The Benefits of Sensor-Measurement Aggregation in Discovering IoT Process Models: A Smart-House Case Study

Massimiliano de Leoni and Lucia Pellattiero

University of Padua
Padua, Italy

Abstract. IoT systems collect and exchange data whose analysis opens up incredible opportunities to improve the human satisfaction with IoT systems. The IoT data can be indeed used to discover human habits and interaction patterns, useful to both improve human experience and further automatize the system. Process Mining can be leveraged on for this purpose, but a gap needs to be bridged between IoT-device event data and logs by aggregating events to take to the right granularity for Process Mining. This papers reports on the experience on real-life data to discover the human habits in a smart house. In particular, the benefits are reported on how to aggregate event data to the right granularity to further apply process-mining discovery techniques. The results illustrate that, when applied on the case study, the proposed technique is able to discover human-habit models that are more readable and accurate, thus providing actionable insights for a subsequent optimization of the human experience with the IoT system.

Key words: Sensor Data, Event-log Abstraction, Clustering, Human Habits Model Discovery, Smart House.

1 Introduction

The Internet of Things (IoT) is the inter-networking of physical objects, which can range from sensors and actuators to software systems and devices [1]. The interest for IoT systems is nowadays gaining momentum. In general, IoT systems collect and exchange data via local networks or the Internet. Analysing these data opens up incredible opportunities to improve IoT systems from many viewpoint, e.g. considering the human satisfaction. The data harvested from the sensors can be used to discover the personal habits of the subjects who interact with the IoT system. The discovery can potentially provide valuable input to guide an improvement of the interaction of the IoT-system devices. Furthermore, discovering the main interaction patterns can also be used to reduce the needs of human intervention, thus increasing the level of automation.

Personal habits can be seen as instances of a process, namely a sequence of steps/activities/actions aiming at a certain goal (e.g., cooking food, dressing or washing). With this view in mind, Process Mining can provide the techniques to discover personal habits from sensor data. However, "*a challenge is to bridge the gap between clouds of sensor data and event logs for process mining.*" [1]. As a matter of fact, "*sensor data must be aggregated and interpreted to detect activities that can be used as input for process mining algorithms that support decision making*" [1]. Otherwise, the risk is that the results are given in terms of sensor measurements, which are too low-level to provide interpretable insights. As a matter of fact, existing approaches that apply Process Mining in IoT scenarios overlook the practical need of aggregation (see, e.g., [2, 3, 4, 5, 6]). Typical, they tackle the intrinsic variability by focusing on the main paths, thus potentially filtering out plenty of relevant behavior. In fact, they "sweep the actual problem under the carpet", as well as they generate low-level models that provide little insights for analysts.

This paper reports on an experience to employ process-mining techniques to discover the habit of a human subject who was living in a fully-automated house, which was equipped with sensors to monitor the subject's presence in the different rooms. The sensors produced a stream of events, each of which referred to a measurement: the presence of the subject in a room, or environmental information (e.g., the presence/absence of ventilation in room and its temperature, or power and water usage. Process Mining required a set of event traces, each of which is the recording of one instance of human behavior. The stream of events were transformed in traces, each of which containing the events of a different day. As a result, the mined model encodes the typical subject's behavior in one day.

This case study shares the typical IoT challenge indicated above: the events were sensor measurements instead of actual instances of human actions. This triggered the need to tune the granularity of the events and to abstract these low-level measurements to a higher-level event concepts. To this aim, we employed an *event-log abstraction technique* developed in the realm of process mining [7] and specialized here to the IoT case. In a nutshell, the idea is that the IoT system events are split per day, and each day corresponds to different trace. Events of the same trace can be clustered into sessions such that the time distance between the last event of a session and the first event of the subsequent session is larger than a user-defined threshold. Each trace is thus seen as a sequence of sessions of events. These sessions are encoded into data points, which are later clustered. Each cluster is given a name. The abstract event log is created such that the entire session is replaced by a sequence of two high-level events with the name of the cluster, indicating the moments when the session started and concluded.

The abstraction effort has ultimately made possible to mine a model of the human habit from an event log at the right granularity. The resulting *abstract model* is clearly more informative and accurate than the *flat model* obtained without abstracting. The abstraction has in fact enabled to only focus on the salient aspects.

The abstract model is actually the highest level of a *hierarchical model*. Each transition/activity represents an entire sub-process, whose activities are referred to by the low-level events. Recall that each abstract-model activity is associated to a cluster, which contains a multiset of session traces, i.e., a sub event-log. This enables discovering the sub-process associated to each abstract-model activity by using so-constructed sub event-logs. The hierarchical model has now the same granularity as the flat model. The experiments show the hierarchical model to be more structured and readable, while better balancing fitness (i.e., recall) and precision.

2 The Abstraction Technique

The starting point is an **event log**, which is composed by a set $\mathcal{E}$ of events. Events can be arranged in traces $\sigma = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$ that group events that refer to executions of activities within the same instance of process (e.g., every event that refers to human activities in a domotics house within a certain day). Each event e_i carries information. For the scope of this paper, any e_i is at least associated with the activity $\lambda_A(e_i)$ to

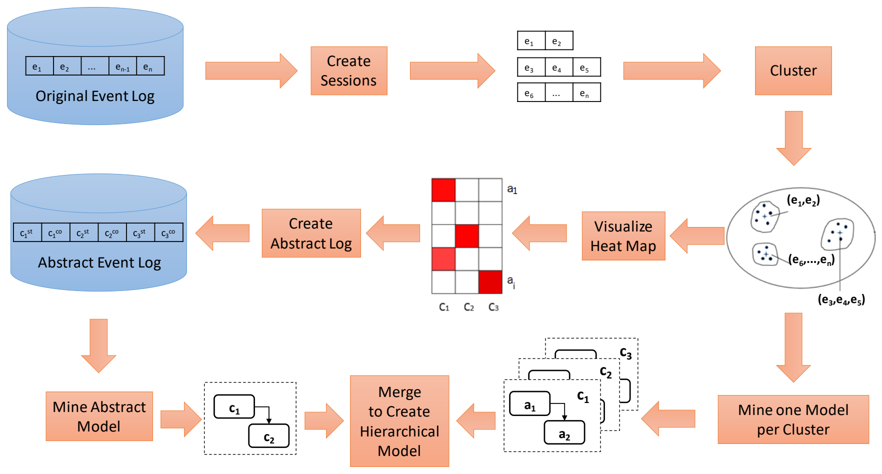

Fig. 1. The Abstraction-Technique Framework for Process Discovery, extended from [7]. For compactness, we assume that each cluster c_i is given name c_i.

which the event refers, and the timestamp $\lambda_T(e_i)$ when the event occurred. Figure 1 summarizes the idea behind the technique: the framework is composed by seven steps, which are discussed below in detail. Note that the first four steps coincide with those in [7], whereas the others are discussed in this paper as new.

Create Sessions. All the traces of the event log are split into sessions; each session of a log trace is a the smallest sub-sequences of the trace such that, for each session, the difference between the timestamp of the last event of the session and that of the first event of the session that follows is larger than a user-defined threshold. For instance, let us consider a trace $\sigma = \langle a_1, b_3, c_4, a_{10}, d_{13} \rangle$. The letter indicates the activity name associated with the event, and the subscript is the timestamp of the event's occurrence (e.g. d occurred at time 13). Assume that the time interval $\Delta = 5$. One can easily see that the time difference between the second occurrence of a and the first of e is greater than Δ: $\lambda_T(a_{10}) - \lambda_T(c_4) = 6 > \Delta = 5$). This yields two sessions: $Ⓢ_\Delta(\sigma) = \langle s_1, s_2 \rangle$ where $s_1 = \langle a_1, b_3, c_4 \rangle$ and $s_2 = \langle a_{10}, d_{13} \rangle$. Note that their concatenation results in σ.

Cluster. Each session s is converted into a vector via an encoding function $\textsc{Encode}(s)$ that abstract the behavior observed in the session. All these vectors (e.g. the sessions) are clustered. The encoding can be obtained in many ways. Here we consider $\textsc{Encode}(s) = (v_{x_1}, \ldots, v_{x_n})$, where $x_1, \ldots, x_n$ are the activities observed in the log and any v_{x_i} is obtained in one of the two following alternatives:

Frequency-based encoding. The value of dimension v_{x_k} is the number of events for activity x_k in s. If both starting and completion events are observed for an activity x_k, the number of events is ultimately divided by two, to prevent double counting.

Duration-based encoding. The value of dimension v_{x_k} is the average duration of instances of activity x_k in the session s (a zero value is given if x_k does not occur in s). To further clarify, let us again consider session s_1 introduced above in paragraph *Create Sessions.* We can estimated that the duration of activity a is 2, because a occurred at time 1 and b at time 3. Similarly, b lasted 1 time unit because b was at time

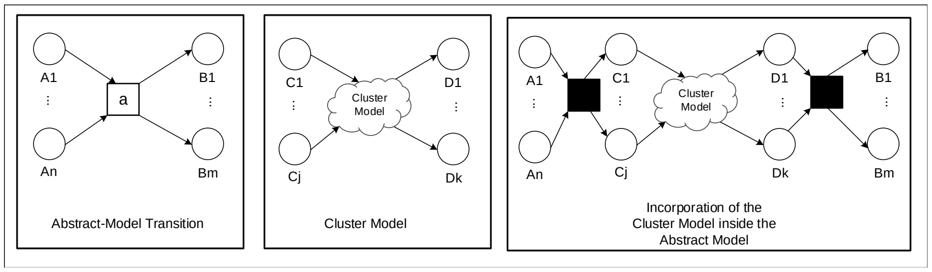
A1
An
a
B1
Bm
Abstract-Model Transition
C1
Cj
Cluster Model
D1
Dk
Cluster Model
A1
An
C1
Cj
Cluster Model
D1
Dk
B1
Bm
Incorporation of the Cluster Model inside the Abstract Model

Fig. 2. Merging one Cluster model into the Abstract Model to create the Hierarchical Model

.

3 and the subsequent event, for c, was at time 4; in the example, the duration of c cannot be estimated because it is the last of the session. In [7], the average duration of c was considered among those executions when the duration was estimated.

Note that the output of the cluster phase can be abstracted as a set $C \in 2^{\mathcal{E}^*}$ of clusters where each cluster is a set of sessions (i.e. sequences). Given a cluster $c \in C$ and an encoding function, the centroid can be easily computed by encoding every session in c and compute the center.

Visualize Heat Map. Clusters are visualized on heatmap to provide analysts with some insights on how sessions were clustered. See example in Figure 1: each column is a cluster (e.g. a human activities) and each row is a low-level activity (e.g. a sensor measured in a domotics house). The cell related to a cluster c a low-level activity a corresponds to the value of the dimension a for cluster c for the centroid of c. Note that, for comparability of the dimension values at centroid, the values are normalized. The normalization of a given centroid $(y_1, \ldots, y_n)$ is achieved by dividing by the sum of the centroid's values: $(\frac{y_1}{sum}, \ldots, \frac{y_n}{sum})$ where $sum = y_1 + \ldots + y_n$.

Create Abstract Event Log. The third step focuses on visualizing the centroids of the clusters on a heatmap, thus providing information about the most predominant activities for low-level events. This information is used by process analysts to assign cluster names $\textsc{Name}(c)$ to each cluster $c \in C$. The fourth step creates the abstract event log. For each log trace σ, we compute the sessions' sequence $Ⓢ_{\Delta}(\sigma) = \langle s_{\sigma_1}, \ldots, s_{\sigma_m} \rangle$ and then the abstract trace $\sigma_{\textsc{Abst}} = \langle f^{st}_{\sigma_1}, f^{co}_{\sigma_1} \ldots, f^{st}_{\sigma_n}, f^{co}_{\sigma_n} \rangle$ where, for every $1 \leq i \leq m$, $f^{st}_{\sigma_i} and f^{co}_{\sigma_i}$ are the abstract events for session s_{σ_i}: the activity names of $\lambda_A(f^{st}_{\sigma_i})$ and $\lambda_A(f^{co}_{\sigma_i})$ are equal to the name $\textsc{Name}(c_i)$ of the cluster c_i to which s_{σ_i}.

Mine Abstract Model and Mine one Model per Cluster. Here, we can use employ any discovery algorithms available in literature [8]. The step to mine one Model per Cluster is only necessary when one wants to create a hierarchical model. In particular, it aims to mine the sub-process related to each activity of the abstract model. Each cluster $c \in C$ consists of a set of sessions, which is in fact an event log to be used as input to mine a model of the behaviour of the sessions within the cluster. Here we also assume to employ a miner that returns strongly connected Petri nets.

Merge to Create Hierarchical Model. Figure 2 illustrates how one cluster model can be incorporate into the abstract model to create the hierchical model: the leftmost part shows the portion related to the high-level activity a where the center part depicts the

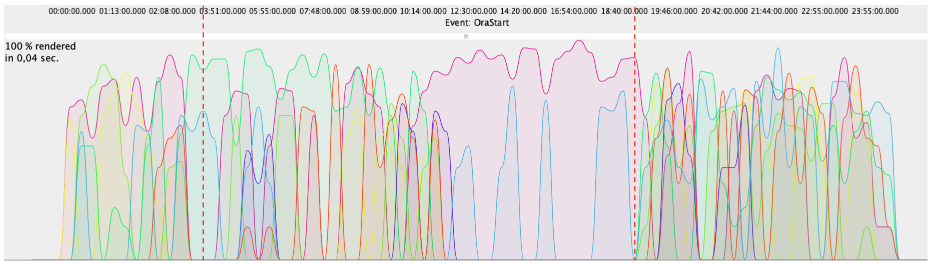

Fig. 3. Curves indicating the distribution of event occurrence over time. Each curve is associated with a different weekday and colored differently. Pink: Sunday, Yellow: Monday, Green: Tuesday, Aqua: Wednesday, Purple: Thursday, Orange: Friday, Light Blue: Saturday.

structure of the model of the cluster referring to high-level activity *a*. The incorporation of the cluster model into the abstract model is illustrated in the rightmost part of the figure, where two invisible transitions are introduced (the black squares) in the abstract model along with the cluster model, which is connected as shown in figure. This procedure is repeated for each cluster, thus finally producing the hierarchical model. Note that often those invisible transitions are not necessary (see, e.g., the case study): in that case, the invisible transitions can finally be removed.

3 A Case Study in Domotics

This case study is based on the smart-home event data that were employed as challenge for the 4th International Workshop on BP-Meet-IoT in 2020.[1] The complete dataset includes three different scenarios: morning routine of one person, daily routine of one person and daily routine of two people. For our experiments we used the second one which includes recordings for all the activities of one user within 21 days. All the datasets provide an event stream of all the activities performed by the user inside the smart home.

3.1 Preprocessing of the Event Dataset

In order to apply our abstraction method we had to perform a preprocessing of the event data. The aims of this phase were two: to remove the activities that represent noise and to split the event stream into traces.

In the original log the noisy activities were conveniently named as "Noise", so they were removed using a filter on the activity name. Secondly, to split the stream into traces we analysed the time distribution of the activities within a day. To do that we extract the hour from the timestamp and we used a dotted chart visualization. We arranged all the activities in an interval from midnight to 11:59pm. Figure 3 shows the time distribution of low-level events over day time. Each curve refers to a different weekday and is associated with a different color. Note that, e.g., between around 12:30pm and 7pm, events only occurred on Saturdays and Sundays, since the house inhabitant was likely at work on the other days. The identification of potential breaking points is based on the analysis of the the areas with the lowest density of activities. We can identify two possible breaking points, highlighted by the red dashed lines in the figure: one around

[1] BP-Meets-Iot Challenge page: `http://pros.webs.upv.es/sites/bp-meet-iot2020/#six`

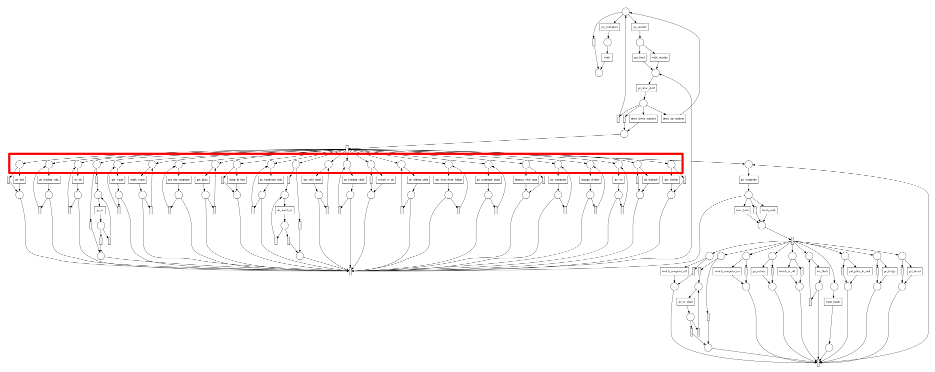

Fig. 4. *Flat model* of human habits, discovered on the smart-home transaction data via Inductive Miner. Due to the structure in the area highlighted in red, the model is very imprecise, allowing for too much behavior.

3am and one around 7pm. Based on this analysis, and on the domain knowledge we had about the user habits, we have chosen to split the traces at 7pm, because this is the hour at which the user ends its working day. The examination of the log and the underlying process shows the alternative at 3am to not represent a meaningful splitting: no recognizable pattern was observed around that time.

The result event log was composed by 22 traces, with overall 2918 events for 66 low-level activities.[2]

3.2 Application of the Event-Log Abstraction Technique and Results

The resulting event log was used to discover a *flat model*. The Inductive Miner [8] was employed, using the default configuration: a noise threshold equal to 0.2. The model is depicted in Figure 4. The model was discovered on 70% of the traces of the post-processed event log, while 30% were used for fitness computation. Conversely, precision was computed using every trace in the post-processed log. Here, we have employed a token-based computation of fitness and precision [8] because the alignment-based computation never completed on the flat model due to the its intrinsic complexity. The fitness value is 0.94, but the precision was extremely poor: 0.034. This clearly indicates that the model allows for an excessive amount of behavior, compared with what recorded in the event log. This is ultimately caused by an excessive level of parallelism: several activities are possible in any order (see in particular the area shown in red). The application of the event-abstraction method requires to provide a session threshold. In this case study, we computed the elapsing time between all couples of consecutive events in the log. Figure 5 shows the box plot of the

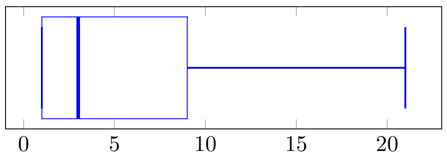

Fig. 5. Box plot of the distribution of elapsed times between consecutive events in minute. Outliers are removed for readability.

[2] The post-processed event log is available at `https://github.com/Ciaaa95/TechniquesForClustering_LowLevelEvents`.

Encoding	Clustering	Fitness	Precision	Harmonic Mean
Frequency	KMeans	0.51	0.69	0.59
Frequency	DBScan	0.56	0.64	0.60
Duration	KMeans	0.20	0.80	0.32
Duration	DBScan	0.15	0.70	0.25

Table 1. Results of the abstraction

elapsed-time distribution, where the upper whisker is equal to 1.5 times the interquartile range, namely 21 minutes. Note that the lower quartile is equal to 1 minute, which is also the minimum value. Around 17% of values are outliers, namely larger than the upper box-plot whisker, with values till 847 minutes. This shows that most of measures are small, but a portion of values are very large. This motivates the reason to use the median as session threshold, which is 3 minutes. This enabled the application of the methods presented in Section 2. We employed *KMeans* and *DBScan* [9] as clustering algorithms, which are the most widespread.

Parameters Tuning for KMeans and DBScan. KMeans requires determining the number of clusters to be obtained. Here, we employed a well-known technique called *Elbow Method*, in which we compute the error curve in a given interval of number of clusters. By displaying this curve in a graphic we can then easily identify the value that offers the best tradeoff between too much generality and excessive specialization. DBScan relies on two parameters: *epsilon*, which determines the cluster dimension of the region to examine at each iteration, and *minPts*, which is the minimum number of samples that identifies an high density region. Parameter *epsilon* was estimated by computing the distance of the nearest neighbour for all the samples, then by ordering the points by distance we chose the value that ensure us to include most of the samples. Then, we computed the number of neighbour inside a region of radius *epsilon* for each sample and we ordered the results by number of neighbours. The resulting distribution presents a initial peak which means that most of the samples have very few neighbours within *epsilon* distance meaning that their surroundings have low density of points. After the peak the number decreases until it reach a local minimum. This minimum identifies a good estimation for *minPts* because it represents the breaking point between low-density and high-density regions.

Determination of the Best Abstract Model. We evaluate the quality of the abstract models by (1) abstracting the log trace using DBScan and KMeans and via both a frequency and time encoding of the sessions, and (2) discovering a model via inductive miner, using the same configuration as for the flat model (cf. the beginning of this Section). To evaluate the models we have calculated fitness and precision by computing alignments, which has become a “de facto” standard [8]. We used 70% of the traces as training set to mine the model and 30% to calculate the fitness, whereas for the precision we use the entire log. Table 1 summarizes the results that we have obtained for fitness and precision, highlighting their harmonic mean. We observe that frequency encoding generates better abstract models in terms of harmonic means of fitness and precision, both via KMeans and DBScan. Therefore, the used clustering algorithm is largely irrelevant. We finally opted for DBScan with the highest mean of 0.6. Figure 6 shows the heatmap obtained for this abstraction, which highlights how each cluster has one or two evident dominant low-level activities (the cells of the darkest colors). Table

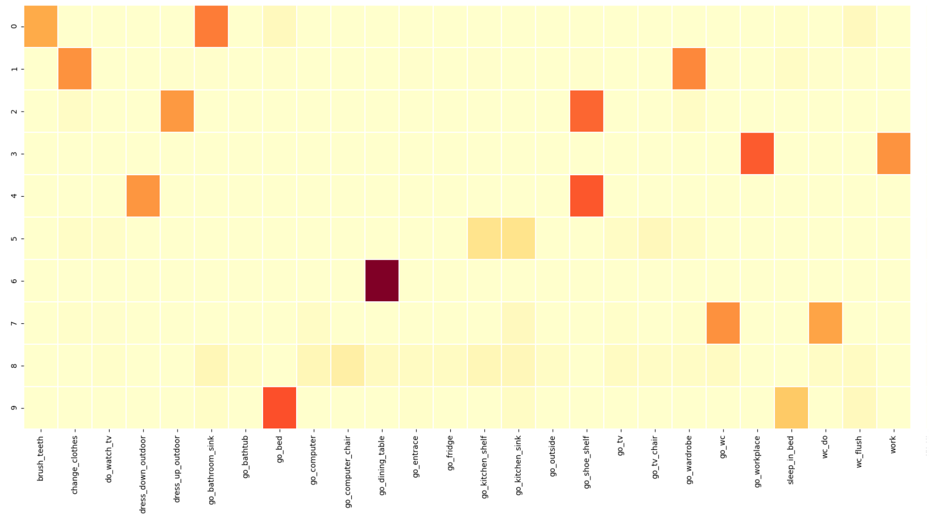

Fig. 6. Heatmap for the abstraction obtained with DBScan clustering and frequency encoding

Cluster	Most predominant activity(ies)	Automatic Cluster Name	Manual Cluster Name	Distance
0	brush_teeth, go_bathroom_sink	Go bathroom sink0	WC	0.29
1	go_wardrobe, change_clothes	Go wardrobe1	Go work, Walk	0.50
2	dress_up_outdoor, go_shoe_shelf	Go shoe shelf2	Go work, Walk	0.50
3	go_workplace, work	Go workplace3	Go work	0.25
4	dress_down_outdoor, go_shoe_shelf	Go shoe shelf4	Go work, Walk	0.50
5	go_kitchen_sink	Go kitchen sink5	Drink	1.00
6	go_dining_table	Go dining table6	Eat cold, Eat warm	0.40
7	go_wc, wc_do	Go WC7	WC	0.57
8	go_computer_chair	Go computer chair8	Use computer	0.33
9	go_bed, sleep_in_bed	Go bed9	Prepared to sleep	0.44

Table 2. Comparison between domain knowledge based abstraction and our abstraction method

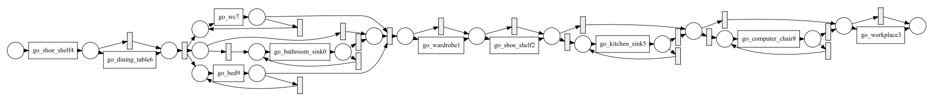

Fig. 7. Abstract model obtained using the frequency encoding and DBScan clustering, using Inductive Miner with noise threshold of 0.2

2 reports on the associations between cluster number, name, and dominant activities in the three leftmost columns: the name of each cluster coincides with that of the most dominant activity in the cluster, followed by the cluster number. Figure 7 shows the abstract model.

Automatic vs Manual Cluster Creation. The dataset used in this case study is accompanied by a document that provides a manual creation of the clusters of low-level events. The manual abstraction was provided by the challenge as a separate log, in which each trace identifies a high-level activity.[3] It is hence worthwhile comparing our clusters - hereafter *automatic* - with those *manual* created via domain knowledge. For

[3] The dataset description is available at `http://pros.webs.upv.es/sites/bp-meet-iot2020/challenge/BP_Meets_Iot2020_Challenge_Dataset.pdf` (Accessed June 2th, 2021)

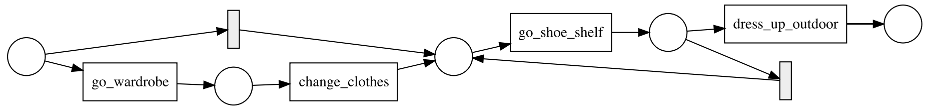

Fig. 8. Model for cluster number 2 named go_shoe_shelf

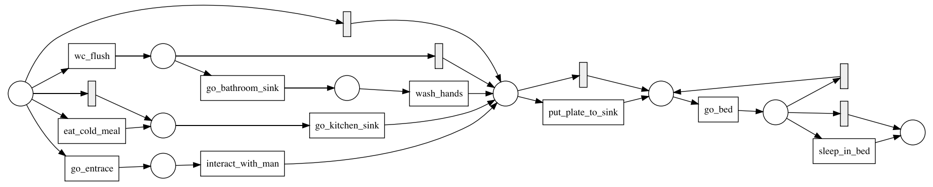

Fig. 9. Model for cluster number 9 named go_bed

each automatic cluster, we compute the closest manual cluster. The distance between an automatic A and a manual cluster M is ${}^{2\cdot|Att_A \cap Att_M|}/_{|Att_A|+|Att_M|}$ where Att_A and Att_M are respectively the set of activities in A and M with a relative frequency above 0.01 in the centroid (cf. definition of frequency in paragraph *Cluster* in Section 2).

Note that the manual clusters were already given a name by the experts, using domain knowledge. We have made the automatic cluster names correspond to the dominant activity name. The results are reported in Table 2: for each automatic cluster, the table reports the automatic cluster name (i.e. via our technique), the manual cluster names (i.e. via the domain knowledge) and the distance. Sometimes, two manual clusters were equally distant from the corresponding automatic (see cluster 1, 2 e 4). A qualitative comparison of the automatic and manual cluster names illustrates that our technique seems to discover high-level activities that reflect similar concepts as the manual clusters. See, e.g., cluster 0 (*Go bathroom sink* vs *WC*), cluster 3 (*Go workplace* vs *Go work*, or cluster 5 (*Go kitchen sink* vs *Drink*). Furthermore, clusters 2 e 4 have the same most prominent activity (i.e., *go_shoe_shelf*), but the second prominent changes: *dress_up_outdoor* and *dress_down_outdoor*, respectively. This means that, very likely, automatic cluster 2 should correspond to manual cluster *Go work*, and automatic cluster 5 should corresponds to *Walk*. Unfortunately, there were manual clusters, such as *Wash dishes* and *Doing exercises*, that were not matched by any automatic cluster. This is due to the fact that those manual clusters refer to low-level events that occur more rarely in the log. As a consequence, our technique has failed to group them in a standalone cluster, clustering them along with other events that were more frequent and thus predominant.

Hierarchical Model The case-study last step was that to mine the inner workflow of the high level activities. Figures 8 and 9 shows two example models, respectively for cluster 2 (go_shoe_shelf) and cluster 9 (go_bed). These models were mined using the Inductive Miner with the same noise threshold as when we mined the abstract model.

In accordance with the technique discussed in Section 2, we replaced each high-level activity of the model in Figure 7 with a whole process model that was mined for the correspond cluster. By doing that we obtain the hierarchical model shown in Figure 10. For a quantitative evaluation, we have compared fitness and precision of this hierarchical model with the flat model in Figure 4. To ensure a failr comparison, fitness and precision is again computed based on token replaying [8] because the alignment-

	Fitness	Precision	Harmonic Mean
Flat Model	0.91	0.034	0.065
Hierarchical Model	0.66	0.35	0.45

Table 3. Comparison between flat model and hierarchical model based.

based computation never completed on the flat model due to the its intrinsic complexity. The precision and fitness values are summarized in Table 3 and compared with those of the flat model: although the fitness of the hierarchical model is lower than that of the flat model, the precision is ten-times higher. This large difference is mainly due to the fact that the hierarchical model is much structured to capture the order relations between the activities. In sum, it follows that the harmonic mean of fitness and precision shows that the hierarchical is significantly better model. More qualitatively, if we compared the lasagna-like structure of the hierarchical model with a more spaghetti-like structure of the flat model, we can certainly conclude that the hierarchical model is more readable and suitable to convey insights to process analysts.

4 Discussion and Conclusion

This paper has reported on the successful experience to employ event-log abstraction techniques to learn the daily human habits. Data were harvested from sensors and devices in a domotics house. Event-log abstraction is likely necessary in many IoT settings that generate low-level logging data. If these data were directly used to discover some model of the IoT system usage or human habits, the model would possibly be of little use because it would talk in terms of technical details (e.g., from sensor), instead of higher-level concepts. The abstraction enables tuning the granularity to a level that is meaningful from a conceptual/business perspective.

Even when one does not need to change the event granularity, the event-log abstraction technique leads to hierarchical models where concepts are represented at different levels. The human-habit case study has illustrated that it is possible to create hierarchical models that *(i)* can better balance fitness and precision and *(ii)* are significantly more insightful than flat models.

We acknowledge the low precision of the abstract and hierarchical model for the case study declinated in this paper. We expect similar values for those of other IoT systems. This is not caused by the abstraction techniques: in fact, the flat model scores almost zero in precision. This is likely due to the use of Petri nets, which is less suitable for processes that show high behavioral variability, in comparison with more structured processes. Future research directions ought to investigate declarative modelling approaches [10] or hybrid. The latter refers to a modelling notation that can alternate parts with a declarative notation with others using a Petri-net-like notation [11].

Every model for the case study was discovered via Inductive Miner [8], with the same configuration of parameters. However, a more thorough assessment requires the employment of different process-discovery techniques and the analysis of the quality of results, when abstraction techniques are used. We do not expect though to see significant differences.

Furthermore, we also tried to employ contextual information to provide additional information to improve the clustering accuracy. The dataset of the human-habit case study indeed contained information related to the room temperature and humidity or

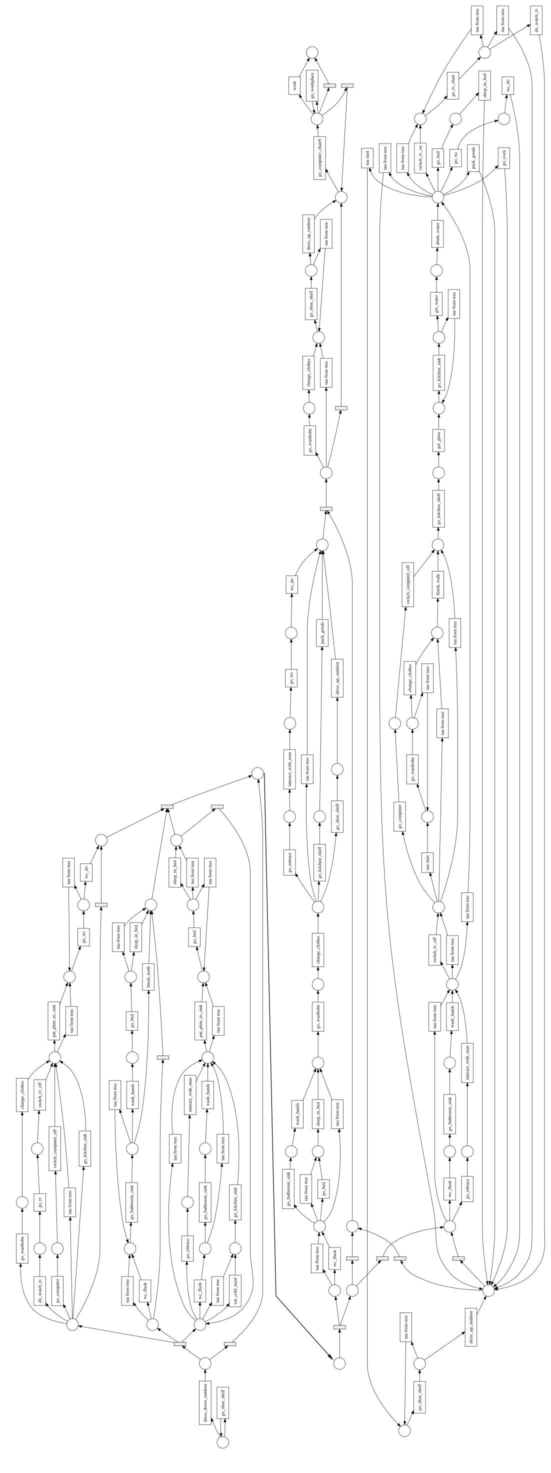

Fig. 10. Hierarchical Model of the daily human habits.

the level of ventilation (e.g. because of open windows). As an example, the human presence in a room might have a different meaning depending on the room temperature and whether the window is open or close. However, this additional information showed to not impact the clustering, which makes us believe that information is not relevant, at least for the case study in question.

The case study certainly hints at the relevance of the abstraction approach employed here. However, for a conclusive, definitive assessment, we plan to conduct further case studies in different IoT system settings.

References

1. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Fortino, G., Gal, A., Kannengiesser, U., Leotta, F., Mannhardt, F., Marrella, A., Mendling, J., Oberweis, A., Reichert, M., Rinderle-Ma, S., Serral, E., Song, W., Su, J., Torres, V., Weidlich, M., Weske, M., Zhang, L.: The internet of things meets business process management: A manifesto. IEEE Systems, Man, and Cybernetics Magazine **6**(4) (2020) 34–44
2. Hemmer, A., Badonnel, R., Chrisment, I.: A process mining approach for supporting iot predictive security. In: Proceedings of OMS IEEE/IFIP Network Operations and Management Symposium (OMS 2020). (2020)
3. Coltellese, S., Maria Maggi, F., Marrella, A., Massarelli, L., Querzoni, L.: Triage of iot attacks through process mining. In: On the Move to Meaningful Internet Systems: OTM 2019 Conferences, Springer International Publishing (2019)
4. Dimaggio, M., Leotta, F., Mecella, M., Sora, D.: Process-based habit mining: Experiments and techniques. In: 2016 IEEE 13th International Conference on Ubiquitous Intelligence & Computing. (2016) 145–152
5. Tax, N., Sidorova, N., Aalst, W.M.P.: Discovering more precise process models from event logs by filtering out chaotic activities. Journal of Intelligent Information Systems **52**(1) (2019) 107–139
6. Cameranesi, M., Diamantini, C., Potena, D.: Discovering process models of activities of daily living from sensors. In: Business Process Management Workshops. Volume 308 of LNBIP., Springer (2018)
7. de Leoni, M., Dündar, S.: Event-log abstraction using batch session identification and clustering. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. SAC '20, New York, NY, USA, Association for Computing Machinery (2020) 36–44
8. van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer (2016)
9. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
10. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: Full support for loosely-structured processes. In: EDOC 2007, IEEE (2007) 287–287
11. van der Aalst, W.M.P., De Masellis, R., Di Francescomarino, C., Ghidini, C.: Learning hybrid process models from events - process discovery without faking confidence. In: 15th International Conference on Business Process Management (BPM 2017). Volume 10445 of LNCS., Springer (2017) 59–76