is a pattern that many users meet or users perform very often, it will result in more traces. Therefore, the size of the sent data will be closer to the size of the complete event logs. On average, for the five queries under consideration, the amount of data sent to the server is reduced to 23% of its original value, that is, from 201 KB to 24, 47, 105, 51, and 4 KB for Query 1–5, respectively. Moreover, the time necessary to generate the integrated log and its process mining model is reduced to 22% and 37% of their original values, that is, from 39.18 ms to 3.96, 8.92, 19.84, 7.93, and 1.48 ms and 1.44 s to 0.34, 0.55, 0.92, 0.65, and 0.17 s, respectively.

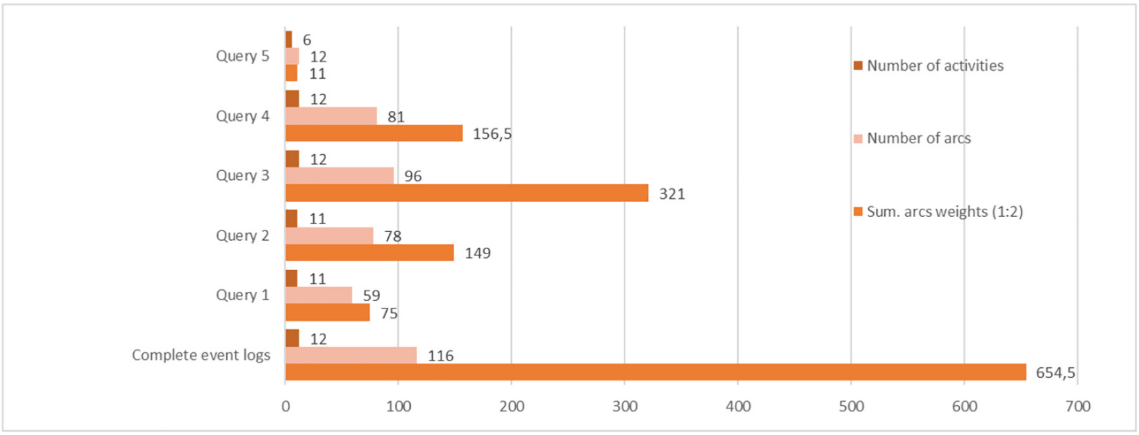
Therefore, it can be concluded that SOWCompact improves the amount of data that needs to be sent to the server and computational load required to process them compared to the use of server-based architectures; as was expected because of the federated nature of the proposal.

7.2. Evaluation of the models

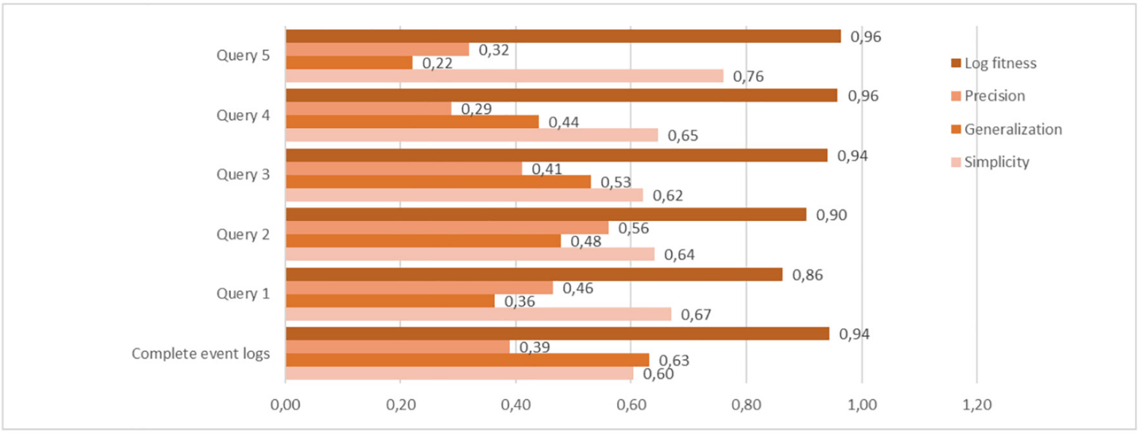
Even though the benefits in terms of computing requirements offered by SOWCompact are very significant, its main advantage comes from the fact that it allows for lighter process models. In this respect, the quality and size of the models discovered by SOWCompact were evaluated against those discovered using a traditional non-federated process mining technique. Four parameters were used to compare quality: *fitness*, *precision*, *generalization*, and *simplicity*. These parameters are widely used to measure the quality of process-mining models. Additional parameters such as number of activities, number of arcs, and sum of the weights of the arcs were included to evaluate model sizes. The visual representations of these models are included as heuristic nets corresponding to the models of different SOWs. This provides a graphic representation of the differences between the models generated by each approach. The full results of this validation, including more quality parameters and visual representation of each model, can be found in the supplementary material. For brevity, only the more relevant parameters are discussed.

7.2.1. Size of the models

The first three parameters that are evaluated are those referring to the size of the generated model. The results of these parameters are presented in Fig. 6a, comparing the server-based (*complete event log* value) and SOWCompact approaches. The values of these parameters are represented by the *number of activities*, *number of arcs* and *sum of arc weights* bars. These



(a) Size of the models.



(b) Quality of the models.

Fig. 6. Evaluation of the models.

values were obtained from the models generated using the PM4PY tool. The weights of the arcs are obtained from the heuristic net generated by the heuristic miner algorithm, which indicates how many times that the transition between the two activities has occurred (in the direction indicated by the transition).

7.2.2. Quality of the models

The remaining evaluated parameters were those referring to the quality of the generated model. These values were obtained using the PM4PY tool as well. The first of these parameters is *fitness*. The PM4PY tool divides into three different values: 1) *Percentage fit traces*, which indicate what percentage of the log traces are perfectly trained according to the model. 2) *Average fitness trace*, which indicates how well the model represents the behavior in the traces (with values between 0 and 1, with 1 being a perfect representation). 3) *Log fitness*, which offers a metric based on the two previous ones (1 when the traces are perfectly fitting). For all measurements, the values of *log fitness* and *average trace fitness* were similar.

The next parameter evaluated is *precision*. PM4PY used a proposal called *ETConformance* [31] to measure the precision. Another parameter evaluated was *generalization*. PM4PY's implementation of *generalization* is based on the work of Buijs et al. [6]. The last commonly used parameter evaluation parameter regarding process mining models is *simplicity*. PM4PY uses this idea to implement its "inverse arc degree" formula for simplicity. The results of these three parameters and *log fitness*, including a comparison of the two approaches, are presented in Fig. 6b.

In addition to the parameters discussed here, other quality parameters are included in the supplementary material. The value of *metrics average weights* is calculated based on the value of the last parameters (*fitness, precision, generalization*, and *simplicity*). This value indicates the overall quality of the model based on the above-mentioned characteristics. In addition, the parameter *fscore* is included, which is calculated based on the accuracy of the fitting traces and accuracy of the non-fitting traces.

7.2.3. Results of evaluating the models

In terms of size, the results obtained using different queries on SOWCompact demonstrate that smaller models can be obtained with a smaller number of arcs, lower arc weights, and fewer activities. This reduction in size is associated with an improvement in the *simplicity* metric. However, although some improvements are achieved in terms of simplicity, for all the scenarios presented, the simplicity increases by 11% on average, increasing from 0.60 in the classical non-FPM model to 0.67, 0.64, 0.62, 0.64, and 0.76 for the five FPM models. These results are less significant than those achieved regarding the model size. This is because, to increase simplicity, the number of arcs in relation to the number of activities must be reduced. SOWCompact goes one step further and filters the arcs and activities, instead of only filtering arcs. For the scenarios presented, on average, the number of activities decreased to 87% of the total, from 12 to 11, 11, 12, 12, and 6 for the five queries discussed. Moreover, the number of arcs decreases to 56%, from 116 to 59, 78, 96, 81, and 12, and sum of the weight of the arcs decreases to 22%, from 1309 to 150, 298, 642, 313, and 22.

For the rest of the quality parameters, the results are similar to those obtained for simplicity. The models generated with the federated process mining approach are mostly similar to the ones generated with a non-federated solution. However, they improve, the quality of the models generated in some cases.

8. Discussion

To discuss whether the SOWCompact proposal is substantially helpful for process mining, we proceed based on the the results obtained in the validation, which are as expected. Moreover, our hypothesis regarding the complexity of interpreting the models generated by the two different approaches has been corroborated. In addition, we have discussed how the benefits of the proposal are linked to the query used to filter the event logs. If a more restrictive query, or one that represents a less frequent behavior, namely, scenario 1, is used, simpler models are obtained with less data sent and computational loads. When more generic queries or queries that represent usual behavior are used, namely, scenario 3, benefits are still observed but on a smaller scale.

This validation was subjected to a robustness check to verify that there are no assumptions that might affect its results when the proposal is applied to a real-world environment. The robustness of the test cases, demonstrating the correct operation of the model in the face of erroneous inputs or stress situations was also tested. For the first type of situation, erroneous inputs were defined as incorrectly formatted queries, which depends on the analyst who defines the queries, or an event log generated in an incorrect format in one of the mobile devices, that is, XES format. In either case, the proposal handled these situations without crashes. In the first case, the data were not analyzed because the pattern of interest could not be interpreted. In the second case, the event log data in the wrong format were discarded. Regarding stress situations, these have not been validated beyond the seven users' devices. However, it has been proven that the system reacts to situations involving many synchronous queries. Specifically, it has been proven that the system works when 1–30 queries from the validation scenarios are performed simultaneously. No limits above the number of queries were tested.

After commenting on the robustness of the validation performed, we also want to note that this section is not intended to discuss the results offered by the algorithm employed to generate the SOW models. This work does not intend to compare the results of using different process mining algorithms because SOWCompact and federated process mining are generally ready to incorporate any alternative algorithm. Therefore, discussions regarding the robustness of individual models generated by the Alpha algorithm or social models generated by the heuristic miner are not considered here. The metrics depend on the algorithm employed [27]. Any other algorithm can be applied in our proposal by replacing the algorithms we have used. Instead, our discussion focuses on presenting the benefits, limitations, and applications of the proposed method, as well as defining its contributions to the field of process mining.

By using SOWCompact it was possible to create simpler models that contained the requested pattern, discarding all the other traces where this pattern was not present. This is achieved by carrying out two process mining phases: first an individual phase, and then a second one on the traces filtered. Filtering out not only the users that comply with a pattern in their individual model, but also their traces, to select only those that contain the pattern sought, means that the users' behaviors that do not comply with this pattern are also discarded. For example, in the case study of this paper, if the aim is to study the users' routines when they take part in sports, the traces that represent routines without sports would be discarded, eliminating their noise from the resulting model.

Producing a smaller model, but with the same quality (as demonstrated in the validation) and still containing the behavior of interest, helps improve the rest of the process mining tasks. Therefore, this proposal should not be considered a different tool for process mining, but as a novel method to improve the results yielded by other process mining tools.

A clear contribution of this method to process mining is the discovery of more readable models (see Figs. 1 and 2) for a visual comparison of models generated during validation with a classical non-federated approach and SOWCompact, respectively. The generated graphical model contained only relevant information about the social group of interest. Information that only adds noise and complicates the visualization of the model was excluded. This makes it easier for analysts to read the behavior represented in the model.

However, its contribution was not limited to this. In real applications, it is probable that even the model obtained using this method will remain complex for visual analysis. Nowadays, tools such as frequent pattern mining help to filter out the most erratic or less accurate behaviors from models. Using the proposed federated process mining as a method to apply these tools can help to improve their results, filtering the traces that are not of interest and looking for frequent patterns on the traces that contain a behavior of interest.

Thanks to all the above, the proposed method opens a new world of possibilities in the analysis of social workflows. First, this proposal enables the analysis of social workflows created from user action data that do not follow a well-defined process —not only in activity daily living (ADL) data, such as in the case study presented here, but also in other domains. For example, it can be used to analyze user interactions with smart devices (IoT). These types of actions can be chaotic and subject to user variability. When this variability is filtered at the individual level, integrating all data from users with different behavioral patterns, the model generated would not adequately represent any user or social group. However, if users were filtered to gather only data from those with a specific behavioral pattern, we would obtain a model that represents the social group formed by those users with a certain common behavior.

As a result, we can conclude that the limitations raised at the end of the related work have been satisfactorily addressed by the proposal: it allows both filtering the users that comply with a behavior of interest and those that do not; it takes advantage of the computational power of smartphones for this purpose, and yields better models—models that take into account the type of improvement that was intended.

However, even accepting the contribution of federated process mining makes to current process mining and its tools, the use of the proposed architecture can still be questioned. In this architecture, various distributed and connected components are needed, as well as the development of process mining tools for smartphones, which, to the best of our knowledge, are not yet widespread. An alternative hypothetical architecture could be proposed, where the same process is carried out in two phases, but on the server instead.

In this alternative version, users would have to send all traces to the server for filtering and processing. However, even without considering aspects such as privacy and other advantages of maintaining the user's virtual representations on their own devices, this would eliminate the improvements of our proposal in terms of the amount of data sent to the server and the server's computational load.

Although with the amount of data in the case study under consideration these kinds of optimizations might not seem efficient, it is important to remember that this case study only contained data from seven users, collected over an average of 10 days. However, for a larger case study, improvements obtained with this system are necessary. For example, it can be supposed that the described case study contained data from 100 million users instead of seven. This number of users, although high, could correspond to that of a social network (where the analysis of SOWs is particularly interesting) with several users below those in the Top 15 of the *Digital 2020: Global Digital Overview*[7]. Sending the logs of these 100 million users to the server would mean sending 2738 TB of data, that is, assuming that the logs were still for just ten days. If we assume that the proportion of users that comply with the pattern and the frequency with which they perform sequences of activities that comply with this pattern are maintained, we can extrapolate the percentages of improvement. In this case, if this architecture is used, instead of 2738 TB only 629 GB, 328 GB, 1423 GB, 684 GB and 54 GB would be sent for each of the five discussed scenarios. If, in addition to increasing the number of users, the time they interacted with the app that generated the data also increased, the size of the event logs would grow even more. Therefore, a distributed architecture similar to that in our proposal has direct repercussions and advantages, both for users and for those conducting process mining.

[7] https://datareportal.com/reports/digital-2020-global-digital-overview

Given the type of advantages offered by Federated Process Mining and in light of the above example, notice that federated process mining, although applicable to any type of scenario, is particularly suited for scenarios with large amounts of data. Scenarios where there are numerous different users presenting different behaviors, integrating all the users' data in the same model would result in a totally random and chaotic model, from which it would be impossible to extract knowledge.

Finally, it is necessary to keep in mind that, during the validation, some aspects of the design of the solution were noticed that are still subject to discussion. For example, in the proposed solution, every time a query is launched, users will send all the traces that comply with the required pattern, independently of whether that query had been previously executed and part of those traces had already been sent. Implementing a mechanism that allows tagging a trace if it has already been sent, to avoid resending it, would help further reduce the amount of data sent to the server (if the same query is repeated periodically).

9. Conclusions and future research directions

The aim of this study is to address the problems inherent to process mining over SOWs on a server, such as the variability of integrated traces and the large storage and computing requirements for the server. For this purpose SOWCompact, a federated process mining approach, is proposed. In SOWCompact the process mining of SOWs is divided into two stages: first one mining on the users' individual data, and a second one mining on the SOW of the users that meet a certain behavior of interest. Considering that smartphones are the best way to obtain a virtual representation of the user and those that are connected to all their services, these devices should collect the user data and perform the individual process mining stage.

Thus, the performance of the server is optimized, reducing both the need for data transfer and the computational load of the server. It also achieves more compact models, which are easier to interpret visually and offer less noise, by reducing the variability in the traces that are integrated with those existing between people with similar behavioral patterns.

To assess the validity of the proposed approach, a federated process mining tool, SOWCompact, was implemented and applied to a case study related to process mining of activities of daily living (ADL) data. The results are demonstrated as a comparison between a classic, server-based approach and our proposed method.

Based on the data obtained in the validation, we suggest that by using the federated process mining tool a clear improvement is achieved in terms of computing time and the amount of data sent to the server. This improvement differs according to how often the users meet the behavior of interest. It has also been verified that the federated approach produces models of the same quality as in the pure server approach, but is more compact, enabling a better visual interpretation of the data. This is based on the fact that the traces have been filtered using one of the simplest algorithms available, the Alpha algorithm, and that the server uses an algorithm capable of filtering by itself, the heuristic miner algorithm. This can be considered the worst-case scenario for FPM, while the best would be the opposite—a good filtering algorithm on the mobile device and a simpler algorithm on the server. This scenario was chosen to demonstrate that, even in the worst-case scenario, the proposal achieves its goal. Therefore, no other process-discovery algorithms were tested. The number of classic process discovery algorithms and new proposals, such as frequent pattern mining or clustering, is so large that the entire spectrum can never be covered. Therefore, considering the worst-case results as a reference is considered a good benchmark.

Moreover, the benefits obtained by this proposal could be further improved by including frequent pattern mining. This would produce models that are better at eliminating the noise of less frequent actions. In this way, models with better *generalization* and greater *simplicity* are obtained. This is one of the future lines of work for our proposal.

Additionally, our proposal opens new possibilities for the use of process mining for software development. First, our proposal can be used to manually and visually analyze data in a social workflow of users within a specific social group, as current process mining techniques and methods already do. However, it also opens the door to creating new automatic systems capable of making decisions, modifying the behavior of the system, or adapting to new situations, based on the information provided by federated process mining. The creation of tools that ensure an easy integration of federated process mining into such systems is one of many possible future research directions.

As additional future lines of work, a mechanism would be needed to update the individual models without regenerating them by including all user traces each time the event log is updated. For instance, incremental process mining allows updating of the model without generating a new one with all traces. Finally, future work should also include checking the improvements of federated process mining if a more intelligent algorithm of process mining in the individual process mining component is applied. In addition, for the actual validation it would be interesting to add an assessment of the energy footprint of the individual mining phase on smartphones.

CRediT authorship contribution statement

Javier Rojo: Writing - original draft, Writing - review & editing, Investigation, Software, Validation, Visualization, Formal analysis. **Jose Garcia-Alonso:** Validation, Methodology, Visualization, Writing - review & editing, Formal analysis, Funding acquisition, Project administration. **Javier Berrocal:** Validation, Funding acquisition. **Juan Hernández:** Methodology, Supervision. **Juan Manuel Murillo:** Conceptualization, Writing - review & editing, Funding acquisition, Supervision, Project administration. **Carlos Canal:** Conceptualization, Writing - review & editing, Funding acquisition, Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the projects 0499_4IE_PLUS_4_E (Interreg V-A España-Portugal 2014–2020), RTI2018-094591-B-I00 (MCIU/AEI/FEDER, UE), and UMA18-FEDERJA-180 (Junta de Andalucía/ATech/FEDER), by the FPU19/03965 grant, by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

[1] W.M.P. van der Aalst, Process discovery: An introduction. In Process Mining: Discovery, Conformance and Enhancement of Business Processes (pp. 125–156). Berlin, Heidelberg: Springer, Berlin Heidelberg, 2011.https://doi.org/10.1007/978-3-642-19345-3_5..
[2] S. Aleem, L.F. Capretz, F. Ahmed, Business Process Mining Approaches: A Relative Comparison, 2015, arXiv:1507.05654..
[3] M.L. Bernardi, M. Cimitile, F. Martinelli, F. Mercaldo, A fuzzy-based process mining approach for dynamic malware detection, in: In 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2017, pp. 1–8, https://doi.org/10.1109/FUZZ-IEEE.2017.8015490.
[4] J. Berrocal, J. Garcia-Alonso, C. Vicente-Chicote, J. Hernández, T. Mikkonen, C. Canal, J.M. Murillo, Early analysis of resource consumption patterns in mobile applications, Pervasive Mobile Comput. 35 (2017) 32–50.
[5] J.C. Buijs, B.F. Van Dongen, W.M. van Der Aalst, On the role of fitness, precision, generalization and simplicity in process discovery, in: OTM Confederated International Conferences On the Move to Meaningful Internet Systems, Springer, 2012, pp. 305–322.
[6] J.C.A.M. Buijs, B.F. van Dongen, W.M.P. van der Aalst, Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity, Int. J. Cooperative Inf. Syst. 23 (2014), https://doi.org/10.1142/S0218843014400012.
[7] A. Burattin, F.M. Maggi, A. Sperduti, Conformance Checking Based on Multi-Perspective Declarative Process Models, Expert Syst. Appl. 65 (2015) 194–211, https://doi.org/10.1016/j.eswa.2016.08.040, arXiv:1503.04957.
[8] D. Chapela-Campa, M. Mucientes, M. Lama, Mining frequent patterns in process models, Inf. Sci. 472 (2019) 235–257, https://doi.org/10.1016/j.ins.2018.09.011.
[9] C. Charron, J. Favier, C. Li, Social computing: How networks erode institutional power, and what to do about it. Forrester Customer Report, 2006..
[10] R. Cirne, C. Melquiades, R. Leite, E. Leijden, A. MacIel, F.B.D.L. Neto, Data Mining for Process Modeling: A Clustered Process Discovery Approach. In Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020, 2020, pp. 587–590. Institute of Electrical and Electronics Engineers Inc. 10.15439/2020F95..
[11] A.A. De Medeiros, B.F. van Dongen, W.M. Van der Aalst, A. Weijters, Process mining: Extending the α-algorithm to mine short loops, 2004..
[12] A.K. De Medeiros, B.F. Van Dongen, W.M. Van Der Aalst, A.J. Weijters, Process mining for ubiquitous mobile systems: An overview and a concrete algorithm, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3272 (2004) 151–165, https://doi.org/10.1007/978-3-540-30188-2_12.
[13] A.K.A. De Medeiros, A. Guzzo, G. Greco, W.M. Van Der Aalst, A.J. Weijters, B.F. Van Dongen, D. Saccà, Process mining based on clustering: A quest for precision, in: In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2007, pp. 17–29, https://doi.org/10.1007/978-3-540-78238-4_4, volume 4928 LNCS.
[14] O. Dogan, C. Fernandez-Llatas, B. Oztaysi, Process mining application for analysis of customer's different visits in a shopping mall. In Advances in Intelligent Systems and Computing (pp. 151–159). Springer Verlag, 2020, vol. 1029. DOI: 10.1007/978-3-030-23756-1_20..
[15] E.A. Emerson, Chapter 16 – temporal and modal logic. In J. VAN LEEUWEN (Ed.), Formal Models and Semantics Handbook of Theoretical Computer Science (pp. 995 – 1072). Amsterdam: Elsevier, 1990.https://doi.org/10.1016/B978-0-444-88074-1.50021-4..
[16] M.C. Gonzalez, C.A. Hidalgo, A.-L. Barabasi, Understanding individual human mobility patterns, Nature 453 (2008) 779–782, https://doi.org/10.1038/nature06958.
[17] S. Görg, R. Bergmann, Social workflows – Vision and potential study, Inform. Syst. 50 (2015) 1–19, https://doi.org/10.1016/j.is.2014.12.007.
[18] J. Guillén, J. Miranda, J. Berrocal, J. Garcia-Alonso, J.M. Murillo, C. Canal, People as a Service: A mobile-centric model for providing collective sociological profiles, IEEE Softw. 31 (2014) 48–53, https://doi.org/10.1109/MS.2013.140.
[19] C.W. Gunther, H.M.W. Verbeek, D. Dolech, A.Z. Eindhoven, C.W. Günther, E. Verbeek, XES-standard definition. Technical Report, 2014..
[20] E. Gupta, A.P. Esmita Gupta, Process mining a comparative study, Int. J. Adv. Res. Comput. Commun. Eng. 3 (2014) 2319–5940, https://doi.org/10.17148/ijarcce.
[21] D.C. Hay, Making data models readable, Inform. Syst. Manage. 15 (1998) 21–33, https://doi.org/10.1201/1078/43183.15.1.19980101/31099.4.
[22] I. Hwang, Y.J. Jang, Process mining to discover shoppers' pathways at a fashion retail store using a wifi-base indoor positioning system, IEEE Trans. Autom. Sci. Eng. 14 (2017) 1786–1792, https://doi.org/10.1109/TASE.2017.2692961.
[23] S. Jablonski, M. Röglinger, S. Schönig, K.M. Wyrtki, Multi-perspective clustering of process execution traces. Enterprise Modelling and Information Systems Architectures (EMISAJ) -, Int. J. Conceptual Modeling 14 (2019) 1–22, https://doi.org/10.18417/emisa.14.2.
[24] S. Katz, Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living, J. Am. Geriatr. Soc. (1983), https://doi.org/10.1111/j.1532-5415.1983.tb03391.x.
[25] S. Kim, D. Kim, Analyzing mobile application logs using process mining techniques: An application to online bookstores, ICIC Express Letters, Part B: Applications 9 (2018) 607–614, https://doi.org/10.24507/icicelb.09.06.607.
[26] S. Laso, M. Linaje, J. Garcia-Alonso, J.M. Murillo, J. Berrocal, Deployment of apis on android mobile devices and microcontrollers, in: In 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2020, pp. 1–3, https://doi.org/10.1109/PerComWorkshops48775.2020.9156208.
[27] S. Leemans, Robust process mining with guarantees. Ph.D. thesis Mathematics and Computer Science. Proefschrift, 2017..
[28] D. Loreti, F. Chesani, A. Ciampolini, P. Mello, A distributed approach to compliance monitoring of business process event streams, Future Gener. Comput. Syst. 82 (2018) 104–118, https://doi.org/10.1016/j.future.2017.12.043.
[29] I. Meddah, B. Khaled, Discovering Patterns using Process Mining, Int. J. Rough Sets Data Anal. 3 (2016) 21–31, https://doi.org/10.4018/ijrsda.2016100102.
[30] J. Miranda, N. Mäkitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, J.M. Murillo, From the internet of things to the internet of people, IEEE Internet Comput. 19 (2015) 40–47, https://doi.org/10.1109/MIC.2015.24.
[31] J. Muñoz-Gama, J. Carmona, A fresh look at precision in process conformance. In R. Hull, J. Mendling, & S. Tai (Eds.), Business Process Management (pp. 211–226). Berlin, Heidelberg: Springer, Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-15618-2_16..
[32] M. Ocaña, Á. Llamazares, P.A. Revenga, M.A. García-Garrido, N. Hernández, P. Álvarez, J. Fabra, D. Chapela-Campa, M. Mucientes, M. Lama, A. Bugarín, J. M. Alonso, Estimation of customer activity patterns in open malls by means of combining localization and process mining techniques. In Advances in

Physical Agents II: Proceedings of the 21st International Workshop of Physical Agents (WAF 2020) (p. 30). Springer Nature.https://doi.org/10.1007/978-3-030-62579-5_3..

[33] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, H.A. Reijers, Imperative versus declarative process modeling languages: An empirical investigation. In Lecture Notes in Business Information Processing (pp. 383–394). Springer Verlag volume 99 LNBIP, 2012.https://doi.org/10.1007/978-3-642-28108-2_37..

[34] A. Polyvyanyy, C. Ouyang, A. Barros, W.M. van der Aalst, Process querying: Enabling business intelligence through query-based process-analytics, Decis. Support Syst. 100 (2017) 41–56, https://doi.org/10.1016/j.dss.2017.04.011.

[35] M. Russell, Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites, O'Reilly Media, 2011.

[36] H. Schonenberg, B. Weber, B. van Dongen, W. van der Aalst, Supporting flexible processes through recommendations based on history, in: In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008, pp. 51–66, https://doi.org/10.1007/978-3-540-85758-7-7, volume 5240 LNCS.

[37] S. Schönig, C. Di Ciccio, F.M. Maggi, J. Mendling, Discovery of multi-perspective declarative process models. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (pp. 87–103). Springer Verlag volume 9936 LNCS, 2016.https://doi.org/10.1007/978-3-319-46295-0_6..

[38] D. Schuler, Social computing, Commun. ACM 37 (1994), https://doi.org/10.1145/175222.175223, Social Computing: any type of computing application in which software serves as an intermediary or a focus for a social relation.

[39] P. Soffer, A. Hinze, A. Koschmider, H. Ziekow, C. Di Ciccio, B. Koldehofe, O. Kopp, A. Jacobsen, J. Sürmeli, W. Song, From event streams to process models and back: Challenges and opportunities, Inform. Syst. 81 (2019) 181–200.

[40] M. Song, C.W. Günther, W.M. Van Der Aalst, Trace clustering in process mining. In Lecture Notes in Business Information Processing (pp. 109–120), 2009. Springer Verlag volume 17 LNBIP. DOI: 10.1007/978-3-642-00328-8_11..

[41] R.A. Sutrisnowati, H. Bae, I.R. Pulshashi, A.D.A. Putra, W.A. Prastyabudi, S. Park, B. Joo, Y. Choi, BAB Framework: Process Mining on Cloud, Procedia Comput. Sci., 72, Elsevier B.V, 2015, pp. 453–460, https://doi.org/10.1016/j.procs.2015.12.126.

[42] T. Sztyler, J. Carmona, J. Völker, H. Stuckenschmidt, Self-tracking reloaded: Applying process mining to personalized health care from labeled sensor data. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (pp. 160–180). Springer Verlag volume 9930 LNCS, 2016.https://doi.org/10.1007/978-3-662-53401-4_8..

[43] T. Sztyler, J. Völker, J. Carmona, O. Meier, H. Stuckenschmidt, Discovery of personal processes from labeled sensor data - An application of process mining to personalized health care, in: CEUR Workshop Proceedings, 2015, pp. 31–46, CEUR-WS volume 1371.

[44] N. Tax, N. Sidorova, W.M. van der Aalst, Discovering more precise process models from event logs by filtering out chaotic activities, J. Intell. Inform. Syst. 52 (2019) 107–139.

[45] W.M. Van Der Aalst, M.H. Schonenberg, M. Song, Time prediction based on process mining, Inform. Syst. 36 (2011) 450–475, https://doi.org/10.1016/j.is.2010.09.001.

[46] F.-Y. Wang, K.M. Carley, D. Zeng, W. Mao, Social computing: From social informatics to social intelligence. IEEE Intelligent Systems, 22, 2007, 79–83. https://doi.org/10.1109/MIS.2007.41. Social Computing: Computational facilitation of social studies and human social dynamics as well as the design and use of information and communication technologies that consider social context..

[47] A. Weijters, W.M. van Der Aalst, A.A. De Medeiros, Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP 166 (2006) 1–34.

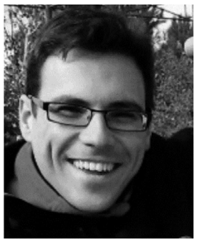
[48] J. Wen, M. Zhong, Z. Wang, Activity recognition with weighted frequent patterns mining in smart environments, Expert Syst. Appl. 42 (2015) 6423–6432, https://doi.org/10.1016/j.eswa.2015.04.020.

[49] A. Yassine, S. Singh, A. Alamri, Mining human activity patterns from smart home big data for health care applications, IEEE Access 5 (2017) 13131–13141, https://doi.org/10.1109/ACCESS.2017.2719921.

[50] K. Yongsiriwit, N.N. Chan, W. Gaaloul, Log-based process fragment querying to support process design. In Proceedings of the Annual Hawaii International Conference on System Sciences (pp. 4109–4119). IEEE Computer Society volume 2015-March, 2015.https://doi.org/10.1109/HICSS.2015.493..

Javier Rojo is a PhD student at the University of Extremadura. He has a predoctoral position at the same university, through a grant for the formation of university professors, from the Spanish Ministry of Science, Innovation and Universities. He obtained his MSc. in Computer Science at the University of Extremadura in 2021. His research interests include software engineering, eHealth, pervasive computing, mobile computing, and gerontechnology.

Jose Garcia-Alonso is an Associate Professor at the University of Extremadura and co-founder of Gloin, a software-consulting company and Health and Aging Tech, a eHealth company. He got his PhD on software engineering at the University of Extremadura in 2014. His research interests include software engineering, mobile computing, pervasive computing, eHealth, gerontechnology.



Javier Berrocal received the Ph.D. degree in computer science from the University of Extremadura, Spain, in 2014. In 2016, he obtained an Associate position at the University of Extremadura. His main research interests are mobile computing, context awareness, pervasive systems, crowd sensing, the Internet of Things, and fog computing. He is a cofounder of the company Gloin, which is a software-consulting company.

Juan Hernandez received the Ph.D. degree in Computer Science from the Technical University of Madrid, Spain, in 1995. He is currently a Full Professor at the University of Extremadura and the Head of the Quercus Software Engineering Group. His research interests include service-oriented computing, ambient intelligence, and model-driven development.

Juan Manuel Murillo is a co-founder of Gloin and a full professor at the University of Extremadura. His research interests include software architectures, mobile computing, and cloud computing. Murillo has a PhD in computer science from the University of Extremadura.

Carlos Canal is a full professor at the University of Malaga, Spain. His research interests include mobile and cloud development. Canal has a PhD in computer science from the University of Malaga. Contact him at carloscanal@uma.es.