Extraction of User Daily Behavior from Home Sensors through Process Discovery

Marco Cameranesi, Claudia Diamantini, Alex Mircoli, Domenico Potena, and Emanuele Storti

***Abstract*—In the last years, the wide availability on the market of low-cost smart devices paved the way for the development of smart environments, which offer an unprecedented opportunity to recognize patterns of activities from the large amount of collected data, with the ultimate aim of monitoring user behavior. In this paper, we propose a methodology which relies on Process Discovery techniques to analyze sensor data in terms of activation sequences and to discover process models representing user's behavioral patterns. The extraction of such models is valuable not only in the perspective of gaining a better insight on how a certain task is performed, but also in supporting novel smart services. In order to evaluate the effectiveness of the approach, in this work we also consider a real-world case study set in an ambient assisted living environment.**

***Index Terms*—process discovery, ambient assisted living, internet of things, smart environments, sensors.**

I. Introduction

In recent years, much effort has been put in the design and development of *smart environments*, that aim to improve people's quality of life providing them with services supporting their daily activities [1]. The term smart environment, also related to the pervasive/ubiquitous computing paradigm [2], refers to an environment that has been equipped with a number of sensors and actuators of different types (e.g., monitoring sensors, RFID tags, power-line controllers) for a variety of applications ranging from health monitoring to security, light management and energy saving. Logging the value of certain classes of sensors allows to track the activities performed by a user. Starting from these activities, it is also possible to identify each task fulfilled by a user in the environment, as it can be considered as a sequence of activities. For instance, when a user performs a cooking task in a house, he/she moves towards the light switch to turn it on, opens the fridge and after a while he/she switches on the oven to warm the food. Each of these activities triggers several sensors (e.g. doors, motion sensors) and a sensor activation sequence can be seen as the execution, or a trace, of an (implicit) process.

Making sense of this information may be useful to recognize actual performed activities and, more in general, to derive typical patterns or processes capable to describe the user behavior in performing some tasks, or macro-activities, such as cooking. Due to the huge amount of data that can be potentially generated, analysing sensor data and deriving a workflow/process model from them must be however supported by techniques capable to provide a summarization of event logs monitored by sensors. In the Literature, Process Mining and in particular Process Discovery techniques and algorithms have been used to derive a process model representing a set of given traces, to discover *behavior models* that represent the "standard" behavior of the monitored user in the form of a process model, where activities are tasks or macro-activities, e.g. cooking, reading, watching TV for a smart home. This type of models can be effectively used to get a better understanding of how a user behaves in a given environment. As also mentioned in [3] three research challenges in the specific application of Process Discovery techniques are indeed (i) the gap between sensor logs and tasks employed in Process Mining, (ii) designing process modeling and mining techniques fitting the variability of human habits and (iii) the segmentation of logs.

With such an aim, this paper presents a methodology for user behavior analysis that exploits Process Discovery techniques to derive activity models from sensor activation logs in a smart environment. Unlike other work in the Literature, the purpose of this approach is to derive descriptive models able to represent user behavior. The methodology proposed in this work is helpful in both human activity recognition and daily behaviour monitoring. On the one hand, it helps to identity a macro-activity on the basis of the activation sequence of sensors and to gain a better insight on how a certain task is performed; on the other hand, it supports novel smart services. Among them, for instance, adaptive environments able to align parameters to the specific behavior of users. These models can also be used to support the monitoring of human behavior and the identification of anomalies, i.e. sequences of activities that are not aligned with the model generated from sensor data related to past process instances. According to the specific environment, their recognition can be indeed useful to react more quickly to emergencies due to accidents or health issues in a house or a factory, or possible threats in a public area.

The proposed approach includes these steps:

- event logs generated by sensor data are characterized in terms of macro-activities, annotated by the user, and micro-activities (i.e., sensor activations);
- logs are preprocessed in order to remove noise and make them compliant with the adopted techniques;
- Process Discovery techniques, specifically infrequent inductive mining, are applied over the event logs to derive process models for activity flows.

The original contribution of the paper consists in the adoption of Process Discovery techniques to represent user activities as a process, where each activity is a sensor activation. While the application of Process Discovery to derive activity models is reported by some work in the Literature, as shown

M. Cameranesi, C. Diamantini, A. Mircoli, D. Potena and E. Storti are with the Department of Information Engineering, Universitá Politecnica delle Marche, Ancona, 60131 Italy (email: $m.cameranesi@pm.univpm.it$, $\{c.diamantini, a.mircoli, d.potena, e.storti\}@univpm.it$)

in the next section, to the best of our knowledge this is the first work that proposes the derivation of a model for user activities from logs related to sensor activations. By helping to better and more deeply understand the intrinsic workflows behind a given macro-activity, e.g. having a shower, in terms of activated sensors, useful insights can be derived, such as recognizing trends or spotting deviances from the model.

As a further contribution, in order to assess the effectiveness of the proposed approach, we include an experimentation conducted on a real-world case study from the CASAS project of the Washington State University, specifically dealing with ambient assisted living environments. In particular, in the project the behaviors of a user and her pet have been monitored for almost two months through multiple sensors deployed in a smart home. The retrieved data have been used in this work as a basis for the application of the approach.

An early version of the approach presented in this paper is available in [4]. This work builds on our prior study including a more formal description of the approach and its extension towards a domain-independent methodology, also including an extended discussion on the preprocessing phase, a thorough extension of the case study and of the experiments and a larger analysis of related work.

The rest of the paper is structured as follows: next section is devoted to discuss relevant related work, while in Section III we provide some preliminaries on Process Discovery. Section IV includes a detailed description of the proposed methodology. A case study is presented in Section V together with experimental results. Finally, Section VI draws some conclusions and presents possible future work.

II. Related Work

Most work in the Literature dealing with user behavior analysis in smart environments refers to domotic scenarios in the context of smart homes. In the Ambient Assisted Living (AAL) field, some work in the last years has focused on recognizing user's activities starting from raw data collected through a set of sensors. AAL community classifies this work into two different groups: Activities of Daily Living (ADLs) and Instrumental Activities of Daily Living (IADLs). The main difference is that in IADLs there is an interaction between the user and electronic devices (e.g. phone, home appliances), usually through a user interface, contrary to ADLs where there is not such kind of interaction. In [5] authors propose an approach aimed at predicting a macro-activity from values collected by simple and cheap sensors. The prediction is performed by using an extension of the Naïve Bayes classifier that also takes into account temporal relationships. Experiments are conducted on a dataset composed of 436 activities divided in 16 classes. The limit of the resulting classification model is that it is not able to describe the user behavior, but only to recognize that a trace is an execution of a given activity.

In [6], authors propose an unsupervised approach aimed at discovering interesting ADLs patterns and compressing the resulting patterns through a clustering technique. The introduced solution is able to identify a pattern independently from the time intervals between the activation of two sensors. It means that the algorithm is able to identify a pattern even if the steps of the same activity are performed with different time duration by different users. We highlight that also the approach proposed in this work is time independent, because timestamps are only used to identify an ordering among activities. A further contribution in the field of activity recognition is described in [7], where authors explain how monitoring ADLs at home can be exploited to predict clinical score of inhabitants.

The approach proposed in [8] is based on a pattern mining algorithm which is able to identify activity patterns by also considering unexpected changes in the users' behavior and abnormalities occurred within a specific threshold, as typically happens with Alzheimer's patients.

Another methodology is introduced in [9], where authors propose to recognize daily routines as a probabilistic combination of activity patterns. They show how probabilistic topic models can be exploited for the automatic discovery of patterns.

In the last years, researchers have started to address the problem of activity recognition with a novel approach which is based on Process Mining techniques and algorithms [10]–[15]. These approaches basically share the view that every activity executed by a user can be seen as a step of a process instance, namely a way in which a process can be executed. In particular, in [10], the use of Process Mining to perform several analyses on activities performed by a user is introduced. Such an approach shares with our work the analysis of user behavior, defined as a process model of macro activities, but it logs activities through smartphones and smart watches. The adoption of personal and/or wearable devices introduces a simplification in the analysis, as it provides more precise data about user behavior (i.e., they minimize the presence of noise due to the activation of multiple sensors or the presence of more than one user).

In [11], authors proposed a system to analyse the behavior of health professionals in a surgical area of an hospital. As initial step, an indoor location system (also known as Real-Time Location Systems or RTLS) is used to collect data, that are then assigned to a (macro) activity. In the end, Process Mining techniques are exploited to discover a model able to represent and describe the staff behavior. The authors developed a specific tool designed to use Process Mining techniques with indoor location system data. In particular, they adopted the PALIA [16] algorithm which exploits syntactical pattern recognition techniques to build a formal automaton, known as Timed Parallel Automaton (TPA) [17], i.e. a safe Petri net that can represent the process.

In [12] authors propose an interesting approach focused on a preprocessing step aimed at mapping sensor data to event logs based on domain knowledge. In particular their approach is devoted to transform precision raw sensor measurements containing locations and timestamps of process entities into standardized event logs comprising process instances. To this aim, the authors introduce the definition of "interaction" which acts as intermediate knowledge layer.

Another contribution in the field of activity recognition comes from [13]. Authors address the problem of the identification of habits related to multiple inhabitants living within the same house, proposing a solution based on the Fuzzy Miner [18] algorithm. In contrast to our methodology, all the above mentioned work do not deal with the discovery of macro-activity models that can be extracted from sensor data.

Among the approaches focused on the way of grouping low-level events into higher-level activities, in [14] authors propose a supervised event abstraction method, where behavioral activity patterns are used to capture domain knowledge about the conjectured relations between high-level activities and recorded low-level events. Instead of inferring such relations from data, however, activity patterns encode assumptions made by experts on how high-level activities manifest themselves in terms of recorded low-level events. The supervised approach of [15], although not aimed to derive behavior models, takes a similar approach in extracting high-level user operation from low-level software execution log.

III. Background

This Section aims to introduce the notation used hereafter in the paper and the main concepts behind the Process Discovery techniques adopted in the methodology.

A. Preliminaries

In this work we refer to a quite general notion of process, as a flow of activities, each of which performed by some resource and aimed to reach a certain goal.

In order to formally characterize a process instance, we provide in the following the notion of event, trace and event log. In particular, we refer to a simple notion of event described through a set of properties. Although we do not constrain the set of properties, hereafter we assume at least three properties, respectively related to: the activity performed, the resource (or agent) who performs the activity and the time at which the activity is executed.

Definition 1. *(Event) Let A be the set of activities, R the resource domain and T the time domain. An event σ is a tuple $\langle a, r, t \rangle$, where $a \in A$ is an activity performed by a resource $r \in R$ at a certain timestamp $t \in T$.*

Example. Let us consider an agent named *Paul* $\in R$ executing a certain *kitchen activity* $\in A$ at 8:45 on May, 2nd 2019. The corresponding event would be represented by the tuple ⟨ Kitchen_activity, Paul,'2019-05-02 08:45:00' ⟩.

Definition 2. *(Trace) A trace l is a finite non-empty sequence of events $\{\sigma_1, \dots, \sigma_n\}$ that are temporally ordered and such that each event appears only once, i.e. $\nexists \sigma_i, \sigma_j$ with $i \neq j / \sigma_i = \sigma_j \wedge \forall i, j$ with $i < j, t_i \leq t_j$.*

An *event log* is the dataset recording all past executions (i.e., instances) of a process. Although instances can include parallel execution of activities, event logs store only the *trace* of a process instance, i.e., the sequence of the activities stored according to their temporal order of occurrence.

Definition 3. *(Event log) An event log $\mathcal{L}$ is a set of traces such that every event appears at most once in the entire log, i.e. $\forall \sigma \in \mathcal{L}, \nexists l_i, l_j / \sigma \in l_i \wedge \sigma \in l_j$*

TABLE I
List of events related to the *Morning Activities* example

	activity	resource	timestamp
σ_1	a	r_1	2017-11-10 10:24:43
σ_2	b	r_1	2017-11-10 10:57:19
σ_3	c	r_1	2017-11-10 11:09:32
σ_4	e	r_1	2017-11-10 12:23:54
σ_5	a	r_2	2017-11-10 8:15:23
σ_6	d	r_2	2017-11-10 8:26:08
σ_7	e	r_2	2017-11-10 8:54:34
σ_8	a	r_1	2017-11-11 9:30:15
σ_9	c	r_1	2017-11-11 10:02:08
σ_{10}	b	r_1	2017-11-11 10:43:27
σ_{11}	e	r_1	2017-11-11 11:31:35

Example. Let us suppose a process related to the morning activities performed in a house, with a set of activities $A = \{a, b, c, d, e\}$ and a set of resources (i.e., people living in the house) $R = \{r_1, r_2\}$. Let us consider the following traces which include the sequence of events as recorded by a set of sensors installed in the house: $l_1 = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, $l_2 = \{\sigma_5, \sigma_6, \sigma_7\}$ and $l_3 = \{\sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}\}$ where each event σ_i is defined as in Table I.
From such an event log it is possible to recognize that every instance of process (as recorded by a corresponding trace), starts with activity a and ends with activity e. The sequence of activities between those two points seems to differ from trace to trace, although some regularities can be spotted e.g., c and b always appear together. In general, real-world event logs include thousands of single events and a manual analysis is not feasible. For such reason, a much more efficient and effective way to look through the possible process flows is a process model.

Processes are frequently represented through models representing the control-flow of the performed activities, that is their ordering relation. Among the several formalisms available to represent process models, in this paper we refer to Petri nets, i.e. state-transition systems that can be represented as directed bipartite graphs where nodes represent transitions (i.e. events that may occur, graphically represented by bars) and places (i.e. conditions, graphically represented by circles).

Definition 4. *(Petri net) A Petri net is a tuple $\langle P, T, F, m_i, m_f \rangle$, where P is a finite set of places, T a finite set of transitions such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ the flow relation between places and transitions (and viceversa), $m_i, m_f : P \to \mathbb{N}$ are the initial and final marking representing the distribution of tokens. A firing sequence for a Petri net N is a sequence of transitions.*

To model the workflow of process activities, a subclass of Petri Nets with additional constraints is usually considered, namely Workflow Nets (WF-Nets) with a single input place with no previous transitions and a single output place with no following transitions. The following property is defined for a WF-Net [19].

Definition 5. *(Soundness) A WF-Net is sound if and only if these requirements are satisfied: (1) option to complete: it is always possible to reach the state with a single token in the*

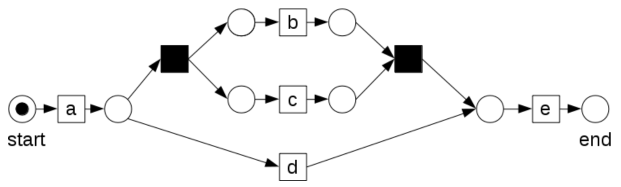

Fig. 1. Petri net for the example.

output place, (2) proper completion: if the output place is marked all other places are empty, and (3) no dead transitions: there is a firing sequence for every activity.

Example. In Figure 1, we show the schema for the morning activity process, represented through a Petri net. Please note that circles represent places, while transition are represented as boxes. In particular, white boxes correspond to activities, while black boxes are invisible transitions, which do not correspond to any specific activity but are needed to obtain a sound net.

The quality of a process model can be determined through several different metrics. Among them, fitness is one of the most relevant, as it evaluates the ability of the model to replay traces within the log. Fitness can be computed by comparing the event log with the discovered model through conformance checking algorithms [20]. A fitness value close to 1 means that each trace in the log can be associated with an execution path specified by the process model, while lower values are typical of models that are not able to represent some traces or portions of them. This metric is a key aspect in the evaluation of model quality since an unfitting model (i.e., a model with low fitness) is not able to represent all the recorded events. Furthermore, in this work we consider also the simplicity as a metric, which is a qualitative measure used to evaluate the complexity of the discovered process model.

B. Process Discovery

If a process model is not available for the process at hand, it can be derived from the event log by using Process Discovery techniques. The most common discovery techniques include Alpha Miner [20], Fuzzy Miner [18], Heuristic Miner [21], and infrequent Inductive Miner [22].

The Alpha Miner algorithm has been designed for discovering process models starting from "clean" conditions, that is noiseless logs. It gives good results with structured processes (known in Literature as *lasagna-like* processes), whereas it fails in the discovery of valid models for highly variable processes (known as *spaghetti-like* processes). In fact, in the latter case the Alpha Miner often returns overgeneralizing models (known as "flower" models), which do not provide useful knowledge about the process as they suffer from an underfitting problem [23].

The Fuzzy Miner algorithm filters and groups the most infrequent behaviors providing as output a high-level (i.e., abstract) representation of the extracted model. The outcome of such an algorithm is a scheme involving just the most relevant activities, displayed as single activities or aggregated in clusters, and their precedence relations. Moreover, Fuzzy Miner is not able to represent split and join constructs. It means that the resulting models do not have an executable semantics.

The Heuristic Miner can be thought as an extension of the Alpha Miner algorithm. The latter is able to find all causal relations among activities within the log, whereas the former takes into account the frequency of relations and applies some heuristics to determine relevant sequence and parallelism relations.

Finally, the infrequent Inductive Miner returns a model by iteratively refining a process tree, where each node represents either an activity of the process (leaf node) or an operator (branch node). Similarly to the Heuristic Miner, this algorithm filters infrequent behaviors and the output can be represented by a Petri net. The infrequent Inductive Miner guarantees to always return a sound model, i.e. a model where all process steps can be executed and the final marking of the Petri net is always reachable. The filtering level of the infrequent Inductive Miner depends on a user-defined threshold, which ranges from 0 to 1; the value of 0 indicates that the infrequent paths are not filtered at all.

IV. Methodology

This section is devoted to discussing the methodology for the extraction of behavioral patterns from an event log produced by sensor data in a smart environment. The approach relies on an event-based interpretation of sensor data, according to which a sensor[1] produces data when an event occurs.

For such a reason, Process Discovery techniques can be used to derive process models as behavioral patterns, since the flow of sensor activations can be represented as a trace, and the entire recording of sensor activations as an event log. In this work, we consider some minimal requirements for each recorded event in the log, namely its case id, the performed activity and its timestamp.

As introduced in the last section, in Process Mining the term *case id* stands for the identifier of a specific process instance. In this context, it represents the execution of a certain human task, allowing to group by *case id* all events performed for its accomplishment. On the other hand, the term *activity* refers to an action performed by a *resource* (or agent) in an organisation. Although event logs can be specified at any level of abstraction, most applications focus on a single, specific level. In this work we consider two levels of abstraction for the specification of activities, namely high- and low-level activities. In order to distinguish them, we refer to the term *macro-activity* to define a high-level task performed by a user, e.g. cooking, having a shower or delivering a package.

On the other hand, we name *micro-activity* a low-level activity measured by sensors deployed in the environment and caused by a specific macro-activity performed by a resource. For instance, in a smart home scenario, the macro-activity "having a shower" will trigger one or more sensors, such as a motion sensor on the bath door, a presence sensor in the room, and so forth. To give another more detailed example,

[1]Specifically, the class of sensors that produce data when triggered by an external cause.

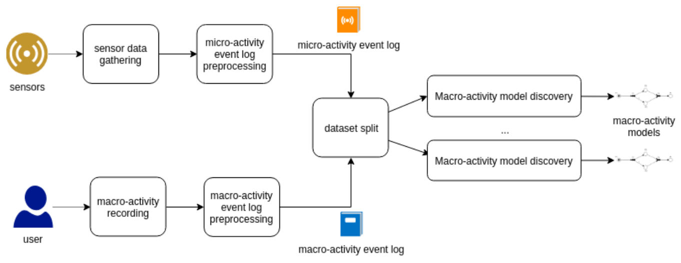

Fig. 2. Workflow for discovery of macro-activity models.

when doing the macro-activity *cooking*, the user may (1) turn the kitchen light on, then (2) open the fridge to take out food, (3) close the fridge, (4) switch on the oven, (5) open the oven door to put the food to warm inside, (6) close the oven door, and after a while (7) open the oven again to take out the food and finally (8) close the oven door again, (9) turn it off and (10) switch the kitchen lights off. Those sensor activation are micro-activity that can be recorded in an event log and that will share the same case id related to the execution of the macro-activity.

It is important to notice that there is no functional dependency between a micro-activity and a macro-activity performed by the resource: a macro-activity can trigger more than one sensor at a time, while a specific sensor may be possibly activated by different macro-activities. To make an example, the presence sensor located in a room can be triggered by opening the door but also by a user moving in the room.

Micro-activities are typically associated to a specific time instant and as such can be considered atomic. On the other hand, macro-activities have a duration specified by a start and an end time. The *sequence* of sensors triggered during the execution of a macro-activity, i.e. the corresponding set of micro-activities, can be viewed as the execution of a process. The related process model represents the model of the macro-activity in terms of micro-activities, namely the *macro-activity model*.

A major challenge regards how to group performed activities into process instances, i.e. how to identify that a set of recorded events in the log belong the same case. To this end, we propose a supervised methodology for the extraction of macro-activity models through the application of Process Discovery techniques to sensor activation logs. The methodology is summarized in Figure 2 and consists of three main phases, namely (1) sensor data gathering and preprocessing, (2) macro-activity recording and preprocessing, (3) dataset split and macro-activity discovery. They will be discussed in the following subsections.

A. Sensor data gathering and preprocessing

The phase introduced in this subsection aims to gather sensor data and process them in order to produce as output a micro-activity event log. In this work we do not make particular assumptions on the type of sensors deployed in a smart environment, provided that they generate data as a consequence of some human actions. In case sensors producing data independently of human intervention are included in the log, e.g. a temperature or a humidity sensor whose outcomes are not typically affected by the activities done by resources, they need to be filtered out during the preprocessing step, because they give no valuable information for the macro-activity discovery.

Sensor data can be analyzed through Process Discovery techniques if, at least, the *case id*, the name of the performed *activity* and the *timestamp* are reported. Typically, a record from a sensor is characterized by the sensor name (or id), the timestamp, and the value returned by the sensor. The *case id*, which is a very important element as it allows us to distinguish among different executions of the same process, is usually not provided with sensor data. Indeed, the sensor does not know who activates it (the user is recognized only in rare cases, e.g., using RFID technologies or wearable devices) and for which purpose it was activated (e.g., the motion sensor in the kitchen can be activated if the user is cooking or is just opening the fridge for a soda). A survey of possible approaches for assigning case id to activities, and thus grouping them in process instances is available in [24]. In our work we use two definitions of case id based on the kind of analysis to perform. In micro-activity event logs, a case is each execution of the specific macro-activity at hand. To address the issue related to map sensor measurements to macro-activities here we assume that the user has been asked to annotate the activity she is performing, e.g. reading, watching TV, taking medicines and so on. In details, the user has to manually specify when she starts and ends the performed macro-activity.

In the micro-activity event log, an activity is defined by the name of the sensor and its value. For instance, when the main door sensor reports the values OFF, the activity ”closing main door” is captured. More precisely, for sensors returning categorical values (e.g. door sensors, motion sensors or in general sensors returning a status) we set the activity as the pair $\langle$*sensor name, sensor value*$\rangle$; while for sensors returning continuous values the values are first grouped in a given set of intervals (based on frequency, fixed intervals or using any clustering techniques), each of which is assigned a label, then the activity is given by the pair $\langle$*sensor name, label*$\rangle$. In the case of devices containing multiple sensors, a set of events with the same timestamp is generated.

Hence, the preprocessing phase involves:

- cleaning sensor data by removing records having missing values;
- transforming sensor data into *micro-activity event log*, used to discover the macro-activity models.

B. Macro-activity recording and preprocessing

In a macro-activity event log, the day is the natural candidate to be used as case. However, some issues arise when identifying a day, namely the timestamp when a day starts/ends. Using a fixed time to identify a day could lead to split a macro-activity (e.g. sleep) into two days. For this reason, we have chosen to set the end of a day dynamically, when the user performs a given activity which usually indicates the end of the day, like going to sleep.

While a micro-activity can be considered atomic and momentary, a macro-activity has a duration. Therefore, for two or

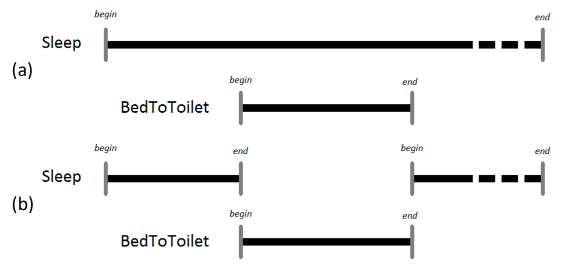

Fig. 3. An example of wrong parallel activities (a) that have been fixed by adding artificial events (b).

more macro-activities we need to consider temporal relations among their executions. In particular, we discuss here the cases of macro-activities executed sequentially (e.g. going to toilette and then going to bed) or in parallel (e.g., having dinner while watching TV). Given two macro-activities A and B, the kind of their mutual parallelism is determined by their relative start/end times. Indeed, we define B to be in *partial overlap* with A if it starts before the beginning of A and finishes before the end of A (or starts after the beginning of A and ends after its end), while it is in *total overlap* if it starts after the beginning of A and ends before the end of A.

Overlapping macro-activities deserve special attention. We identify two main cases in which two macro-activities may overlap:

- *parallel activities*: the two macro-activities have been performed in parallel by the user. For instance, the user watches the TV while cooking;
- *annotation errors*: A and B cannot be executed in parallel by their nature; then the user made an annotation error. For instance, the user cannot perform any other macro-activity while she is sleeping, so the sleep macro-activity must for its nature be performed in strict sequence to any other macro-activities (see Figure 3(a)).

In order to remove annotation errors, we add artificial events in the log. The artificial end of A is added just before the reported begin of B and the artificial begin of A is added just after the reported end of B. Figure 3(b) shows how the wrong parallel behavior of Figure 3(a) has been transformed in a correct sequential behavior.

C. Dataset split and macro-activity model discovery

This section describes the last steps needed to generate models for macro-activities starting from the micro-activity event log and the macro-activity event log that have been pre-processed in the previous phase.

Firstly, the micro-activity event log is enriched with the case id, namely it is labeled with the specific macro-activity that was performed in the same time frame. This step is done by comparing the timestamp of each event with the start/end time of each macro-activity, until a match is found. Then, a *dataset split* step allows to segment the event log by grouping events that are related to different macro-activities, thus obtaining a set of enriched micro-activity event logs. Finally, Process Discovery techniques can be applied to each micro-activity sub-event log to infer a model for each macro-activity.

The choice of the right technique depends on the problem, and hence the kind of event log, at hand. When the problem is structured, namely few variations in the process execution are possible (e.g., sensors in an assembly line), the Alpha Miner algorithm can be successfully used. In highly variable processes, as the ones taken into account in this work, Fuzzy, Heuristic and Infrequent Inductive Miner are more suitable. These algorithms return a process model representing relevant executions, removing infrequent behaviors. The decision on what and how much to remove depends on user-defined parameters, which affect the quality of the discovered process model. More specifically, by filtering too much one can get a simple but inaccurate model, namely a model with a low fitness value. In this case the model is able to represent only a limited part of the traces in the log. On the other hand, by filtering little we obtain a model with a high fitness but so complex (i.e., a flower model) to be practically useless.

V. Case Study

This section is devoted to presenting and discussing the application of the methodology in a real-world case. In Subsection V-A we introduce the dataset and its main features, while in Subsection V-B we apply the methodology and present some quantitative analysis.

A. Dataset

We refer here to the *Milan* dataset which includes a part of the data collected by the CASAS project of the Washington State University[2]. The selected dataset describes the behavior of a user that had been living with her pet for 58 days in a house, suitably equipped with sensors. The analysis of such a dataset is challenging due to the movements of the pet and the sporadic presence of some guests in the house during data collection, which both add noise in the form of sensor activations not related to user's real behavior. The dataset contains data about activities performed by the user in the form of a sequence of sensor activations. An excerpt of the dataset is reported in Table II. Here, each event represents a measurement by a sensor and is described by the record (timestamp, sensor, value, macro-activity):

- *timestamp*, which identifies when an event is recorded. It is the date and time when the sensor has been triggered due to a user's action;
- *sensor*, represented through the unique identifier of the sensor, as mentioned above;
- *value*, which is the value measured by the sensor. For example, it can be an ON/OFF state, a temperature value or an OPEN/CLOSE state of a door;
- *macro-activity*, which is the name of the macro-activity.

As shown in Figure 4, sensors were deployed in various rooms in the house, and belong to different typologies:

- *Motion sensors*, based on infrared technology, track the user moving in the various rooms of the house. These sensors are typically placed one per room, in order to detect the user movements during her daily life activities.

[2]http://casas.wsu.edu/datasets/

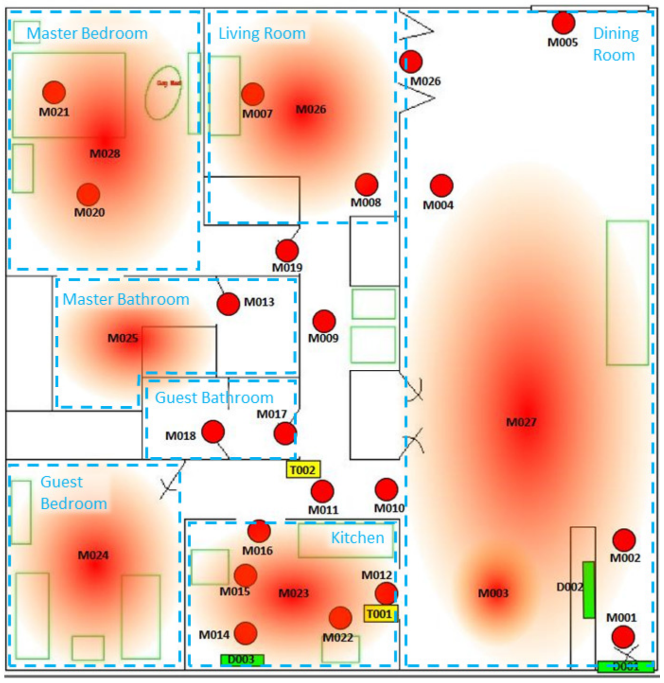

Fig. 4. The map of the house for the Milan dataset with deployed sensors.

In Figure 4, these sensors are denoted with the prefix M followed by an identifying number (e.g., M002). The motion sensors used in these experiments were of two different typologies: spot sensors and wide range sensors. The former are able to detect motion within a small area and are denoted in the map with a small red circle (e.g. M004 or M005). The latter are able to detect motion within bigger areas, that are highlighted in the Figure as red-shaded zones (e.g. M024 or M027). Motion sensors check the presence of movements at a predefined frequency: if no movement is recognized within the detection range, no event is recorded. It triggers to ON if a movement is detected, and again to OFF when no further movement is detected. This behavior can produce noise when either the user stops within the detection area or she moves very fast.

- *Open/close sensors* that are able to detect the status of a door. They are installed on the external doors of the house, allowing to know when someone is entering or leaving. This kind of sensors is denoted with the letter D (e.g., D003).
- *Environmental sensors*, such as temperature sensors, which are represented with the prefix T (e.g., T002).

During the project, the user annotated the macro-activity reporting the timestamps when it begins and ends.

The full list of macro-activities includes: `Bed To Toilet`, `Chores`, `Desk Activity`, `Sleep`, `Read`, `Dining Room Activity`, `Morning Meds`, `Evening Meds`, `Meditate`, `Guest Bathroom`, `Kitchen Activity`, `Leave Home`, `Master Bathroom`, `Watch TV` and `Master Bedroom Activity`. The average number of macro-activities performed by the user per day is around 41 with a standard deviation of 13.7.

TABLE II
AN EXCERPT OF THE INPUT DATASET.

timestamp	sensor	value	macro-activity
...	...	...	...
2009-10-16 08:45:38.000076	M016	ON	Kitchen_Activity begin
2009-10-16 08:45:39.000094	M003	ON	NULL
2009-10-16 08:45:41.000000	M011	OFF	NULL
...	...	...	...
2009-10-16 08:58:52.000004	M016	ON	Kitchen_Activity end
2009-10-16 08:58:52.000053	M023	OFF	NULL
...	...	...	...
2009-10-16 08:59:02.000054	M019	ON	Chores begin
2009-10-16 08:59:04.000072	M019	OFF	NULL
...	...	...	...
2009-10-16 09:14:47.000090	M006	ON	Chores end
...	...	...	...

B. Application of the methodology

1) Preprocessing: The integration between the user's activity journal (i.e., macro-activity event log) and sensors records (i.e., micro-activity event log) allowed to identify the events corresponding to the start/end of an activity. The enriched dataset is reported in Table II (the NULL value in the last column means that no annotation has been done at the given timestamp). The first preprocessing step is aimed at identifying overlapping macro-activities. From a preliminary analysis of the dataset, we observed that in most cases macro-activities are recorded in strict sequences. Overlaps occur only in few cases and annotation errors have been removed as discussed in Subsection IV-B.

Then the dataset has been split with the purpose of producing a micro-activity event log for each macro-activity. The structure of each micro-activity event log is similar to the one reported in Table II, where the *macro-activity* field has been deleted and an identifier representing the different instances (i.e. case ids) of the macro-activity has been added. It should be noted that the values returned by temperature sensors are poorly correlated with the user's activities. Hence, with the aim to minimize noise, temperature sensors have been removed from the log. In addition, the door sensors are placed on the external doors of the house; therefore they are activated very few times for each instance of a macro-activity, and have a much lower percentage of activation than motion sensors. Although low-frequency activation sequences can be discarded by the Process Discovery algorithm (see Subsection III-B), in order to reduce the complexity of the problem at hand, door sensors have also been excluded. As a consequence, we only considered motion sensors that can record an ON or OFF event. The former means that a movement is detected within the sensor's range, while the latter means that after an ON event no further movement has been detected. This behavior can produce noise when either the user stops within the detection area or she moves very fast. In the first case, the sensor stores an OFF value in the event log, but the user is still in the detection area. As the user starts moving again, the sensor will detect another ON event followed by an OFF event, and so on until the user leaves the area. This produces in the event log a sequence of ON and OFF values of the same sensor. In the second case, if the sampling rate is set quite slow, the user could leave the detection area before the

TABLE III
NUMBER OF TRACES AND AVERAGE NUMBER OF EVENTS PER TRACE IN MICRO-ACTIVITY EVENT LOGS USED IN THE EXPERIMENTS.

activity	**number of traces**	**avg events per trace**
Watch TV	114	207.79
Kitchen Activity	554	232.75
Read	314	160.13
Desk Activity	54	141.26
Meditate	17	77.35
Master Bathroom	306	49.25
Master Bedroom Activity	117	233.65
Guest Bathroom	330	32.12

OFF event is stored. Hence, in the event log there could be the activation of a new sensor before the OFF of the previous one. In order to reduce noise, we removed the OFF events from each micro-activity event log. Furthermore, information conveyed by OFF events is related to the time when the user ends a micro-activity, hence it is not relevant for the goal of detecting the process model of macro-activities performed by the user.

In order to have unique start and end points for any instance of the process, for each trace of all event logs, artificial events corresponding to “START” and “END” activities have been added as first and last event respectively. This widespread practice is aimed at improving the soundness of Petri nets resulting from a Process Discovery algorithm [22]. This operation is particularly important when analysing micro-activity event logs. Indeed, it is noteworthy that different traces in the same micro-activity event log could have different activities in the first and last position. This is due to: (a) different ways (i.e., process variants) a macro-activity is executed (e.g., `Kitchen Activity` could start triggering the sensor closed to one of the two kitchen doors, i.e., M012 and M016), and (2) the time shift between the sensor activation and the manual annotation of macro-activities.

2) Macro-Activity Model Discovery: In this subsection we present the results of experiments aimed at extracting macro-activity models. Without loss of generality, we focus only on some of the most relevant macro-activities. For each of these macro-activities, Table III shows some statistics of the related micro-activity event logs, namely the number of traces per log and the average number of events per trace.

From the Table, it turns out that each trace is characterized by several events, namely triggered sensors. It is worth noting that infrequent events slow down the execution of the Process Discovery algorithm and do not contribute to the definition of the model. This is the case of sensors triggered by the guest and the pet. Indeed, the guest performs sporadic, and hence infrequent, activities. As to the pet, we can assume its behavior is independent of user’s behavior; for example, when the user cooks the pet could be in any other room randomly triggering sensors other than those activated by the user. So the sensors activated by the pet are outliers in each trace of considered micro-activity event logs. Hence, in order to further reduce the complexity of the problem, for each macro-activity we only consider the most frequently triggered motion sensors. In particular, for each micro-activity event log, we keep only those sensors whose occurrence frequency

TABLE IV
SENSORS AND RELATED OCCURRENCE FREQUENCIES.

Watch TV		**Kitchen Activity**		**Read**		**Desk Activity**	
Sensor	Freq.	Sensor	Freq.	Sensor	Freq.	Sensor	Freq.
M008	48.08%	M023	26.86%	M004	65.67%	M007	63.38%
M026	8.16%	M014	17.26%	M027	5.61%	M026	15.78%
M003	2.98%	M022	14.24%	M003	3.79%	M008	2.83%
M007	2.96%	M015	11.05%	M001	2.28%		
M001	2.87%	M012	7.24%	M012	2.25%		
M027	2.53%	M027	3.84%	M022	2.15%		
M006	2.10%	M003	3.67%	M005	2.05%		
M017	2.03%						
Master Bedroom Act.		**Meditate**		**Master Bathroom**		**Guest Bathroom**	
Sensor	Freq.	Sensor	Freq.	Sensor	Freq.	Sensor	Freq.
M028	25.35%	M024	62.66%	M025	51.46%	M018	38.76%
M025	23.83%	M011	7.00%	M013	23.04%	M017	18.37%
M020	12.06%	M028	3.65%	M028	8.85%	M003	6.17%
M021	7.08%	M016	3.19%	M020	3.86%	M010	6.08%
M013	4.53%	M025	2.89%			M011	5.74%
M019	4.04%	M020	2.13%			M009	3.43%
M009	2.81%	M022	2.05%			M016	2.45%
						M022	2.42%
						M012	2.24%
						M015	2.16%

TABLE V
FITNESS VALUES OBTAINED BY CHECKING THE CONFORMANCE OF MACRO-ACTIVITY MODELS OVER RELATED REDUCED LOGS.

activity	fitness (%)
Watch TV	99.69
Kitchen Activity	97.96
Read	94.15
Desk Activity	94.91
Meditate	99.69
Master Bathroom	99.49
Master Bedroom Activity	98.56
Guest Bathroom	93.82

is greater than or equal to 2% of the corresponding event log. Table IV shows these sensors and related frequencies. After this filtering operations, we obtain a set of *reduced logs* for the considered macro-activities. These logs have been used to extract macro-activity models adopting the infrequent Inductive Miner algorithm. In order to choose the threshold parameter, we performed several experiments with threshold ranging from 0 to 1 (step 0.1). Hereafter we set the threshold to 0.20, which on average returned the best balance between *fitness and simplicity*. Discovered models show a high fitness with respect to related reduced logs, as reported in Table V. This means that each model is able to correctly replay its reduced log. In order to evaluate the effectiveness of models in describing the related macro-activity, we have also computed the fitness of each macro-activity model with respect to any micro-activity event logs. Results are shown in Table VI.

The discovered model for a certain macro-activity a is an effective representation if it is able to replay the traces in the corresponding micro-activity event log and is not able to replay traces of other event logs. This means that, for the i-th row of Table VI corresponding to a certain macro-activity a_i, the maximum fitness value should be at the i-th column of the event log (in bold in the Table), while values for other columns should be lower. This behavior

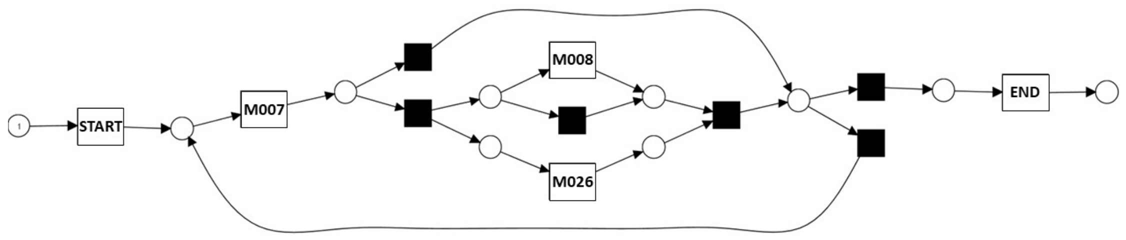

Fig. 5. Macro-activity model for Desk Activity.

TABLE VI
FITNESS VALUES OBTAINED BY REPLAYING ALL MICRO-ACTIVITY EVENT LOGS OVER ALL MACRO-ACTIVITY MODELS.

Reduced Models	**Micro-Activity Event logs**							
	Watch TV	**Kitchen Activity**	**Read**	**Desk Activity**	**Meditate**	**Master Bathroom**	**Master Bedroom Act.**	**Guest Bathroom**
Watch TV	**77.30**	14.73	17.15	89.61	11.60	15.17	13.24	34.77
Kitchen Activity	14.29	**88.42**	19.37	8.20	17.09	13.72	10.01	24.96
Read	7.30	19.57	**80.55**	7.98	10.71	14.65	7.63	18.67
Desk Activity	13.49	6.58	6.02	**82.70**	8.35	14.31	6.74	15.85
Meditate	8.97	23.14	7.03	8.27	**83.8**	53.54	56.29	23.33
Master Bathroom	5.77	6.81	5.66	8.00	11.30	**90.72**	67.40	15.52
Master Bedroom Act.	8.77	7.37	5.88	9.48	12.64	86.85	**82.32**	16.89
Guest Bathroom	11.79	20.00	7.75	7.37	14.15	15.56	11.46	**81.15**

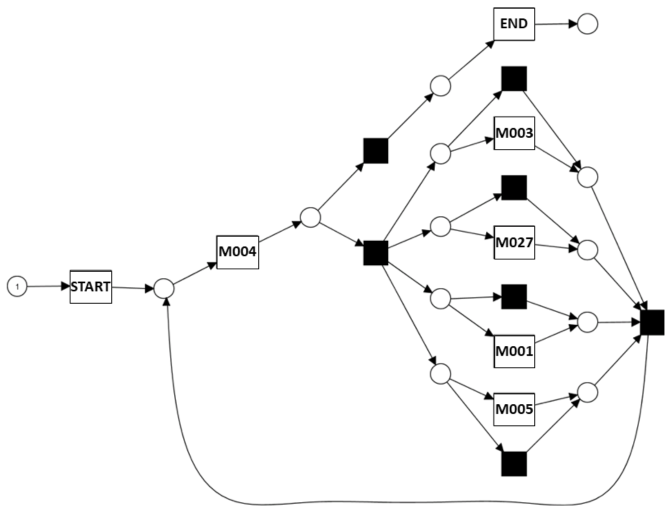

Fig. 6. Macro-activity model for Read.

is experimentally confirmed for the following macro-activity models: Kitchen Activity, Read, Desk Activity, Meditate, Master Bathroom and Guest Bathroom. Models obtained for Watch TV and Master Bedroom Activity return high fitness values both for the related activity and for another one. In particular, the Watch TV model seems to be able to better represent the event log of Desk Activity than the one of Watch TV. The main reason for this inaccurate result is that the two activities take place in the same room and in both cases the user often stops. As a matter of fact, the two event logs share the most frequent sensors, i.e., M007, M008 and M026. Furthermore, the reduced log for Watch TV includes all sensors in the reduced log of Desk Activity. Hence, the model for Watch TV is able to replay traces of Desk Activity, but the viceversa does not hold. We observe the same behavior for Master Bedroom Activity and Master Bathroom. In this case, the reduced log of Master Bedroom Activity contains sensors of the one of Master Bathroom. When we check the conformance of the micro-activity event logs of Master Bathroom and Master Bedroom Activity over the Master Bedroom Activity model, the resulting fitness values are 86.85% and 82.32% respectively. Replaying the same logs over Master Bathroom model, fitness values become 90.72% and 67.40%, as expected. In order to improve the results one could reduce the granularity (e.g., by merging Desk Activity and Watch TV in a sort of living room activity) or the filtering frequency for those micro-activity event logs with few sensors in the reduced logs (e.g., Desk Activity and Master Bathroom). For the sake of space, in Figures 5 and 6 we show two of the eight discovered models[3], namely Desk Activity and Read respectively.

The two models highlight how the use of the reduced log allows to simplify the discovered model and to make them more understandable to human analysts. The model for the Desk macro-activity (Figure 5) is formed only by living room sensors (see Figure 4). In particular, the process is characterized by repeated activations of the sensor closer to the desk (i.e., M007), interspersed with sporadic activations of M008 and M0026. This behavior is compatible with a user who is sitting at her desk for a long time, but she is not perfectly stationary and several subsequent activations of the sensor are stored in the event log (see Section V-B1). Furthermore, the user occasionally leaves the room and so both M008 and M026 are triggered (the parallel path in the middle of the model).

The higher simplicity of the discovered model when using the reduced log can also be appreciated in the macro-activity model of Read (Figure 6). In the main path (the upper in the model) only the M004 sensor is activated, showing that, while reading, most of the time the user stands still without moving. Sometimes she moves while reading to reach the window (near

[3]All models are available at https://github.com/KDMG/process-mining/tree/master/behavior-models

M005) and the main door (near M001), so other sensors in the room are triggered.

VI. Conclusions

This work discusses a methodology aimed at discovering user behavioral models from event logs generated by sensors activation in a smart environment. After a preliminary preprocessing step needed to address issues related to errors and environmental noise, Process Discovery techniques are exploited for model generation, in particular the infrequent Inductive Miner algorithm [22]. While traditional activity models characterize the way a user executes a sequence of daily tasks (e.g., reading, watching TV, sleeping etc. etc.), the specific focus on micro-activity models is novel in the Literature, and allows to retrieve the flow of sensors activations when a given macro-activity is performed by the user.

The effectiveness of the approach has been experimentally evaluated by considering a real-world case study from the CASAS project of the Washington State University. Further experimentations are in line, both taking into account other datasets from the same project and from HicMO [25] a project funded by Marche region which involved several enterprises and universities to establish a laboratory for active ageing. Several extensions of the methodology are currently being developed. Firstly, we are studying techniques to detect the most frequent behaviors and we are also planning to exploit sub-graph mining techniques [26], [27]. Moreover, the application of this technique to environments with multiple users will require particular care in defining the criteria to distinguish activities performed by each of them separately or collaborative activities performed jointly.

References

[1] A. Rashidi, Parisa; Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] F. Leotta, M. Mecella, and J. Mendling, "Applying process mining to smart spaces: Perspectives and research challenges," in *Advanced Information Systems Engineering Workshops*, A. Persson and J. Stirna, Eds. Cham: Springer International Publishing, 2015, pp. 298–304.

[4] M. Cameranesi, C. Diamantini, and D. Potena, "Discovering process models of activities of daily living from sensors," in *Proc. of the Int. Workshop on BP-Meet-IoT, 15th International Conference on Business Process Management*, 2017.

[5] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *International Conference on Pervasive Computing*, ser. LNCS. Springer Berlin Heidelberg, 2004, vol. 3001, pp. 158–175.

[6] P. Rashidi and D. J. Cook, "Mining and monitoring patterns of daily routines for assisted living in real world settings," in *Proceedings of the 1st ACM International Health Informatics Symposium*. New York, NY, USA: ACM, 2010, pp. 336–345.

[7] P. N. Dawadi, D. J. Cook, M. Schmitter-Edgecombe, and C. Parsey, "Automated assessment of cognitive health using smart home technologies." *Technology and health care : official journal of the European Society for Engineering and Medicine*, vol. 21, no. 4, pp. 323–43, 2013.

[8] S. Nasreen, M. A. Azam, U. Naeem, M. A. Ghazanfar, and A. Khalid, "Recognition framework for inferring activities of daily living based on pattern mining," *Arabian Journal for Science and Engineering*, vol. 41, no. 8, pp. 3113–3126, Aug 2016.

[9] T. Huynh, M. Fritz, and B. Schiele, "Discovery of activity patterns using topic models," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, ser. UbiComp '08. New York, NY, USA: ACM, 2008, pp. 10–19.

[10] T. Sztyler, J. Völker, J. Carmona, O. Meier, and H. Stuckenschmidt, "Discovery of personal processes from labeled sensor data - an application of process mining to personalized health care," in *Proceedings of the International Workshop on Algorithms and Theories for the Analysis of Event Data*, 2015, pp. 31–46.

[11] C. Fernández-Llatas, A. Lizondo, E. M. Sanchez, J. Benedí, and V. Traver, "Process mining methodology for health process tracking using real-time indoor location systems," *Sensors*, vol. 15, no. 12, pp. 29 821–29 840, 2015.

[12] A. Senderovich, A. Rogge-Solti, A. Gal, J. Mendling, and A. Mandelbaum, "The road from sensor data to process instances via interaction mining," in *Int Conf on Advanced Information Systems Engineering*, June, 13-17 2016, pp. 257–273.

[13] M. Dimaggio, F. Leotta, M. Mecella, and D. Sora, "Process-based habit mining: Experiments and techniques," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, July 18-21 2016, pp. 145–152.

[14] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, and P. J. Toussaint, "From low-level events to activities - a pattern-based approach," in *Business Process Management*, M. La Rosa, P. Loos, and O. Pastor, Eds. Cham: Springer International Publishing, 2016, pp. 125–141.

[15] C. Liu, S. Wang, S. Gao, F. Zhang, and J. Cheng, "User behavior discovery from low-level software execution log," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 13, no. 11, pp. 1624–1632, 2018.

[16] C. Fernandez-Llatas, B. Valdivieso, V. Traver, and J. M. Benedi, "Using process mining for automatic support of clinical pathways design," in *Data Mining in Clinical Medicine*, C. Fernández-Llatas and J. M. García-Gómez, Eds. Springer New York, 2015, pp. 79–88.

[17] C. Fernandez-Llatas, S. F. Pileggi, V. Traver, and J. M. Benedi, "Timed parallel automaton: A mathematical tool for defining highly expressive formal workflows," in *2011 Fifth Asia Modelling Symposium*, May 2011, pp. 56–61.

[18] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining – adaptive process simplification based on multi-perspective metrics," in *Business Process Management: 5th International Conference*, G. Alonso, P. Dadam, and M. Rosemann, Eds. Springer Berlin Heidelberg, 2007, pp. 328–343.

[19] W. van der Aalst, "The application of petri-nets to workflow management," *Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[20] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[21] A. Weijters, W. van der Aalst, and A. de Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, 2006.

[22] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Business Process Management Workshops*, Beijing, China, Aug, 26 2013, pp. 66–78.

[23] C. Diamantini, L. Genga, D. Potena, and W. M. P. van der Aalst, "Building instance graphs for highly variable processes," *Expert Systems with Applications*, vol. 59, pp. 101–118, 2016.

[24] M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst, "Enabling process mining on sensor data from smart products," in *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, June 2016, pp. 1–12.

[25] L. Rossi, A. Belli, A. D. Santis, C. Diamantini, E. Frontoni, E. Gambi, L. Palma, L. Pernini, P. Pierleoni, D. Potena, L. Raffaeli, S. Spinsante, P. Zingaretti, D. Cacciagrano, F. Corradini, R. Culmone, F. D. Angelis, E. Merelli, and B. Re, "Interoperability issues among smart home technological frameworks," in *Int. Conf. on Mechatronic and Embedded Systems and Applications*, Sept, 27 2014, pp. 1–7.

[26] C. Diamantini, D. Potena, and E. Storti, "Mining usage patterns from a repository of scientific workflows," in *Proceedings of the ACM Symposium on Applied Computing*, 2012, pp. 152–157.

[27] C. Diamantini, L. Genga, D. Potena, and E. Storti, "Discovering behavioural patterns in knowledge-intensive collaborative processes," in *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, vol. 8983, 2015, pp. 149–163.