SAPIENZA
UNIVERSITÀ DI ROMA

Unsupervised Human Process Discovery in Smart Homes

La Sapienza Università di Roma
Dipartimento di Ing. Informatica, Automatica e Gestionale "A. Ruberti"

Dottorato di Ricerca in Ingegneria Informatica – XXXVI Ciclo

Candidate

Silvestro V. Veneruso
ID number 1461229

Thesis Advisor

Prof. Francesco Leotta

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Engineering in Computer Science

28 February 2024

Unsupervised Human Process Discovery in Smart Homes
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: May 17, 2024

Author's email: silvestro.veneruso@uniroma1.it

Contents

Extended Abstract

The advances in the Internet of Things (IoT) have enabled the automation of various tasks like switching on the heating at home from work, seeing who is at your front door from the couch, supporting nurses in elderly homes, or the efficient delivery of packages. By enabling the connection between the physical and digital worlds, the IoT has shown how environments can be augmented with technology to enhance their capabilities, making them more intelligent, responsive, and adaptive. This widespread adoption of *embedded systems* turned *pervasive* (or *ubiquitous*) computing into reality: while sensors gather real-time data about the environment, actuators are used to automate the execution of many tasks that help the users of such environments. These environments, referred to as *smart environments* or *smart spaces*, represent an emerging class of IoT-based applications and are centered on their human users. Among smart spaces, smart homes and offices are representative examples. The goal is to enhance the quality of life, improve productivity, and provide personalized services by understanding and responding to the needs and preferences of the users, realizing the paradigm known as *Ambient Intelligence* (AmI) [185].

The literature presents various definitions of AmI. In [48], authors introduce a set of distinct features that characterize AmI systems: sensitivity, responsiveness, adaptivity, ubiquity, and transparency. *Sensitivity* pertains to the AmI system's ability to perceive and comprehend the surrounding environment and its interaction context. *Responsiveness* and *adaptivity*, closely tied to sensitivity, indicate the system's capacity to promptly react, either proactively or reactively, to changes in the context in accordance with user preferences. Collectively, sensitivity, responsiveness, and adaptivity contribute to the overarching concept of *context awareness*. Lastly, the terms *ubiquity* and *transparency* directly relate to the idea of pervasive computing.

Smart environments process and analyze the data collected from sensors to extract meaningful information. In this context, AmI is realized by utilizing techniques such as machine learning, artificial intelligence, and human-computer interaction (HCI). The rich data automatically collected via IoT sensors in smart spaces is used to get insights about the human behavior of the user (e.g., sleep tracking) or to perform automated actions for the user (e.g., automatically opening the blinds). For instance, current applications of human behavior monitoring in smart spaces include smart thermostats (e.g., Google Nest Learning Thermostat) and ambient assisted living (e.g., elderly fall detection systems).

Modeling human activities and habits is not a simple task, due to the flexible and unstructured nature of human behavior. Recently, although it is still difficult to represent them following a precise flow of tasks, approaches have been proposed that

model human habits as workflows [124]. In particular, the research community and manufacturers have shown a great interest in applying *process mining* (PM) to smart spaces. Process mining [4] is a fairly recent research discipline that combines data mining techniques with techniques used in Business Process Management (BPM) [67], such as process modeling and process analysis. Process mining aims to extract, monitor, and improve processes based on real-world data.

In particular, *process discovery* is a process mining technique used to discover and generate the process model describing the underlying behavior shown in the event log. The mined process model can be visualized in different forms, such as Petri nets, process flowcharts, or BPMN diagrams. Visualization helps to understand the structure and dynamics of processes within the smart space. However, even though process models could be extracted from smart space data, multiple important challenges arose [124].

The next section presents an overview of how some of the aforementioned research challenges are handled and to what degree they are addressed by the author of this thesis.

Research Contributions

Scientific contributions need to be validated against datasets, such as sensor logs from smart environments. In order to acquire these datasets, expensive facilities are needed, including sensors, actuators, and an acquisition infrastructure. In addition, frequently employed smart home hubs (e.g., voice assistants like Amazon Echo Dot) do not allow access to raw data.

Even though several freely accessible datasets are available, each of them features a very specific set of sensors, which can limit the introduction of novel approaches that could benefit particular types of sensors and deployment layouts. Additionally, acquiring a dataset requires a considerable human effort for labeling purposes, thus further limiting the creation of new and general ones. Also, labeling is an error-prone activity, which makes the quality of the available datasets unclear. As a consequence, the vast majority of approaches available in the literature are evaluated against datasets gathered in university labs, i.e., datasets that are not general and whose quality cannot be taken for granted.

For this reason, smart environments are one of the many disciplines where we are witnessing the *replication crisis* (or replicability crisis or *reproducibility crisis*), i.e., an ongoing methodological crisis in which it has been found that many scientific studies are difficult or impossible to replicate or reproduce [11][74]. We define the following objective:

O1 Analysis of available datasets in literature and subsequent development of a dataset generation tool able to generate synthetic datasets that take into account features that we can find in a real-world setting, not only in an experimental setting.

With reference to objective **O1**, the following research contributions have been achieved:

R1-1 In [24] we provide several insights on the datasets used by the research community across the primary studies surveyed in this work (e.g., available sensors, number of users involved, number of measurements, labeled activities, and others).

R1-2 In [195] we propose a model-based simulator capable to generate synthetic datasets that emulate the characteristics of the vast majority of real datasets while granting trustworthy evaluation results. The datasets are generated using the eXtensible Event Stream (XES) international standard commonly used for representing event logs. This format is then further extended by the work presented in **R2**. The simulator and related resources can be downloaded at the link in the footnote[a].

[a] `https://github.com/silvestroveneruso/smart_space_model_based_simulation`

As the utilization of IoT devices in support of business processes (BPs) becomes more frequent, there is a growing recognition of the potential to leverage the data collected by these devices for process mining (PM). Most current PM methods that can incorporate IoT data follow a similar approach: the IoT data are pre-processed with event abstraction and event-case correlation techniques to be translated into an event log in XES format [26, 97, 164, 171].

Although this is an interesting initial approach to integrating IoT data into PM and it allows for the application of existing control flow and data-aware techniques, this method does not fully exploit the potential of IoT data. Often, the resulting high-level event log lacks contextual information (i.e., properties that can influence process execution [26, 166]) that could be derived from the IoT data, or it has limited capability to incorporate such context information. Furthermore, by separating the abstraction phase from the analysis phase, the true potential of advanced algorithms to optimize both abstraction and model discovery together cannot be harnessed. For example, the development of an IoT-enhanced decision mining algorithm requires direct access to lower-level IoT data to learn the most relevant features directly from the source data instead of relying on an error-prone event abstraction step that might leave important information behind at a lower granularity level [27].

This shortcoming of existing approaches is to a large extent due to limitations of the most common event log standards, i.e., the eXtensible Event Stream [86] (XES) and the Object-Centric Event Log [81] (OCEL), and has been acknowledged in the IoT PM literature [95] and beyond [20]. In [27], the authors listed ten requirements for the storage of IoT-enhanced event logs and showed that both XES and OCEL failed to meet more than half of these requirements. We define the following objective:

O2 Definition of a suitable event log format integrating IoT data to process data. In particular, the format has to follow the list of requirements for the storage of IoT-enhanced event logs outlined in [27].

With reference to objective **O2**, the following research contribution has been achieved:

R2 In [28] we present the Native Iot-Centric Event (NICE) log, a new event log format designed to incorporate IoT data into a process event log, ensuring traceability, flexibility, and limiting data loss. We evaluated our format against requirements previously established for an IoT-enhanced event log format, showing that it meets all requirements, contrary to other alternative formats. We then performed an analysis of a synthetic log to show how IoT data can easily be used to explain anomalies in the process. Furthermore, the new format was linked to the smart space data simulator presented in **R1-2** [195].

The rationale behind applying process mining techniques in a smart space is to exploit the vast set of data mining methodologies targeting classical business processes to so-called cyber-physical processes [115]. In order to apply techniques from the PM area, a sensor log must be converted into an event log. However, as pointed out in [124, 192], turning sensor measurements into events is a complex challenge that can hardly be solved without human manual labeling, additional knowledge, or probabilistic reasoning.

From a more practical point of view, many of the challenges are related to the difference between *sensor logs* produced by smart spaces and *event logs* produced by information systems, which are usually fed as input to PM algorithms. Whereas events in event logs record the execution of tasks, e.g., their start and their completion, sensor logs contain fine-grained sensor measurements, e.g., the temperature in a room at a certain point in time or the presence of a user near a piece of furniture.

The converted event log must contain at least three elements [158]: *(i)* the case id, which identifies a specific process instance, *(ii)* the label of the task performed, and *(iii)* the timestamp. Since (i) and (ii) are absent from the sensor log, a pre-processing step is required to infer events from the sensor log. This task usually consists of two steps, respectively: *(1)* bridging the gap between sensor measurements and events in order to derive the task label, and *(2)* segmenting the event log into traces in order to assign a case ID to each event. For this purpose, we define two objectives:

O3 Investigation on techniques available in literature to convert sensor logs into event logs.

With reference to objective **O3**, the following research contribution has been achieved:

R3 In the work already presented in **R1-1** [24, 129] we provide an overview of the techniques used by the research community to convert sensor events into process events and how they deal with the challenges related to the conversion task, i.e., bridging the abstraction gap between sensor and event logs and segmenting logs in traces. The main contribution of this article is two-fold: *(i)* providing the research community with an analysis of the existing applications of process discovery to smart spaces and how they address the common challenges in the field, and *(ii)* assisting further research efforts by outlining opportunities for future work.

When discussing smart spaces, the following terminology is usually employed [126]:

- *Action*, i.e., atomic interaction with the environment or a part of it (e.g., turning on the TV).

- *Activity*, i.e., a group of human atomic interactions with the environment (actions) that are performed with a final goal (e.g., cleaning the house).
- *Habit*, i.e., a group of actions or activities (one in the extreme case) that define what happens in specific contextual conditions (e.g., what the user usually does in the morning between 08:00 and 10:00).

Being able to recognize the onset and end of an activity and/or a habit performed by a human can be helpful for different purposes. Examples of applications include the definition of automation rules or the detection of potentially harmful deviations from usual behavior.

Unfortunately, the practical applicability of techniques proposed in the literature is limited by the effort required by the final user to manually label smart home logs to train recognition models. The vast majority of approaches are indeed based on supervised learning, thus requiring the logs to be labeled with markers denoting the onset and end of all (or at least of a consistent subset) of the activity (or habit) occurrences.

Manual labeling of logs is perceived by the final users as annoying, which ends up in imprecise labeling. Despite the fact that automatic techniques to label a smart home log have been proposed, they suffer from several limitations when applied to real-world scenarios [90].

First, many approaches (e.g., [112]) require manually specifying window lengths or other kinds of numerical thresholds (e.g., number of events, minimum distance between two events). The selection of such parameters is hard and does not take into account the peculiarities of the different activity types. In second place, the vast majority of proposed solutions are directly applied to raw sensor measurements. This strategy does not exploit the meaning of a sequence of sensor measurements, only highlighting the statistical distribution of occurrences and co-occurrences of sensor events. Finally, in many cases, automatic segmentation techniques are only used to complement manual segmentation and are not intended to segment the full log (e.g., [46]). We define the following objective:

O4 Definition of an *unsupervised* segmentation methodology to automatically and fully segment a smart home log by applying techniques such as machine learning, artificial intelligence, and process mining. The methodology has to avoid the annoying labeling task effort from final users.

With reference to objective **O4**, the following research contributions have been achieved:

R4-1 In [73, 71] we propose a methodology allowing, given a sensor log, to automatically segment *human habits* by applying a classical bottom-up discretization strategy on the timestamp attribute. Such a class of discretization algorithms finds the best division of a continuous attribute by iteratively merging contiguous sub-ranges (also called "bins") following a quality evaluation heuristic. In our proposal, the heuristic is based on quality measures computed on the process models automatically mined, through process discovery, from the intermediate bins. In particular, we drive the discretization targeting process models with high *simplicity* and low *structuredness*. Each obtained bin then represents a time range in which the human is supposed to perform activities following a clearly identifiable human process. This work has been extended with new experiments in [128].

R4-2 In [127] we introduce a fully automated log segmentation technique able to mark the beginning and end of each activity repetition in a sensor log. In order to obtain this result, the proposed technique employs the information about human position in the log to extract high-level actions (e.g., standing still or operating in a specific area of the house). Then, inactivity periods are analyzed in order to perform the first segmentation. Finally, clustering is employed to identify classes of segments representing activities. The work introduced in **R2-1** only focuses on temporal-based segmentation targeted at defining *habits*. Conversely, in this work, we focus on activities instead of habits, which allows for finer-grained control over human routines.

Once the methodologies in **R4-1** and **R4-2** and their related quantitative results are introduced, the need for a robust qualitative analysis of such results emerges. By bridging the gap between quantitative and qualitative analysis, a more comprehensive understanding of the problem is provided, offering a unique perspective that goes beyond mere numerical comparisons.

To perform this kind of qualitative analysis, several statistical procedures exist. The paired sample t-test, sometimes called the dependent sample t-test, is used to determine whether the mean difference between two sets of observations is zero. In a paired sample t-test, each subject or entity is measured twice, resulting in pairs of observations. Common applications of the paired sample t-test include case-control studies or repeated-measures designs. For instance, suppose you are interested in evaluating the effectiveness of a SW tool for educational purposes. One approach you might consider would be to measure the performance of a sample of users before and after completing the program and analyze the differences using a paired sample t-test (as done in [199]).

When there are two or more independent groups, the *Analysis of Variance* (ANOVA) technique applies. The ANOVA procedure is used to compare the sample sizes, sample means, and sample standard deviations in each of the comparison groups and determine whether there are significant differences between the means of the groups under analysis. We define the following objective:

O5 Execution of qualitative analyses with ad hoc statistical tools to support the results obtained through quantitative tests. This type of analysis aims to complete and reinforce those already obtained.

With reference to objective **O5**, the following research contributions have been achieved:

R5 To support the results already obtained in **R4-1** and **R4-2**, a qualitative analysis was carried out. Given the nature of the results, it was chosen to use ANOVA as a statistical tool to carry out these analyses. In this way, the models mined by three different mining algorithms (fuzzy, inductive, and heuristic) were compared with each other, and their statistical significance was determined.

The work of the author is published in the following papers. Each paper is labeled with the research challenges addressed.

[73] **Lucia Esposito, Silvestro Veneruso, Francesco Leotta, Flavia Monti, Jerin George Mathew, and Massimo Mecella** *Unsupervised Segmentation of Human Habits in Smart Home Logs Through Process Discovery.* 1st ITalian forum on Business Process Management (ITBPM 2021).
Research Contributions: **R4-1**

[129] **Francesco Leotta, Silvestro Veneruso** *VPM: Analyzing Human Daily Habits through Process Discovery.* Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration and Resources Track at BPM 2021 co-located with 19th International Conference on Business Process Management (BPM 2021).
Research Contributions: **R3**

[71] **Lucia Esposito, Francesco Leotta, Massimo Mecella, and Silvestro Veneruso.** *Unsupervised segmentation of smart home logs for human habit discovery.* 18th International Conference on Intelligent Environments (IE), 2022.
Research Contributions: **R4-1**

[28] **Yannis Bertrand, Silvestro Veneruso, Francesco Leotta, Massimo Mecella, and Estefanía Serral Asensio.** *NICE: The Native IoT-Centric Event Log Model for Process Mining.* Lecture Notes in Business Information Processing. Springer Verlag (Germany), 2023.
Research Contributions: **R2**

[195] **Silvestro Veneruso, Yannis Bertrand, Francesco Leotta, Estefanía Serral, and Massimo Mecella** *A model-based simulator for smart homes: Enabling reproducibility and standardization.* Journal of Ambient Intelligence and Smart Environments (JAISE), 2023.
Research Contributions: **R1-1**, **R1-2**

[25] **Yannis Bertrand, Bran Van den Abbeele, Silvestro Veneruso, Francesco Leotta, Massimo Mecella, and Estefanía Serral Asensio.** *A survey on the application of process mining to smart spaces data.* Process Mining Workshops: ICPM 2022 International Workshops.
Research Contributions: **R1-1**, **R3**

[24] **Yannis Bertrand, Bran Van den Abbeele, Silvestro Veneruso, Francesco Leotta, Massimo Mecella, and Estefanía Serral Asensio.** *A survey on the application of process discovery techniques to smart spaces data.* Journal of Engineering Applications of Artificial Intelligence (EAAI) 126, 2023.
Research Contributions: **R1-1**, **R3**

[127] **Francesco Leotta, Massimo Mecella, and Silvestro Veneruso.** *Unsupervised Segmentation of Smart Home Position Logs for Human Activity Analysis.* 2023 19th

International Conference on Intelligent Environments (IE). IEEE, 2023.
Research Contributions: **R4-2**

[128] **Francesco Leotta, Massimo Mecella, and Silvestro Veneruso.** *Discovering Human Habits through Process Mining: State of the Art and Research Challenges.* Activity Recognition and Prediction for Smart IoT Environments. Cham: Springer International Publishing, 2024.
Research Contributions: **R4-1**, **R5**

[197] **Silvestro Veneruso, Francesco Leotta, and Massimo Mecella.** *On the Usefulness of Human Behaviour Process Models: a User Study.* 2024 20th International Conference on Intelligent Environments (IE). IEEE, 2024.
Research Contributions: **R5**

Thesis Outline

- Chapter 1 introduces background concepts and definitions related to smart spaces, ambient intelligence, and process mining.
- Chapter 2 surveys existing approaches that apply process discovery to smart space data.
- Chapter 3 proposes a model-based simulator capable of generating synthetic datasets that emulate the characteristics of the vast majority of real datasets while granting trustworthy evaluation results. The datasets are generated using the eXtensible Event Stream (XES) international standard commonly used for representing event logs. This format is then further extended by the work presented in Chapter 4. Finally, the datasets produced by the simulator are validated against two real-life scenarios' logs from the literature.
- Chapter 4 presents the Native Iot-Centric Event (NICE) log, a new event log format designed to incorporate and represent IoT data into a process event log, ensuring traceability, flexibility, and limiting data loss. The new format is linked to the smart space data simulator discussed in Chapter 3 to generate synthetic logs. The presented format is evaluated against requirements previously established for an IoT-enhanced event log format, showing that it meets all requirements, contrary to other alternative formats. A final analysis of a synthetic log shows how IoT data can easily be used to explain anomalies in the process.
- Chapter 5 shows how concepts and techniques borrowed from the area of *process mining* can be used to ease the operation of segmenting a smart home log in an unsupervised manner, avoiding annoying labeling efforts from final users. In particular, we propose a bottom-up discretization strategy to automatically segment a smart home log into meaningful portions called *habits.*
- Chapter 6 proposes another unsupervised technique that segments smart home logs containing position sensor measurements. With respect to the technique discussed in Chapter 5, in this chapter, we focus on human activities instead of habits, which allows for finer-grained control over human routines. The proposed technique exploits information about the position of the human to automatically extract basic actions, which are then segmented on a temporal basis and clustered. The approach is evaluated against a seminal dataset from

the literature and a synthetic dataset produced by a smart home simulator. Finally, results are compared with a state-of-the-art method.
- Chapter 7 presents the results of the qualitative analysis carried out with ad hoc statistical tools on the output data obtained from the methodologies discussed in Chapters 5 and 6.
- Chapter 8 concludes the discussion and highlights future works that might arise on the basis of this thesis.

Additional Work

During the Ph.D. program, the author has been involved in other research projects not directly connected to the area of smart spaces.

NOTAE project. The author is actively involved in the *NOT A writtEn word but graphic symbols* (NOTAE) project[1]. The NOTAE project investigates the presence of graphic symbols in documentary records as a historical phenomenon from late antiquity to early medieval Europe. Graphic symbols are meant here as graphic signs (including alphabetical ones) drawn as a visual unit in a written text and representing something other than a word of that text.

Videogames, virtual reality, and educational tools. The use of videogames has become an established tool to educate users about various topics. Videogames can promote challenges, cooperation, engagement, motivation, and the development of problem-solving strategies, all of which have important educational potential. In [79, 196], the author presents the design and realization of several virtual reality (VR) tools that act as an interactive learning experience to improve user awareness of cybersecurity-related issues. In particular, in [199] the author reports the results of a user study showing that the videogame CyberVR is equally effective but more engaging as a learning method toward cybersecurity education than traditional textbook learning. Furthermore, in [198] the author proposes a real-time VDR application titled V-DOOR that leverages the features of Oculus Rift to create an immersive experience that enables customers to try on clothes virtually in the comfort of their own home rather than physically in the retail shop.

We report the contributions to these projects here in terms of published papers for sake of completeness, though the themes covered in their subjects are out of scope for this thesis.

[79] **Lauren S. Ferro, Andrea Marrella, Silvestro Veneruso, Massimo Mecella, and Tiziana Catarci.** *An interactive learning experience for cybersecurity related issues.* International Workshop on Human-Centered Cybersecurity (In conjunction with CHITALY 2019).

[196] **Silvestro Veneruso, Lauren S. Ferro, Andrea Marrella, Massimo Mecella, and Tiziana Catarci.** *A game-based learning experience for improving cybersecurity awareness.* CEUR WORKSHOP PROCEEDINGS. Vol. 2597. CEUR-WS, 2020.

[1] https://notae-project.digilab.uniroma1.it/

[198] **Silvestro Veneruso, Tiziana Catarci, Lauren S. Ferro, Andrea Marrella, and Massimo Mecella.** *V-DOOR: A Real-Time Virtual Dressing Room Application Using Oculus Rift.* Proceedings of the International Conference on Advanced Visual Interfaces, 2020.

[199] **Silvestro Veneruso, Lauren S. Ferro, Andrea Marrella, Massimo Mecella, and Tiziana Catarci.** *CyberVR: an interactive learning experience in virtual reality for cybersecurity related issues.* Proceedings of the International Conference on Advanced Visual Interfaces, 2020.

[23] **Tiziana Catarci, Antonella Ghignoli, Francesco Leotta, Massimo Mecella, Silvestro Veneruso, et al.** *Exploring the historical context of graphic symbols: the NOTAE knowledge graph and its visual interface.* CEUR WORKSHOP PROCEEDINGS, Vol. 2816. CEUR-WS, 2021.

[22] **Tiziana Catarci, Antonella Ghignoli, Francesco Leotta, Massimo Mecella, Silvestro Veneruso, et al.** *NOTAE: NOT A writtEn word but graphic symbols.* CEUR WORKSHOP PROCEEDINGS, Vol. 3144. CEUR-WS, 2022.

Chapter 1

Introduction

Recent years have shown a growing interest in the market for embedding sensors and actuators in physical environments to facilitate the execution of numerous tasks. The idea is to use sensors to collect real-time information about the environment and to use it to trigger actions through actuators in order to automate physical tasks, aiding humans in their daily lives. The term *pervasive* (or ubiquitous) *computing* is usually employed to indicate the set of techniques apt to this aim [206].

Smart environments, or smart spaces, represent an emerging class among pervasive computing application areas. Cook and Das [41] define a smart environment as "a small world where different kinds of smart devices are continuously working to make inhabitants' lives more comfortable". Smart environments aim to satisfy the experiences of individuals and improve their lives, by replacing hazardous work, physical labor, and repetitive tasks with automated agents, realizing the paradigm known as *Ambient Intelligence* (AmI) [29]. Examples include smart homes, smart cities, and smart factories.

An information system supporting AmI takes as input raw sensor measurements, analyzes them in order to obtain a higher level of understanding of what is happening in the environment, i.e., the current context, and eventually uses this information to trigger automated actions through a set of actuators, following final user preferences and needs. In particular, the context extraction and decision-making steps are supported by a set of models representing [126]:

- *Actions*: atomic interactions with the environment or a part of it (e.g., turning on the TV). Recognizing actions can be easy or difficult, depending on the sensors installed in the environment.

- *Activities*: groups of human atomic interactions with the environment (i.e., actions) that are performed with a final goal (e.g., cleaning the house). They can be collaborative, including actions carried out by multiple users, and can overlap with each other. *Human activity recognition* (HAR) is a common task in smart spaces that aims at recognizing various human activities (e.g., walking, sleeping, watching TV) using machine learning techniques based on data gathered from IoT environments [99]. Authors in [137] argue that HAR is part of a bigger picture with the ultimate aim of providing assistance, assessment, prediction, and intervention related to the identified activities. An example of this is *ambient assisted living* (AAL), which supports the elderly in their daily routines by using various technical systems (e.g., IoT devices) and aims at increasing the quality of life, wellbeing, and safety of those elderly people [64].

- *Habits*, *routines*, or *behavior patterns*: an activity, or a group of actions or activities that happen in specific contextual conditions (e.g., what the user usually does in the morning between 08:00 and 10:00).
- *Context*: the state of the environment, including the raw output of sensors and actuators, as well as the state of the inhabitants, consisting of the actions/activities/habits they are performing, and whatever high-level consideration can be automatically derived. In this very comprehensive sense, the term *situation* is sometimes used.

1.1 Classical ambient intelligence

Models of human habits and activities can be either manually defined (i.e., *specification-based* methods) or obtained through machine learning techniques (i.e., *learning-based* methods).

In specification-based methodologies, models are usually based on logic formalisms, which are relatively easy to read and validate (once the formalism is known to the reader), but their creation requires a major cost in terms of expert time and effort. In the learning-based case, the model is automatically learned from a training set (whose labeling cost may vary according to the proposed solution), but employed formalisms are usually not "explainable" due to the statistical techniques they are based on, making them less immediate to understand [88].

The effort and time of experts required by specification-based methods can rapidly become unsustainable if sensor measurements are directly used as atomic terms (i.e., basic modeling elements) of the models. As a consequence, such models usually employ high-level actions and events as basic terms. Learning-based methods usually directly refer to sensor measurements, thus losing focus on human actions and making it even more difficult to visually inspect and validate the produced models. On the other hand, taking as input raw sensor measurements usually makes learning-based methods easier to apply in a practical context, whereas, in the vast majority of cases, specification-based methods do not face (and solve) the problem of translating sensor measurements into actions.

Learning-based approaches can be further divided into three sub-categories:

1. *Supervised* learning is the machine learning task of inferring a function from labeled training data. The training data consists of a set of training examples. A vast collection of works employ supervised learning techniques in order to learn models of habits, activities, and simple actions.
2. *Unsupervised* learning techniques are mainly based on machine learning techniques that do not need the input data to be labeled.
3. *Weakly supervised* learning techniques need only a part of the input data to be labeled.

The practical applicability of techniques proposed in the literature is limited by the effort required by the final user to manually label smart space logs. Approaches based on supervised (or weakly supervised) learning require the logs to be labeled with markers denoting the onset and end of all (or at least of a consistent subset) of the occurrences.

Manual labeling of logs is perceived by the final users as annoying, which could ends up in imprecise labeling, possibly tampering with the performance of algorithms at runtime. Despite the fact that unsupervised techniques to label a smart home log have been proposed, they suffer from several limitations when applied to real-world scenarios [90]. First, many approaches (e.g., [112]) require manually specifying window lengths or other kinds of numerical thresholds (e.g., number of events, minimum distance between two events). The selection of such parameters is hard and does not take into account the peculiarities of the different activity and habit types. In second place, the vast majority of proposed solutions are directly applied to raw sensor measurements. This strategy does not exploit the meaning of a sequence of sensor measurements, only highlighting the statistical distribution of occurrences and co-occurrences of sensor events. Finally, in many cases, automatic segmentation techniques are only used to complement manual segmentation and are not intended to segment the full log (e.g., [46]).

Human-readable formalisms should be used alongside unsupervised machine learning techniques; this is the most important challenge in the field of AmI research. Applying process mining to smart spaces allows you to get the best of both worlds because processes are human-readable, formally grounded, and can be mined automatically.

The vast majority of activity or habit recognition techniques are supervised. This means that, in order to train the models, a smart space log must be segmented into sub-traces representing activities or habits with assigned labels. Many approaches in the literature are evaluated using datasets manually labeled by human operators, thus limiting the suitability of such techniques in a commercial application scenario where it is difficult to imagine a final user involved in annoying and imprecise training sessions. On the other hand, automatic segmentation methods available in the literature, aiming at partially or completely eliminating human effort in labeling, often show limitations related to the absence of domain knowledge. In this section, we will present automated approaches to log segmentation in the literature.

Authors in [16] introduce the concept of Active DataBase (ADB), composed of event-condition-action (ECA) rules. An ECA rule generally takes the form of "ON event IF condition THEN action", i.e., once a specific event is triggered and certain contextual conditions are satisfied, then the execution of an action is promptly executed. These have been extended in Augusto et al. (2008) with more complex temporal operators (ANDlater and ANDsim) and by adding uncertainty management. In [17], authors propose the APUBS algorithm to automatically mine ECA rules. The algorithm considers the typology of the sensors used in the measurements and the time relationships between their activations. Specifically, their approach relies on a distinct categorization of three types of sensors: *(i)* type O sensors that are embedded within objects (e.g., a sensor detecting the opening of a bathroom door); *(ii)* type C sensors that gather environmental data (e.g., temperature measurements); *(iii)* type M sensors that track the user's location within the house.

Events in the *event* component of an ECA rule always come from type O and type M sensors. *Conditions* are formulated based on the values acquired from type C sensors. The *action* component exclusively involves type O sensors, which can also operate as actuators. This collection of type O sensors is referred to as the *mainSeT*.

The APUBS methodology computes, for each sensor in the *mainSeT*, the *associatedSeT* composed of type M and O sensors that have the potential to be correlated as triggering events. This computation is carried out by leveraging the seminal *APriori* algorithm. Then, it discovers the temporal connections between the events present in *associatedSeT* and those within the *mainSeT*. In this stage, irrelevant relationships are eliminated, resulting in the extraction of the conditions necessary for the ECA rules.

In [55], authors extract ECA rules by using a modified version of APUBS. In particular, they used a variation of the seminal *APriori* algorithm employed in the original approach.

Instead, [46] presents an approach that relies on the principle of minimum description length (MDL) to automatically extract activity patterns. Their algorithm requires a dataset containing a sequence of sensor events representing human interactions with the smart environment. At each iteration, the algorithm searches for patterns that yield the best compression of the dataset. A pattern is defined as a sequence of sensor events and their occurrences within the dataset.

Initially, a single pattern is associated with each different sensor event. Subsequently, the algorithm iteratively attempts to expand these patterns, with the goal of achieving the most efficient compression. In particular, each occurrence of a pattern is replaced with a symbol that corresponds to the specific pattern in question. The algorithm stops once no additional compression can be achieved, returning all the patterns found. A clustering step is then implemented to identify and distinguish variations of human routines.

The method proposed by [46] is intended to implement a semi-supervised approach where automatic techniques are only employed to segment the part of the log not manually segmented by the final users. Nothing prevents the algorithm from being applied to an entire log in a fully unsupervised fashion.

Authors in [112] propose a sliding window-based methodology to perform real-time activity recognition from sensor data, enabling the recognition of activities as new sensor events are recorded. Knowing that different activities can be defined by different window lengths of sensor events, the authors consider the time decay and apply mutual information-based weighting to the sensor events within a window. Furthermore, they consider supplementary contextual information, such as the preceding activity and the activity from the previous window. This automatic log segmentation technique is only intended to be used at runtime, whereas at training time, more precise segmentation techniques (manually or automatically) are required.

Similarly to what was done by [112], [92], derive representations for sensor data, which are then aggregated into activity models. Also, [134] proposes a multi-task deep clustering framework. By employing unlabeled multi-dimensional sensing signals as input, they use a K-means clustering algorithm that relies on the extracted features to divide the log into distinct groups. This process generates pseudo-labels for the instances. However, these last two approaches rely on wearables, while in this thesis we focus on home sensors.

In [71], authors introduce a methodology that enables automatic segmentation of human habits. This is achieved by implementing a bottom-up discretization strategy specifically focused on the timestamp attribute of the sensor log. In particular, this approach relies on merging small-range portions of the original log, taking account of quality measures describing the structural behavior of the related process models. The rationale is that adjacent intervals are merged only if the mined models result in simpler and less structured models, i.e., if the underlying human habit is easy to read. Differently from the approach seen in [46], they focus on habits instead of activities. Additionally, in [46] patterns are extracted with the sole goal of recognizing them at runtime, without providing neither a visual analysis tool nor a structured description of human routines. This approach is described in detail in Chapter 5.

This last work, anyway, only focuses on temporal-based segmentation targeted at defining *habits*. In [127], the same authors deal with the segmentation challenge from another perspective: they focus on activities instead of habits, which allows for finer-grained control over human routines. This process can then be employed to later extract enactment rules. As for [71], they still employ the Visual Process Maps (VPM) system [125] to convert sensor measurements into actions. This approach is described in detail in Chapter 6.

The challenge of transforming low-level logs into high-level event logs, defined as event log abstraction (ELA) in [193], is also addressed in [203] and [40]. Here, the authors employ a reasoning algorithm to register a series of sensors and discover the related activities. Activity recognition often has a data structure resembling IoT data. In [165], the authors proposed a framework to transform location sensor data into an event log via interaction mining so that business users can understand what happened.

In [193], the authors identify three different families of ELA techniques:

1. *pattern-based*: local patterns in the log are identified and used to describe frequent behavior in an event log in terms of local process-like patterns [184, 180, 149];

2. *session-based*: this type of abstraction is based on the idea that traces are divided into batch sessions, and each session is abstracted into a high-level activity execution. Then sessions are clustered based on similarity [122, 119];

3. *hierarchical ELA*: a framework takes a log at level i and discovers the applicable abstraction classes at level $i+1$. A sub-log is generated for each class, containing the sub-process data representing that class. A hierarchy of abstractions is obtained by iteratively applying the framework until the user-specified level is reached [130].

Recently, deep generative models have become dominant for unsupervised learning. There are two distinct phases to the activity recognition process: *(i)* feature extractors, which are often deep generative models, receive the input data beforehand in order to pre-train them to extract features; *(ii)* a top-layer or alternative classifier is incorporated and subsequently trained in a supervised manner using labeled data for classification. The feature extractor's weights may be adjusted throughout the

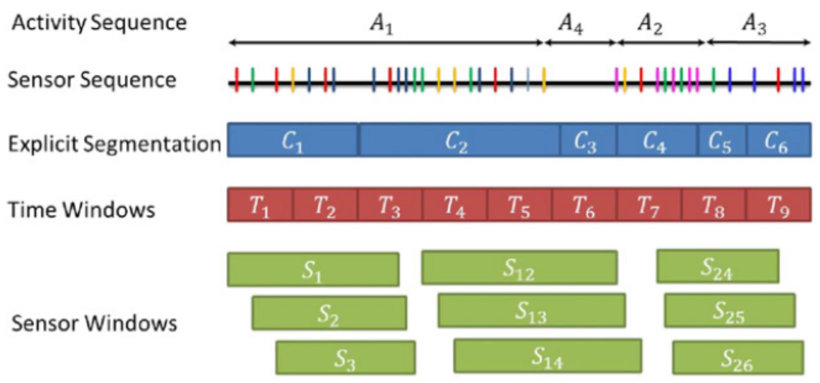

Activity Sequence
A_1
A_4
A_2
A_3
Sensor Sequence
Explicit Segmentation
C_1 C_2 C_3 C_4 C_5 C_6
Time Windows
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9
Sensor Windows
S_1 S_2 S_3 S_{12} S_{13} S_{14} S_{24} S_{25} S_{26}

Figure 1.1. Illustration of the different segmentation approaches discussed in [112].

supervised training. In [9], deep belief networks (DBN) activity recognition models are implemented. In [150], authors proposed to use autoencoders for unsupervised feature learning as an alternative to *principal component analysis* (PCA) for activity recognition in ubiquitous computing. Compared to discriminative models, deep generative models are more robust against overfitting problems [141].

Although deep learning models have demonstrably succeeded in unsupervised learning for human activity recognition (HAR), you still need to provide labeled samples as ground truth. Consequently, these approaches are more accurately categorized as semi-supervised learning, where both labeled and unlabeled data are utilized for neural network training [39].

1.1.1 Sensor data aggregation methodologies

Performing learning techniques from sequences of raw sensor measurements require to group them into aggregates of interests (i.e., actions, activities, and habits). This is a fundamental and crucial task, even if the most proposed approaches in the literature ignore this aspect (especially *supervised* learning ones). Windowing mechanisms are required, and, as described in [112], we can identify three main classes:

- **Explicit segmentation:** the sequence of sensor data is divided into chunks, each chunk possibly corresponding to an activity/habit. Finding the appropriate chunk size for learning the activity/habit models during the training phase is not trivial. Typically, a pre-segmented sequence of sensor events that corresponds to an activity is used to train the classifier. However, a possible drawback is that an instance of activity/habit could be broken down into two or more chunks (Figure 1.1 shows how the activity A_1 is broken down to chunks C_1 and C_2). Since the chunks could not necessarily represent the entire sequence of sensor events for a particular activity/habit, the performance of the classifier is thus lowered, i.e., there could emerge instances of single activities/habits being divided into multiple chunks and multiple activities/habits being merged.

- **Time-based windowing:** this approach splits the entire sequence of sensor data into equal-sized time intervals (Figure 1.1 shows the equal-sized chunks $T_1, T_2, ..., T_9$). It further reduces the computational complexity of the explicit segmentation process. This is a good approach when dealing with data obtained from sources that operate continuously in time (e.g., data from accelerometers and gyroscopes). However, the choice of the window size is fundamental, as a small window size could not contain any relevant activity or habit; on the other hand, a wide window size could involve multiple activities or habits, and the instance that dominates the time interval will influence the classification decision. In the extreme case an interval could not include any sensor data (e.g., T_6 in Figure 1.1).

- **Event-based windowing:** this last approach splits the entire sequence into windows containing an equal number of sensor measurements. Trivially, the duration of the windows varies from one window to another. Furthermore, by dividing the sequences on a quantity basis and not on a time basis, we avoid the problem of having "silent" windows, i.e., windows where no actions are performed by the resident and consequently no sensors are triggered. However, it shows drawbacks similar to those introduced for time-based windowing, i.e., the identified window may not contain any relevant information or may contain more than one activity or habit instance affecting the decision-making process. Furthermore, in the presence of multi-resident environments, sensor firings from two different activities performed by different inhabitants will be grouped into a single chunk, thereby introducing conflicting influences for the decision task.

1.1.2 Sensors in smart spaces

In the last few years, sensing technologies have made significant progress. Sensors can thus be embedded in an environment and integrated into everyday objects and human bodies without affecting users comfort. An initial rough division can be made between *physical* sensors, which provide information about the environment (e.g., humidity, brightness, temperature), the devices, and the users, and *digital* ones, which provide information such as user calendars and weather forecasts. Different types of sensors provide different types of information that are useful for different purposes [17]. They define three distinct types of sensors: *object sensors* (type O), *context sensors* (type C), and *motion sensors* (type M). In [156], authors argue that activities can also be captured using *wearable sensors* (type W) and *video-based systems* (type V). A short overview of each sensor type:

- *Type O* sensors are installed in objects and provide information about the actions a user may perform (e.g., a sensor in the TV that is triggered when the device is turned on).
- *Type C* sensors provide information about the context in which the user performs actions (e.g., a temperature or light detector in the kitchen).
- *Type M* sensors capture motion and indicate where a user is located (e.g., a motion sensor detecting the passage in the hallway).

- *Type W* sensors offer a variety of information, including medical (e.g., smartwatch) and location (e.g., smartphone).
- *Type V* sensors capture video information (e.g., security cameras).

1.2 Process mining

Business process formalisms can be used to model human activities and habits in smart spaces [124]. A *business process* is a set of interrelated tasks performed in a company in order to conduct specific functions (e.g., ordering goods). In order to acquire such models, *process mining* techniques can be employed. Process mining (PM) [4] is a fairly recent research discipline that combines data mining techniques with techniques used in Business Process Management (BPM) [67]. Its main objective is to extract meaningful information from event logs.

PM makes use of several techniques to analyze business processes by mining traces left by their execution in information systems [3]. These traces take the form of so-called *event logs*, recording all events that happened in the execution of the process. An event log requires at least three components: 1) a *case identifier*, relating events to the process instance they were executed in; 2) a *timestamp*, specifying when the event happened; 3) an *activity label*, indicating which activity was executed [3]. There are several types of PM techniques:

- *Process discovery* is used to discover the process model describing the behavior recorded in the related event log. Thus, it takes as input an event log and automatically generates the correspondent process model without using any additional information. There are many algorithms that can be used for process mining discovery. The most known ones are the *fuzzy* miner, the *inductive* miner, the *alpha* miner, and the *heuristic* miner. Depending on the kind of discovery algorithm chosen and the parameters used in the algorithm, the output process model has specific characteristics and may be represented as a Petri net (see Section 1.2.2) or in other formats, such as BPMN (Business Process Modeling Notation), EPCs (Event-Driven Process Chains), and others.
- *Conformance checking* is used to check some properties of the given process model. It takes as input an existing process model and an event log belonging to the same process and compares them to identify deviations from the desired process. It can be used, for instance, to check if the actual behavior recorded in the log conforms to the model. In this way, we can detect possible deviations between the process model and reality. It can also be used for models discovered from the event log itself.
- *Enhancement techniques* that improve and refine an existing process model with information recorded in the event log. As for the conformance checking, it takes as input an existing process model and an event log belonging to the same process. However, in this case, the objective is to extend or improve the existing process model, considering the actual behavior of the process that is recorded in the log. Some types of enhancement are “repair”, which changes the model so that it better reflects the real process behavior, and “extension”,

which adds to the model some new information to make other kinds of analyses, such as identifying bottlenecks and computing throughput times.

Although initially focused on the analysis of traditional business processes, e.g., claims management processes, the scope of PM has expanded over time. For instance, process mining has been applied to analyze healthcare processes [142] and even blockchain processes [106]. One recent trend in PM is to leverage the development of IoT devices to extract event logs from sensor and actuator data. Indeed, some processes that are not supported by an information system generate large amounts of IoT data at runtime. Extracting an event log from IoT data enables the analysis of a whole range of processes previously out of the scope of PM, e.g., mining processes [33] and human behavior [172].

1.2.1 Process discovery: model quality metrics

Process discovery algorithms should find a proper balance to avoid the so-called *overfitting* and *underfitting* problems, i.e., to mine a general model that allows for behavior that is not recorded in the given log but that is related to the same process. To determine the quality of a model obtained through process discovery, four different dimensions are considered:

- *Fitness*: it measures how much behavior present in the log can be replayed in the discovered model. If a model has perfect fitness, it means that it represents all the behavior present in the log, i.e., all the traces can be successfully replayed. There are many definitions of fitness: it can be used to measure how many traces of the event log are replayed (case level) or how many events of the log are actually possible, according to the model (event level).

- *Precision*: it is used to avoid the *underfitting* problem. An underfitting model allows for behavior that is not related to the one represented in the model itself. This means that it has a low value of precision and, thus, does not accurately represent the specific behavior observed in the log; instead, it is too generalized and allows for behaviors very different from the ones represented in the log.

- *Generalization*: it is used to avoid the *overfitting* problem, which is the opposite of underfitting. An overfitting model is one that has a low value of generalization, so it allows only for the behavior observed in the log, i.e., it is too specific. We should avoid models that do not generalize because they do not represent the actual process but only the example behavior recorded in the log.

- *Simplicity*: it is based on *Occam's Razor*'s principle, which states that "one should not increase, beyond what is necessary, the number of entities required to explain anything." This means that we should find the simplest model among the ones describing the behavior observed in the log. A practical usage of this quality measure is described in Chapter 5.

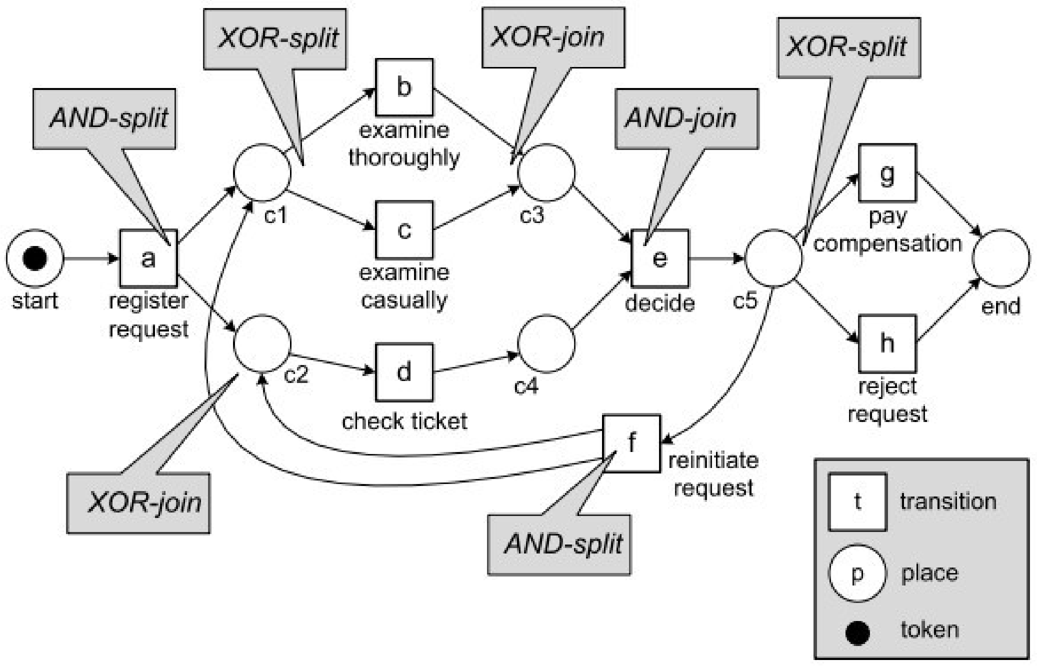


Figure 1.2. Example of a Petri net. Picture taken from [190].

1.2.2 Petri nets

Petri nets are one of the most commonly used notations for representing process models. They have an intuitive graphical notation and allow for the modeling of the concurrency of events. If two events can be concurrently executed, it is fundamental to discover this behavior if we want to avoid the so-called "spaghetti models", in which the same activity is duplicated and the whole graph is complex and difficult to understand and read.

A Petri net is a bipartite graph consisting of places and transitions, where places and transitions are nodes that are connected through direct arcs [190]:

Definition 1.1 (Petri net)**.** *A Petri net is a triplet $N = (P, T, F)$ in which:*

- *P is the finite set of places*
- *T is the finite set of transitions, where $P \cap T = \oslash$*
- *F is the finite set of directed arcs such that $F \subseteq (P \times T) \cup (T \times P)$*

Each transition and each place in a Petri net is a node. In particular, given two nodes x and y, we say that x is an input node of y if and only if there exists a direct arc that goes from x to y. In the same way, y is the output node of x.

Petri nets can easily model the concurrency of two or more events that must be all executed and may happen in any order through the *AND-join* and *AND-split* constructs. Instead, the choice among two or more events is modeled through the *XOR-join* and *XOR-split* constructs, so only one event is executed (see Figure 1.2).

Timestamp	SensorID	Value
...	...	...
2022-05-31 12:34:52	M003	ON
2022-05-31 12:34:58	M005	OFF
2022-05-31 12:35:04	M003	OFF
2022-05-31 12:35:22	T002	22
2022-05-31 12:38:17	M029	OFF
...	...	...

Table 1.1. Example of a sensor log used in smart spaces.

1.3 Ambient intelligence and process mining

The rationale behind applying process mining (PM) in a smart space is to exploit the vast set of data mining techniques targeting classical business processes to so-called cyber-physical processes [115]. As pointed out in [124], the application of PM to this scenario is not trivial. Before discussing technical differences, we first introduce a couple of differences in terms of the terminology employed between the smart space and PM research communities:

- The term *business process* in PM may correspond to different smart space concepts such as *activities, habits, routines* or *behavioral patterns.*
- The use of the term *activity* is a frequent source of confusion, as the smart space community uses this term to refer to one particular type of human process (e.g., sleeping, ordering food, walking), whereas the PM community uses this term to denote the basic units of a process. Therefore, in order to address this common misunderstanding, we will use the terms *activity* and *action*, as intended by the smart space community (consider the terminology introduced at the beginning of this chapter).

From a more practical point of view, many of the challenges are related to the difference between *sensor logs* produced by smart spaces and *event logs* produced by information systems, which are usually fed as input to PM algorithms.

We can imagine a smart space producing, at runtime, a *sensor log* containing raw measurements from available sensors.

Definition 1.2 (Sensor Log)**.** *Given a set S of sensors, a sensor log is a sequence of measurements of the kind $\langle ts, s, v\rangle$ where ts is the timestamp of the measurement, $s \in S$ is the source sensor, and v is the measured value, which can be either nominal (categorical) or numeric (quantitative).*

Measurements can be produced by a sensor on a periodic basis (e.g., temperature measurements) or whenever a particular event happens (e.g., door openings). An example of a sensor log is shown in Table 1.1.

As many of the algorithms proposed in the literature borrow the terminology of data mining, the sensor log could be conceived as a sequence of events instead of a sequence of measurements. Hence, we can introduce an alternative definition of a sensor log as an event log:

Definition 1.3 (Event Log)**.** *Given a set $E = \{e_1, \dots, e_{n_E}\}$ of event types, an event sequence is a sequence of pairs $\langle e, t \rangle$, where $e \in E$ and t is an integer, the occurrence time of the event type e.*

Whereas events in event logs record the execution of tasks, e.g., their start and their completion, sensor logs contain fine-grained sensor measurements, e.g., the temperature in a room at a certain point in time or the presence of a user located near a piece of furniture.

The possibility to model and analyze a process at different levels of granularity is not new in the community and was recently acknowledged as one of the major challenge to be addressed in the PM research area [20]. In [193], the authors propose an empirical evaluation of event log abstraction (ELA) techniques in process mining, i.e., the task of transforming a low-level log into a higher-level event log, where events are grouped into more abstract concepts to increase the understandability of the processes.

A sensor log, though, is not simply an extremely fine-grained equivalent of an event log, as while events in event logs are explicit traces of process task executions, sensor measurements are often only loosely related to tasks. Consider, for example, the measurements generated by Presence InfraRed (PIR) sensors which are frequently available in smart space datasets. A single PIR may be triggered during the execution of several different tasks, and during the execution of a single task, different PIR sensors can be triggered in any order, where the exact sequence does not really provide any additional information about the task itself. Turning sensor measurements into events is a complex challenge that can hardly be solved without human manual labeling, additional knowledge, or probabilistic reasoning [192].

While an important pre-processing step has to be performed to obtain high-level quality event logs from information systems too, e.g., to select a case notion or resolve data quality issues, the pre-processing of sensor logs is much more complex. In order to apply techniques from the PM area, a sensor log must be converted into an event log containing at least three elements [158]: *(i)* the case id, which identifies a specific process instance; *(ii)* the label of the task performed; and *(iii)* the timestamp. Since *(i)* and *(ii)* are absent from the sensor log, the conversion from a sensor log to an event log usually consists of two steps: *(1)* bridging the gap between sensor measurements and events in order to derive the task label, and *(2)* segmenting the event log into traces in order to assign a case ID to each event.

While event logs are typically supposed to be split into traces (process executions), sensor logs are not segmented and may contain information related to different processes or habits performed simultaneously.

In addition to this, new issues appear when tackling multi-user environments, i.e., smart spaces where several individuals could contribute to the execution of an activity (e.g., two people collaboratively cleaning the house). In these cases, it might be difficult to identify people associated with specific actions and, in turn, to analyze collaborative processes. The presence of multiple people also has an impact on the above-mentioned segmentation problem, as multiple independent processes may be carried out simultaneously.

The smart space community usually addresses the presence of multiple users by using invasive labeling techniques such as those based on wireless beacons [155] or cameras [123]. The same techniques could be employed, in principle, to associate an identified user with each task of the event log, but in some cases they cannot be applied because of privacy or comfort issues. In these latter cases, indirect identification methods, such as those based on tracking, can be applied [124].

1.3.1 Modeling the human behavior in smart spaces

As mentioned before, we distinguish two main types of human behavior in smart spaces: *activities* and *habits*. What distinguishes them is primarily the importance of intention in triggering and guiding behavior: activities are performed in order to attain a conscious goal, whereas habits are executed automatically in specific contexts, with a lower degree of consciousness. This distinction is in line with findings in behavioral psychology, which suggest that habits follow a specific model of action where intention is blurred by the automaticity of routine [145, 65].

Human behavior, and more specifically, human activities and habits, are flexible and unstructured. Given their intrinsic nature, in [124], authors proposed to represent them via a workflow and discussed approaches that model habits and activities using techniques derived from the PM area. The application of PM in smart spaces requires modeling human behavior as a business process (BP) [172]. The similarity is straightforward: a BP is similar to an activity, as they are both composed of a series of tasks that involve actors and decisions, and they both have a specific goal or desired outcome [68]. One could see an activity as a rather simple BP, usually executed by a single actor. However, there are also several important differences between human behavior and business processes [58]:

- *Context-dependency*: the impact of context on BP execution is often assumed to be limited in BPM but cannot be ignored in smart spaces, especially when considering human habits.
- *Variability*: activities are typically less structured than BPs, in the sense that a particular human behavior can be realized with more equivalent execution pathways than a BP, among other reasons because there is no enforcement of a model.
- *Repetitions*: an activity or habit can be repeated throughout the day in different circumstances or with a different sequence of activities (e.g., preparing a sandwich or cooking fish both come down to preparing a meal).
- *Concurrency*: a human being can execute several activities or habits simultaneously or stop an activity to perform another one before resuming the execution of the first one.

Chapter 2

Process discovery in smart spaces: a literature review

While both process mining (PM) and smart spaces have been evolving quickly as separate fields of study during the last few years, researchers have recently explored combining both disciplines and obtained interesting results that should be analyzed and compared in order to move forward. Applying PM techniques, and in particular *process discovery*, to smart spaces data enables modeling and visualizing human habits and activities as processes [124]. However, even though process models could be extracted from smart spaces data, multiple important challenges arose when applying techniques designed for business processes (BP) to human behavior [124]: *(i)* choosing or designing the proper modeling formalism for representing human behavior, *(ii)* abstracting the gap between sensor and event logs, *(iii)* segmenting logs into traces to be able to apply PM techniques, *(iv)* dealing with multi-user environments, and *(v)* addressing the continuous evolution of human behavior.

This chapter provides an overview of the current approaches in literature that handle the aforementioned challenges and to what degree they are addressed.

2.1 Applying process discovery to smart spaces data

This section surveys existing approaches that apply process discovery to smart spaces and analyzes how they deal with the following challenges identified in Chapter 1.

To perform the survey, a systematic literature review protocol was followed to maximize the reproducibility, reliability, and transparency of the results [105]. The protocol consists of six phases: (1) research question specification, (2) search criteria definition, (3) study identification, (4) screening, (5) data extraction, and (6) results. Figure 2.1 shows the number of studies reviewed and excluded in each phase and the reasoning behind the exclusion.

Research questions

In this work, we highlighted the following research questions (RQs), focusing on the challenges identified in [124]:

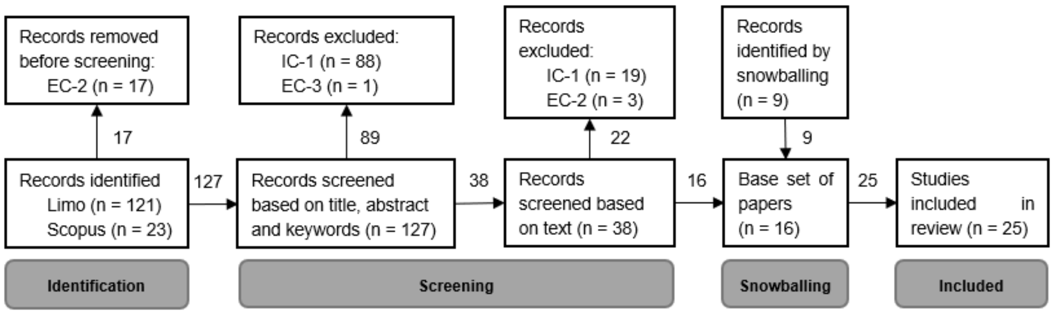

Records removed before screening: EC-2 (n = 17)
Records excluded: IC-1 (n = 88) EC-3 (n = 1)
Records excluded: IC-1 (n = 19) EC-2 (n = 3)
Records identified by snowballing (n = 9)
17
89
22
9
Records identified Limo (n = 121) Scopus (n = 23)
127
Records screened based on title, abstract and keywords (n = 127)
38
Records screened based on text (n = 38)
16
Base set of papers (n = 16)
25
Studies included in review (n = 25)
Identification
Screening
Snowballing
Included

Figure 2.1. Search methodology: overview of the included and excluded papers.

- **RQ-1**: how do primary studies represent human behavior? The study of a proper formalism to represent human behavior has been extensively studied in the smart spaces' literature for many years (see, e.g., [168]), but separated from the PM literature. Also, within the Business Process Management field, specifically in the BP modeling phase, we can find in the literature many modeling proposals for IoT BPs [187], but they are not specifically designed for human behavior or for PM. As such, the selection or design of a proper formalism to be used to represent human behavior in PM is still a research gap, as highlighted by [58]. In this RQ, we will analyze which specific formalisms have been used to model the human behavior output of the existing PM techniques.

- **RQ-2**: how do PM techniques address the gap between sensor events and process events? Nearly all PM techniques are based on event logs that represent high-level data (i.e., tasks easily understandable by the average user) [98]. However, most sensors used in smart spaces output low-level data that represents sensor events (i.e., raw data that has little meaning to the average user) [98, 102]. As a result, the application of process mining techniques to smart space data first requires the sensor logs to be translated into event logs [178, 177]. This RQ will analyze how existing techniques transform low-level sensor events into high-level process events [178, 182].

- **RQ-3**: how do PM techniques tackle logs that are not split into traces? Although PM requires the log to be segmented into traces, it cannot be assumed that this is automatically the case for logs produced by a smart space [124]. In addition, the abundance of available data automatically collected at a very high frequency by the sensors installed in the smart space makes it difficult to perform the segmentation manually [59]. In this RQ, we will investigate how current PM techniques applied to smart space data are dealing with the segmentation of logs into traces.

- **RQ-4**: how do PM techniques tackle multi-user environments? Multiple users can perform actions separately but also collaboratively at any given time [126]. While it is very common to have multiple users in a smart environment, most datasets do not include information about the user or users that triggered an event. In this RQ, we will investigate if and how current PM techniques deal with data collected from more than one user.

- **RQ-5**: how do PM techniques tackle the evolution of routines over time? While both human routines and business processes can change over time, human routines tend to be more flexible and less structured. As a result, human routines vary more frequently, often based on contextual information that is not present in the sensor logs (e.g., the inhabitant has changed jobs and now works from home regularly). This RQ will investigate if and how existing techniques consider concept drift when applying PM.

Search criteria and studies identification

Since this survey is about applying process discovery techniques to model human behavior from smart space data, three groups were identified: group 1 represents process discovery, group 2 represents human behavior modeling, and group 3 represents the smart space environment. Frequently used synonyms were added to ensure full coverage of the relevant literature on each topic, yielding the following search query (SQ):

SQ: (“process mining" OR “process discovery") AND (“behaviour pattern" OR "behavior pattern" OR “habit" OR “routine" OR “activity of daily living" OR “activities of daily living" OR “daily life activities” OR “daily-life activities” OR “daily behaviour” OR “daily behavior”) AND (“smart space" OR “smart home" OR “smart environment" OR “smart building")

The base set of papers was identified by searching the title, abstract, and keywords using the Scopus and Limo online search engines, providing access to articles published by *Springer, IEEE, Elsevier, Sage, ACM, MDPI, CEUR-WS*, and *IOS Press.*

Screening

The papers identified by the search string must pass a quality and relevance assessment in order to be included in the survey. The assessment consists of exclusion and inclusion criteria. The exclusion criteria (EC) are defined as follows:

- **EC-1**: the study is not written in English.
- **EC-2**: the paper is a duplicate of an item already included in the review.
- **EC-3**: the study is a survey or literature review primarily summarizing previous work where no new contribution related to the research topic is provided.

The inclusion criterion (IC) is defined as follows:

- **IC-1**: the study is about discovering and modeling human behavior using PM techniques based on smart space data and answers at least one research question.

The first set of primary studies was formed by all articles that remained after the inclusion and exclusion criteria screening. Once these studies were selected, forward and backward snowballing was performed. Articles identified through snowballing were screened using the same criteria.

Data extraction

First, generic information was extracted, such as title, authors, year of publication, etc. (see Table 2.1). Afterwards, the research questions were answered based on the content of each article.

2.1.1 Results

In this section, we present the results of the systematic literature review. We start with a general description of the primary studies and their smart space setting before covering the five RQs one by one.

Primary studies' settings

Table 2.1 gives an overview of the 25 primary studies identified in this survey. The studies are shown ordered by the year of publication of the work, from oldest to newest, covering a ten-year period from 2013 to 2022, with at least one study per year. It also reports essential information about the settings used by each study, specifically:

- *Environment*: the facility and/or the area in which the data are collected, i.e., the smart space outfitted with sensors and actuators to collect data. Among the selected studies, the most common choice are smart homes, covering 19 out of these studies. In addition, three studies cover healthcare environments (e.g., a nursing home); two studies cover commerce platforms (e.g., a shopping mall); and two studies cover the workplace environment, such as an office. The study S8 stands out as it covers multiple environments: building, home, office, street, and transportation.

- *Dataset(s)*: the dataset(s) used to validate the approach described in the study. Among the selected studies, 14 use a freely available state-of-the-art dataset (the related reference is provided); ten of them validate their methodology with data collected on their own; S22 and S23 use synthetic data produced by a smart home simulator [195]. Two studies use multiple datasets (S6 and S8).

Table 2.2 provides additional information on the 19 datasets used across the 25 primary studies analyzed in this work. We marked as N/A the information that could not be obtained from the papers. This table shows:

- *Dataset*: the reference to the dataset.

- *Employed by*: it indicates which studies (from Table 2.1) used the dataset as a validation benchmark.

- *Origin*: it defines the origin of the data collection: *synthetic* (produced in the context of a simulation) or *real* (collected from real settings). Among the selected studies, only three of them (S7,S22 and S23) used synthetic data for their validation phase.

Table 2.1. Overview of included primary studies.

ID	Ref	Title	Year	Environment	Dataset(s)
S1	[76]	Process Mining for Individualized Behavior Modeling Using Wireless Tracking in Nursing Homes	2013	Healthcare	Own
S2	[36]	Learning and Recognizing Routines and Activities in SOFiA	2014	Office	Own
S3	[37]	Incremental Learning of Daily Routines as Workflows in a Smart Home Environment	2015	Home	[43]
S4	[76]	Process mining methodology for health process tracking using real-time indoor location systems	2015	Healthcare	Own
S5	[59]	Process-Based Habit Mining: Experiments and Techniques	2016	Home	Own
S6	[181]	Heuristic approaches for generating Local Process Models through log projections	2016	Home	[194, 32, 140, 148]
S7	[183]	Event Abstraction for Process Mining Using Supervised Learning Techniques	2016	Home	[194], [30]
S8	[175]	Self-tracking reloaded: applying process mining to personalized health care from labeled sensor data	2016	Multiple	Own
S9	[35]	Discovering Process Models of Activities of Daily Living from Sensors	2017	Home	[47]
S10	[135]	Revealing daily human activity pattern using PM approach	2017	Home	[148]
S11	[173]	Addressing multi-users open challenge in habit mining for a PM-based approach	2018	Home	-
S12	[179]	Generating time-based label refinements to discover more precise process models	2019	Home	[194]
S13	[62]	Analyzing of Gender Behaviors from Paths Using Process Mining: A Shopping Mall Application	2019	Commerce	Own
S14	[63]	Individual behavior modeling with sensors using process mining	2019	Home	Own
S15	[34]	Extraction of User Daily Behavior From Home Sensors Through Process Discovery	2020	Home	[47]
S16	[125]	Visual process maps: a visualization tool for discovering habits in smart homes	2020	Home	[43]
S17	[186]	Process Mining for Activities of Daily Living in Smart Homecare	2020	Healthcare	[174]
S18	[60]	Discovering Customer Paths from Location Data with Process Mining	2020	Commerce	Own
S19	[98]	Process Model Discovery from Sensor Event Data	2020	Home	[44]
S20	[154]	A Multi-case Perspective Analytical Framework for Discovering Human Daily Behavior from Sensors using Process Mining	2021	Home	[176]
S21	[133]	Interactive Process Mining in IoT and Human Behaviour Modelling	2021	Home	Own
S22	[167]	Supporting Users in the Continuous Evolution of Automated Routines in their Smart Spaces	2021	Home	[195]
S23	[120]	The Benefits of Sensor-Measurement Aggregation in Discovering IoT Process Models: A Smart-House Case Study	2021	Home	[195]
S24	[71]	Unsupervised Segmentation of Smart Home Logs for Human Habit Discovery	2022	Home	[43]
S25	[61]	Understanding Patient Activity Patterns in Smart Homes with Process Mining	2022	Home	Own

Table 2.2. Overview of the datasets employed in the studies involved in this work (N/A stands for not available).

Dataset	Employed by	Origin	Available sensors	Values	#users	Length	Labeling	#ADLs	#sensor measurements	#ADL instances
[76]	S1	Real	9 location bracelets	Discrete	9	25 weeks	N/A	N/A	125k	N/A
[36]	S2	Real	Motion, brightness, light, touch, magnetic, temperature, pressure	Mixed	1	45 days	Yes	11	90k	450
[43]	S3,S16,S19,S24	Real	Motion, temperature, magnetic	Mixed	1	220 days	Yes	11	1720k	6.5k
[76]	S4	Real	Proximity	Discrete	N/A	3 months	No	-	39k	-
[59]	S5	Real	Motion (CASAS sensor kit [44])	Discrete	2+guests	4 months	Partial	N/A	N/A	N/A
[32]	S6	Real	wrist-worn accelerometer recordings	Mixed	16	N/A	Yes	7	N/A	N/A
[148]	S6,S10	Real	3 PIR, 4 door, 1 flush, 2 pressure, 2 electric	Mixed	1-2	14 days	Yes	12	N/A	N/A
[175]	S8	Real	Accelerometer, device orientation, GPS sensor data	Mixed	7	2 weeks	Yes	8-37	N/A	N/A
[47]	S9,S15	Real	Motion, temperature, magnetic	Mixed	1+pet	58 days	Yes	11-15	433k	2318
[194]	S6,S7,S12	Real	14 switch sensors on devices and doors	Discrete	1	28 days	Yes	7	2120	245
[30]	S7	Synthetic	Not specified	N/A	1	30 days	Yes	4	40k	N/A
[62]	S13	Real	Proximity (iBeacons devices)	Discrete	642	N/A	No	-	780k	-
[63]	S14	Real	Motion	Discrete	25	68 to 332 days	No	-	N/A	-
[174]	S17	Real	N/A	N/A	Several (not specified)	N/A	Yes	16	N/A	N/A
[60]	S18	Real	Proximity (iBeacons devices)	Discrete	1724 unique customers	8 months	No	-	2000	N/A
[176]	S20	Real	84 activation/deactivation or opening/closing	Discrete	1	16 days	Yes	5	5545	179
[133]	S21	Real	Motion	Discrete	1	70 days	Yes	N/A	80k	N/A
[195]	S22,S23	Synthetic	Configurable within the simulation tool	Mixed	1-2	21 days	Yes	9	25k	788
[61]	S25	Real	Proximity	Discrete	1-3	N/A	No	-	N/A	-

- *Available sensors*: each dataset usually contains data collected from a set of sensors. Sensors can be roughly divided between those providing information about the environment (i.e., environmental sensors) and those providing direct information about human behavior (i.e., behavioral sensors). Nonetheless, environmental sensors may provide indirect information about humans (e.g., an increasing temperature measurement in the bathroom can be the result of a human having a shower) and vice versa. Information obtained by sensors is always affected by a certain degree of uncertainty and imprecision due to noise and intrinsic characteristics of the sensor (e.g., a presence infrared sensor provides information about the position of a human but not information about which action the human is performing, with the exception of what can be inferred by the position of the sensor). Among the selected studies, motion sensors are the most common choice; 16 works use them. S17 did not specify the kind of sensor(s) employed in their work.

- *Values*: values provided by sensors can be either discrete (e.g., switch sensors), continuous (e.g., temperature sensors), or mixed if both types are present in the dataset.

- *#users*: the number of subjects involved in the dataset acquisition process can have a profound impact on tasks usually performed in a smart environment. As an example, when sensors do not directly allow them to recognize which user is responsible for triggering a specific sensor event, activity and habit recognition can be very complex.

- *Length*: the size of the dataset, in terms of days, weeks, or months of acquisition, which affects the robustness of the extracted models or the validity of the performance evaluation.

- *Labeling*: some datasets are labeled with a certain set of activities of daily living (ADLs). Having ADL labels allows you to perform ADL-specific tasks such as recognition and prediction. Labels usually refer to which ADL is performed in a specific time range. The labeling could cover the data collected in its entirety or just partially; in both cases, specific ADL tasks could be implemented. Among the selected studies, 15 of them worked on fully or partially labeled data, while eight of them worked on unlabeled data. On studies S1 and S11, we cannot derive any useful information about this topic.

- *#ADLs*: the number of different activities included in the labeled dataset for the evaluation phase. Excluding unlabeled ones and datasets for which we cannot infer any information about the labeling (i.e., S1 and S11), we have a minimum number of labeled ADLs equal to 5 and a maximum number of labeled ones equal to 37 in S8, where the own dataset contains optional sub-activity labels, e.g., to specify which meal the "eating" activity refers to.

- *#sensor measurements*: the number of raw sensor measurements in the dataset, which in their entirety compose the sensor log.

- *#ADL instances*: the number of instances of ADL in the dataset (if the dataset is labeled). Each instance groups together a sequence of sensor events (e.g., the *cooking* activity is composed of the measurements of the sensors triggered by the subject during the performance of the activity itself). Such grouping is reflected in the difference in terms of numbers between sensor events and ADLs. For example, in the *aruba* dataset [43] the number of ADL instances is 6.5k against the 1720k sensor events.

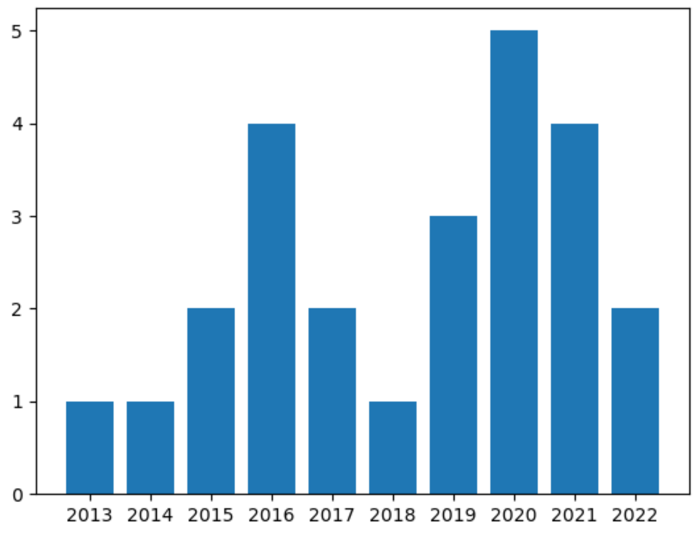


Figure 2.2. Number of publications per year

Figure 2.2 shows the publication trend over the years, which shows the number of papers on the topic has considerably increased in the last 5 years.

2.1.2 Modelling formalisms

An overview of the modeling formalisms used by the primary papers is shown in Figure 2.3 (note that some papers used more than one modeling language, e.g., to compare the output of several PM techniques). Petri Nets (see Section 1.2.2) are by far the most used formalism. This is consistent with the fact that it is a very popular process modeling formalism that can be output by several state-of-the-art discovery algorithms, such as the inductive miner [118].

Petri Nets is followed by weighted directed graphs, mostly as the output of the fuzzy miner algorithm [85]. This algorithm allows to mine models flexibly by determining the level of detail of the models.

A third noteworthy modeling language is timed parallel automata, a formalism introduced in [78] that is designed to be particularly expressive. Other formalisms, i.e., first-order logic, unweighted directed graphs, causal nets, process trees, DECLARE, and context-adaptive task models, are less widespread, only being used by at most two studies. In addition, only S22 uses a modeling formalism that incorporates the process execution context. Also note that S11 only derived an event log from the sensor log and did not mine a model; hence, no formalism is used.

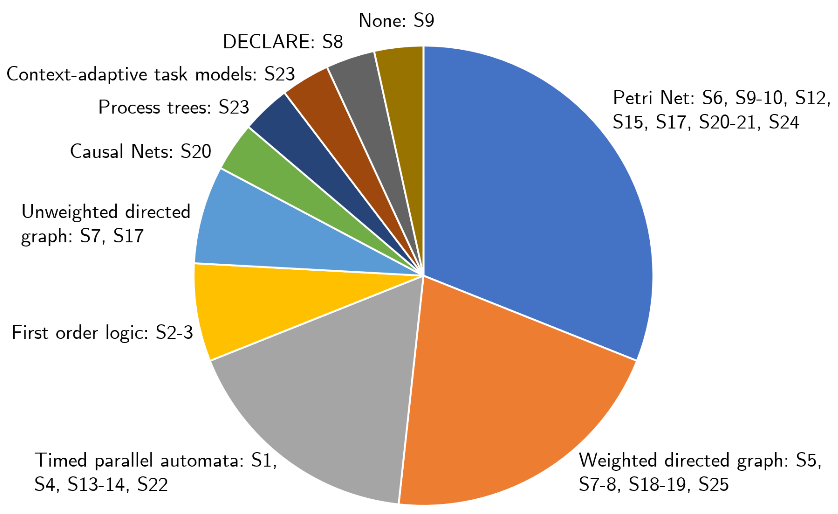


Figure 2.3. Number of publications per formalism.

2.1.3 Abstraction gap between sensor events and process events

This section gives an overview of the techniques that the primary studies use to convert sensor events into process events. Among them, S17, S18, S22, and S23 do not require any conversion steps because they already work with event logs instead of sensor logs. In particular, S22 and S23 make use of synthetic event logs produced by a simulator. All the other studies have validated their approaches with real-life datasets, as shown in Table 2.2. Ten studies (i.e., S1, S2, S4, S8, S11, S13, S14, S18, S21, and S25) have performed the validation step on datasets they generated themselves; all the other ones have applied their methodologies to state-of-the-art datasets, namely [44, 194, 148, 174, 176].

Two general approaches to making groups of sensor measurements that correspond to higher-level events can be identified from the literature: *(i)* classical window-based, time-based, or event-based segmentation, and *(ii)* more complex time-series analysis.

In order to translate raw sensor measurements into proper event labels, the most common method is to derive information from the sensor's location, as in S1, S4, S6, S13, S14, S15, S18, S21, and S25. For instance, if the triggered sensor is above the bed, then the task "sleeping" is derived. However, this method has its drawbacks, as acknowledged in S5: the information provided by motion sensors is not always detailed enough to derive tasks accurately. These ambiguities could be addressed by introducing other types of sensors in the environment (e.g., cameras), but that would make the approach more intrusive.

In S4, S14, and S25, no real conversion step is performed; the models directly show the successive locations users have been in. While this approach may seem naive, it can yield interesting results, as in S4, where it was employed to monitor clinical pathways, showing the flows of patients through different units of a hospital.

In S16, authors perform the conversion task by adapting an already existing algorithm [116] to automatically segment and assign human actions' labels (i.e., `MOVEMENT, AREA, or STAY`), combined with their relative *location* inside the smart environment (e.g., `STAY Kitchen_table`). A more detailed explanation and practical

ID	Log Segmentation Approach	Multi-user Approach	Routine Evolution Approach
S1	/	Sensor-based	Non-automated incremental learning
S2	Manual	/	Fully-automated incremental learning
S3	Manual	/	Fully-automated incremental learning
S4	/	Sensor-based	/
S5	Manual	Algorithm-based	/
S6	Time-based	/	/
S7	Time-based	/	/
S8	Time-based	/	/
S9	Manual	/	/
S10	/	/	/
S11	/	Sensor-based	/
S12	Time-based	/	/
S13	Time-based	Sensor-based	/
S14	Time-based	/	Non-automated incremental learning
S15	Manual	/	/
S16	Task-based	/	/
S17	Manual	/	/
S18	Time-based	Sensor-based	/
S19	Time-based	Algorithm-based	/
S20	Task-based	/	/
S21	Time-based	/	/
S22	Time-based	/	Semi-automated incremental learning
S23	Time-based	/	/
S24	Time-based	/	/
S25	Time-based	Sensor-based	/

Table 2.3. Summary of the approaches used in the studies to manage *(i)* log segmentation, *(ii)* multi-user environments, and *(iii)* routine evolution. The symbol '/' indicates that the related study did not provide any meaningful information about the related topic.

application of this approach can be found in Chapters 5 and 6.

Using a labeled dataset facilitates this conversion task. Studies S7, S12, S14, and S20 have used such labeling to manually deduce event names. However, this approach can be very time-consuming and error-prone, and labels often correspond to tasks at a higher level of abstraction with respect to atomic events.

2.1.4 Log segmentation into traces

PM techniques typically require a log to be segmented into traces with a case ID [158], a requirement that is often not met by sensor logs. S4 is an exception, as each trace in the log represents the path followed by one patient undertaking surgery at a hospital. The beginning and end of each instance of the process were therefore defined as the arrival and discharge of the patient; hence, no further segmentation was necessary. To account for the lack of segmentation in most cases, most of the included studies use a form of segmentation to obtain an event log made of distinct cases, as shown in Table 2.3. We assume that all studies, even those that do not state it explicitly, at least segment the sensor log in one trace per day to meet the requirement posed by PM techniques.

There are two types of segmentation applied in the studies: manual and automatic. In manual segmentation approaches, the user is requested to explicitly mark the beginning and/or the end of an ADL. Authors in S15, for example, use this approach to segment a day into ADLs. Based on the annotations added by the user, S15 adds artificial trace start and end events to the sensor log. For instance, when a user indicates that he or she is starting the *cooking* ADL, a start event is added to the sensor log.

Manual labeling is unfortunately time-consuming and error-prone. Alternatively, some approaches try to automatically segment the log, which appears to be more feasible in real-life scenarios.

Automatic segmentation is usually task-based or time-based. Authors in S9, for example, perform task-based segmentation to segment a log by creating one trace per day. Their approach uses the *sleeping* activity to determine when two consecutive days should be split. Examples of approaches using time for automated segmentation are instead the following:

- Using the time-based technique to split days using midnight as a cut-off point, such as in S6, S8, or S25. We want to highlight that, even though this kind of trivial approach may appear arbitrary, it is reasonable with respect to the employed logs (recording elderly activity).

- Segmenting each day into ADLs or visits by measuring the gap between two sensor events. When the gap is larger than a predefined threshold, the log is split into two traces, such as S13 or S23.

In addition, if the sensor log contains different versions of human ADLs, a clustering step is usually implemented, such as in S14 and S23.

2.1.5 Multi-user environments

In smart spaces (e.g., smart homes and smart factories), multiple entities can be present at the same time in the same environment. It has to be ensured that the analyzed events are all associated with the correct entity.

Among the included works, only eight studies support multiple users, i.e., S1, S4, S5, S11, S13, S18, S19, and S25. Table 2.3 shows that the included studies rely on two types of techniques to handle multi-user environments: *(i)* a sensor-based approach and *(ii)* an algorithm-based one.

The sensor-based approach involves the use of additional sensors and/or smart devices, whose sole purpose is to identify every user within the smart space. The most common application of this approach is to track an individual using their smartphone and Bluetooth Low Energy (BLE) sensors. Sensor events are associated with a certain inhabitant based on their proximity to the sensor itself. In S11, the authors propose an approach based on exploiting BLE beacons to discriminate between the different users interacting in the same smart space, then applying techniques from the area of business process mining. Here, inhabitants interact with beacon sensors by moving near them with their registered smartphones. Sensor data produced at a certain timestamp is then associated with that specific user. In S4 and S25, users wear wristbands that are recognized by motion sensors to identify the user activating them. In S13 and S18, BLE technology is used in the context of a shopping mall, allowing authors to derive a pathway for the customer that moves within it. In particular, authors in S13 use the collected data to discover and discriminate between male and female customer paths. In S1, inhabitants wear radio-frequency identification (RFID) bracelets that connect periodically with the server by using the Zigbee protocol, which computes an estimation of the distance [1].

The algorithm-based approach does not rely on additional sensors to recognize which user performed which task. Instead, it makes use of an algorithm that estimates an association between triggered sensors and users interacting within the same smart space. In S19, the *users' velocity* information is used by the algorithm to infer whether consecutive sensor events could be triggered by the same user. If the time span between two sensor events is considered too short for the user to have moved the distance between both sensor locations, it is assumed that a new user has entered the smart space. In S5, authors also manually derived a *distance policy*, representing logical distances between couples of sensors within the observed smart environment. Such additional information has been used to enrich the association algorithm between triggered sensors and users.

2.1.6 Routine evolution

As shown in Table 2.3, the included studies struggle to deal with routine evolution as only four of them manage to account for changing behavior, i.e., S1, S2, S3, S14, and S22. The included studies rely on three approaches: (i) non-automated, (ii) semi-automated, and (iii) fully-automated incremental learning.

The approach employed in S1 is denoted as non-automated. The authors use a process mining algorithm called PALIA [77] to infer models from location-based log samples. Such models can be filtered at specific time intervals, allowing us to see the model's evolution over time. A distance measure is then used to detect differences from the same model captured at different time intervals. The algorithm captures the differences in terms of transitions and nodes added or deleted. The weights of each type of difference can be customized in order to prioritize some differences over others. This allows them to identify and visualize changing behavior (e.g., an inhabitant who no longer has lunch in the common room), but this approach does not adapt the model based on new behavior. S14 proposes a related approach where traces are clustered to identify variants of the process. The variant executed each day is represented in a calendar view, which enables the visual detection of temporal patterns in the habits of the user (e.g., the user's habits differ on the weekend and during the week) and evolution in the behavior. If the behavior changed radically, generating a new process variant, a model of this variant could easily be discovered.

In S2 and S3, the authors implemented a fully automated approach to handle the routine evolution. Firstly, their algorithm divides the dataset into daily training cases. Then, it takes a new training case and uses it to update an existing workflow model, if any, or to build a new model from scratch, if that is the first case considered. They adopt a process mining perspective to incrementally learn models of routines on a daily basis.

The most recent, among the analyzed studies, follows a semi-automated approach. This technique is similar to the fully automated one, but it also relies on user input to assist the incremental learning process and update the model. In S22, they employ a combination of two techniques called *Cortado* and *MAtE*. Cortado is used to identify traces present in the captured sensor log but absent in the existing model. These missing traces are then presented to the user using MAtE, allowing the user to select the traces they would like to have supported by the model.

2.2 Discussion

This section discusses the datasets used by the identified solutions, the investigated challenges, and explains how validity threats were mitigated in the survey.

2.2.1 Used datasets

As shown in Section 2.1.1, the most common kind of sensors used to collect data in the studied solutions are motion ones. Motion sensors provide a low-cost, low-power, small, and lightweight alternative in many applications if exploited in the right way [144]. For instance, in comparison with cameras, another sensing system very present in smart spaces, motion sensors consume less power for sensing a large area, and they pose fewer privacy concerns. Cameras require often sophisticated image processing algorithms and expensive lenses to improve their detection, while typically a motion sensor consumes approximately $3\mu W$ for movement detection. The approaches shown in Chapters 5 and 6 use this type of sensor.

Concerning the labeling, being able to recognize when a certain activity of daily living (ADL) started and ended can be used, for instance, to trigger automation rules or to detect potentially harmful deviations from the usual behavior. Unfortunately, such solutions require logs to be labeled with markers denoting the onset and the end of all (or at least of a consistent subset) ADL occurrences. Such labeling tasks can be performed manually or they can be derived with some automatic technique. Manual labeling of logs is perceived by the user as a boring and laborious job, which results in imprecise labeling, possibly tampering with the performance of algorithms at runtime. On the other hand, automatic labeling techniques suffer from several limitations when applied to real-world scenarios, e.g., in many cases, automatic techniques are only used to complement manual segmentation and are not intended to segment the full log [46]. The approaches shown in Chapters 5 and 6 are based on automatic techniques to derive activity and habit models.

Regarding the number of different ADLs included in a labeled dataset, excluding the extreme cases with 5 and 37 labeled ADLs, most of the logs used in these studies employ an average of 10–11 labels to describe human behavior in smart environments. For instance, the *aruba* dataset from the CASAS project [44] has the following labels: *Bed to toilet, Eating, Enter home, Leave home, Housekeeping, Meal preparation, Relax, Sleeping, Wash dishes*, and *Work*.

2.2.2 Modelling formalisms

As discussed in Section 2.1.2, papers applying PM to smart spaces data must explicitly or implicitly choose a formalism to represent human processes.

Interestingly, while it is suggested in [124] that human routines are rather unstructured and unpredictable, the most used formalism in the reviewed studies is Petri Nets, an imperative modeling language. This may simply be because Petri Nets are one of the most widely used languages in PM, which allows process checking, simulation, and enactment.

A certain number of studies opted for alternative formalisms that perform well in practical situations and with results that are more readable and easy to understand [201], e.g., weighted directed graphs. This enables the discovery of clearer and potentially better-fitting models, though less precise and actionable. The assumption of having a precise model that perfectly fits the log is limiting. For instance, "spaghetti models" are not necessarily incorrect because they precisely describe every structural detail found in the log. However, a more high-level and less precise solution, which is able to abstract from details, would thus be preferable. In general, the assumption of a perfect solution is not well-suited for real-life applications [85]. A solution to make models more actionable is to implement prediction techniques, as in S10.

It is also remarkable that only one of the studies, S8, mined declarative models, a widespread flexible paradigm that could be able to cope with the variability of human behavior. This may be explained by the fact that declarative models are usually harder to understand than imperative models, making it more complex for users to interact with the smart space system.

Finally, another important aspect in smart spaces is context-awareness or context-dependency: the process model should be context-aware to adapt to the changes in the environment [18]. This is surprisingly still neglected in current research about PM applied to smart spaces. Only S22 supports the modeling of context-adaptive routines by using context-adaptive task models and process trees.

2.2.3 Abstraction gap between sensor events and process events

The abstraction gap has been recognized as one of the main challenges in PM applied to smart space data [208]. The solutions proposed in the literature are dataset- and/or sensor-specific. In most cases, only infrared sensor data (i.e., motion data) is available, witnessing the human performing actions in specific areas of the house. This also makes the techniques proposed very sensitive to the distribution of sensors across the environment. In addition, the scarce availability of datasets makes it difficult to evaluate the proposed approaches across multiple scenarios. In most cases, datasets from the CASAS project[1] are used. This does not provide sufficient heterogeneity to ensure a reliable evaluation.

Finally, input from the broader PM literature could help address this issue. More specifically, generic event abstraction techniques used in PM could also be used to abstract sensor events into process events (see [207]). This being said, in a benchmark study, authors in [192] suggest that typical unsupervised event

[1] See `http://casas.wsu.edu/datasets/`.

abstraction techniques do not always allow discovering more meaningful higher-level models. As shown by this literature review, the vast majority of event abstraction techniques in the case of smart spaces are not fully unsupervised, as even when no labeling is provided, some kind of domain knowledge is employed in the abstraction process. Moreover, IoT PM methodologies also propose techniques to extract an event log from sensor data, such as, e.g., in S20; a deeper dive in this literature could identify relevant abstraction techniques for smart spaces.

2.2.4 Log segmentation into traces

The proposed approaches for segmentation are usually naive (e.g., automatic daily-based segmentation) or rely on extensive output from the user (i.e., manual task-based segmentation). From this point of view, the open research challenge is to perform segmentation by using process semantics and the context [124]. An initial proposal has been given in S24, where process model quality measures are used to iteratively segment the log.

In addition to this, segmentation is only a part of the problem, as traces must be clustered in order to produce event logs that only contain instances of the same ADL, which is a prerequisite for PM [190]. This is analogous to the general issue of case ID definition in PM, i.e., pinpointing what an instance of the process is.

2.2.5 Multi-user environments

As described in Section 2.1.5, two techniques are mainly employed: a sensor-based approach and an algorithm-based approach.

The first approach is not perfect, e.g., there is the possibility that two or more different users get near the same beacon [163]. In this case, as proposed in S11, a solution to this problem could be to duplicate and pair the interested portion of log records with all the interested users. However, this approach does have its perks, namely the availability of contextual information about the user that is performing a task [60]. This contextual information proves to be valuable as it allows for the creation of more detailed process models (e.g., creating a different process model for men and women, supporting gender-based analysis of behavior, like in S13). The main drawbacks of using a sensor-based approach include: *(i)* the lack of privacy (i.e., location data of each user is recorded 24/7) [38] and *(ii)* the impracticality of always needing to have your smartphone or bracelet with you.

In the second approach, the algorithm-based one, the lack of additional sensors makes this approach easier to implement as opposed to the sensor-based approach. The algorithm-based approach does not require the user to wear bracelets or smartphones, making it less intrusive and less reliant on battery-powered devices. However, it does have one main drawback: it can be considered less accurate because it is less resilient to noise [147], e.g., a pet might trick the algorithm into thinking a new user has entered the smart space.

2.2.6 Routine evolution

As described in Section 2.1.6, the included studies rely on three approaches to handling routine evolution. The non-automated approach, as opposed to the other two techniques, does not adapt the model based on the behavior's changes in the inhabitant. As a result, this technique is less suited for applications where the model needs to account for changing routines. However, it is useful in the detection of behavioral anomalies and could therefore be recommended in specific environments, e.g., nursing homes, to implement elderly fall detection systems [76].

The other two kinds of approaches represent two sides of the same coin. The semi-automated technique follows the user-in-the-loop paradigm, i.e., the final user assists the learning process and validates learned models. Such an approach is useful since it prevents undesired behavior from being included in the model (e.g., the user was sick and spent a whole day in bed). However, manual validation can be perceived by the user as an annoying and laborious job and could lead to wrong choices [90]. Whereas fully-automatic techniques reduce the final user's effort by learning models in an autonomous way, even reaching excellent levels of accuracy, it is not possible to exclude that learned models include some undesired behavior.

2.2.7 Threats to validity

When conducting a literature review, threats to validity should be considered and mitigated [12]. The following actions were taken to mitigate:

- *study selection validity* risks: *(i)* search criteria were selected and modified based on pilot searches; *(ii)* forward and backward snowballing were used to find relevant literature; *(iii)* articles were retrieved from the most well-known digital libraries; *(iv)* duplicate articles were managed consistently.
- *data validity* threats: *(i)* a data extraction form was used; *(ii)* variables from the extraction form were mapped to research questions; *(iii)* results were compared to existing studies.
- *research validity* risks: *(i)* the review protocol used in this paper was defined in detail before it was executed; *(ii)* the related work was studied beforehand to ensure the research relevance.

Chapter 3

Generating smart home data

Scientific contributions in the field of smart spaces need to be validated against datasets, for example, sensor logs acquired from smart environments. In order to acquire these datasets, expensive facilities are needed, including sensors, actuators, and an acquisition infrastructure. In addition, frequently employed smart home hubs (e.g., voice assistants like Amazon Echo Dot) do not allow access to raw data.

Even though several freely accessible datasets are available, each of them features a very specific set of sensors, which can limit the introduction of novel approaches that could benefit particular types of sensors and deployment layouts. Additionally, acquiring a dataset requires a considerable human effort for labeling purposes, thus further limiting the creation of new and general ones. Also, labeling is an error-prone activity, which makes the quality of the available datasets unclear. As a consequence, the vast majority of approaches available in the literature are evaluated against datasets gathered in university labs (i.e., datasets that are not always general and/or whose quality cannot be taken for granted).

For this reason, smart environments are one of the many disciplines where we are witnessing the *replication crisis* (or *reproducibility crisis*), i.e., an ongoing methodological crisis in which it has been found that many scientific studies are difficult or impossible to replicate or reproduce [11][74].

Even if the crisis is more evident in the social sciences, the inability to replicate the studies of others has important impacts in the smart environment area. In order to advance in this area, researchers need open datasets, as without them, developed techniques can neither be compared nor validated in a reproducible way, which hinders innovation in the area. Without comparable and proven techniques, practitioners, industries, and users will not be confident in large investments and deployments, thus, in turn, not increasing the number of testbeds, living labs, and datasets: a dramatic spiral that hinders the whole research community.

In addition to the above-mentioned aspects, a recent research trend [95] focuses on the possibility of analyzing smart environments in terms of the ongoing processes by relying on the Internet of Things. A business process is a collection of related, structured activities or tasks performed by people or equipment with a specific goal. A Cyber-Physical System (CPS) [115] integrates computation with physical processes, i.e., business processes, where the vast majority of tasks happen in the physical world. As such, modern smart spaces can be considered examples of CPSs, where human routines are the processes of interest. In CPS, embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa. Physical processes in smart spaces are called Activities of Daily Living (ADLs). ADLs are usually employed as high-level labels without considering the inner details. In order to fully support CPSs, datasets must be labeled not only for higher-level processes (e.g., human activities of daily living) but also for fine-grained actions (e.g., opening a door).

In order to tackle the above challenges, we present a model-based simulator able to generate synthetic datasets using the eXtensible Event Stream (XES) international standard format and containing all the necessary labels. The simulator is based on human activity models and is able to emulate the characteristics of the vast majority of real datasets while granting trustworthy evaluation results.

Such datasets can be employed to propose, for instance, challenges organized in conferences and workshops, to produce fair comparisons between different approaches, and/or to replicate the features of already available datasets and extend them.

The rationale behind this effort is that while the evaluation of algorithms in smart spaces through real datasets still remains important, the possibility to easily create custom datasets by fine-tuning a virtual space and available sensors could greatly improve the evaluation of proposed algorithms. In particular, thanks to the proposed simulator, it will be possible to evaluate the performance of algorithms depending on the availability and distribution of sensors and actuators. Also, this simulator could be employed as a design tool for smart spaces targeted at improving the performance of already available algorithms.

In order to assess the simulator, we evaluate the quality of the datasets that it can generate in terms of how well they can emulate reality in comparison with two real-scenario state-of-the-art datasets.

3.1 Behavior pattern models

Human processes in smart spaces may require specific formalisms. In particular, authors in [168, 169] introduce a specific BPM formalism, extending the Hierarchical Task Analysis (HTA) meta-model [14], that we will denote as *context-adaptive behavior pattern* modeling formalism or simply behavior pattern modeling formalism in the rest of the chapter, to deal with human processes in smart homes.

Figure 3.1 shows an example of a behavior pattern taken from [168]. As it can easily be seen, the task "Waking up" is hierarchically defined as the sequence of tasks "BathroomHeating.turnOn", "Adapt Bedroom", and "ApplianceController.makeCoffee", where "Adapt Bedroom" is a composite task that is in its turn refined into other tasks and so on. Tasks at the same level of the hierarchy can be instantiated according to different constraints. For example, the symbol $A|||B$ denotes concurrent execution of tasks A and B; the symbol $A \gg B$ denotes that task B is performed immediately after the end of task A; and the symbol $A[] \gg B$, in addition, denotes that the task A produces an output used by the task B. In this work, we have chosen the behavior pattern modeling formalism introduced in [168, 169] as a means to define human routines to be simulated. This formalism

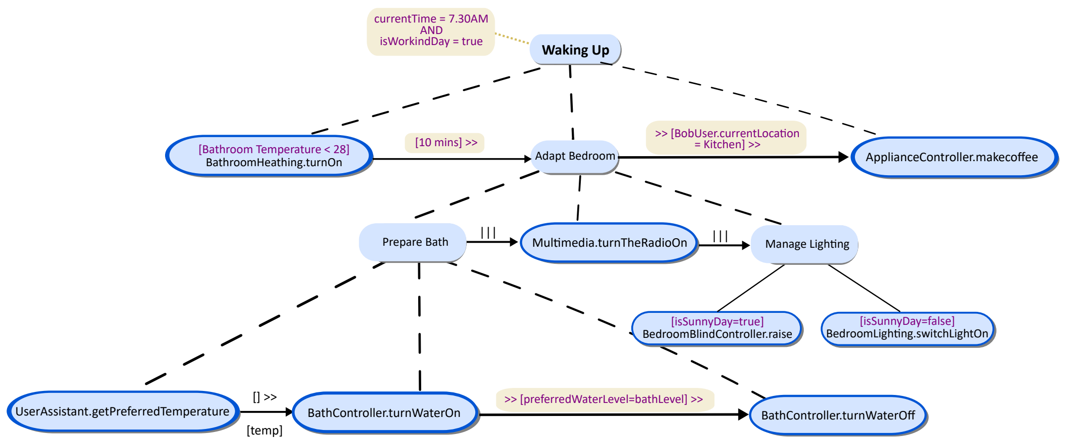

currentTime = 7.30AM
AND
isWorkindDay = true
Waking Up
[Bathroom Temperature < 28]
BathroomHeathing.turnOn
[10 mins] >>
Adapt Bedroom
>> [BobUser.currentLocation
= Kitchen] >>
ApplianceController.makecoffee
Prepare Bath
|||
Multimedia.turnTheRadioOn
|||
Manage Lighting
[isSunnyDay=true]
BedroomBlindController.raise
[isSunnyDay=false]
BedroomLighting.switchLightOn
UserAssistant.getPreferredTemperature
[] >>
[temp]
BathController.turnWaterOn
>> [preferredWaterLevel=bathLevel] >>
BathController.turnWaterOff

Figure 3.1. An example of behavior pattern taken from [168].

extends HTA models with:

- A *context situation*, which is associated with a behavior pattern. It indicates the set of context conditions that must be satisfied in order to execute the full set of coordinated services. It is depicted by a note associated with the root task of the hierarchy that defines the behavior pattern. For instance, Figure 3.1 presents a context-adaptive version of the "WakingUp" behavior pattern. Its context situation indicates that the behavior pattern should be executed every working day at 7:30 a.m.

- Task *context preconditions*, which can be associated with any task of a hierarchy except the root task. They indicate that the task (and therefore also its subtasks) must be executed if and only if the task precondition is satisfied. They are defined over contextual variables and are depicted between brackets just before the task name. For instance, the system must turn the bathroom heating on only if the bathroom temperature is $< 18^\circ C$.

- *Context-dependent constraints*, which can be defined to coordinate the execution of tasks that have been obtained from the temporal refinement of the same parent task. As explained earlier, constraints of this type are depicted by means of arrows between subtasks. These constraints are the following:

  - $T1 \gg [c] \gg T2$: after executing T1, T2 is performed only when the condition c is fulfilled. For instance, the system could decide to prepare coffee after adapting the room, but only when the inhabitant named Bob is in the kitchen (BobUser.currentLocation.name=Kitchen). Similarly, the task of turning the bath water off is performed after turning it on, but only when the bath water level has reached the level preferred by the inhabitant.
  - $T1\ t \gg T2$: after executing T1, T2 is performed when the period of time t has elapsed. For instance, ten minutes after turning on the bathroom heating, the system must adapt the bedroom.

In this work, we used HTA to represent the models that are fed as input to the proposed simulation tool. With respect to other modeling formalisms in the BPM area, HTA are more suitable to represent human activities and habits, as they easily make possible to model conditions, which are fundamental triggers for human actions.

Noteworthy, the employment of HTA as a modeling tool for generation purposes does not require using the same formalisms for analytics and process mining. In particular, HTA is well suited for prediction and conformance checking but not for discovery and enhancement [169].

3.2 XES - eXtensible Event Stream

Event logs are usually made available by the BPM and Process Mining community in the form of eXtensible Event Stream (XES) files [200], an IEEE standard[1] based on XML.

XES event logs are divided into traces, each representing the execution log of a specific instance of a process. Each trace contains a set of events ordered by timestamp. An identification number is assigned to each trace and each event. Attributes can be associated with logs, traces, and events, but XES does not prescribe a fixed set of mandatory attributes for each element (log, trace, and event); an event can have any number of attributes. However, to provide semantics for such attributes, the log refers to so-called extensions.

An XES log also defines an arbitrary number of classifiers. Each classifier is specified by a list of attributes. Any two events that have identical values with respect to these attributes are considered to be equal for that classifier. These attributes should be mandatory event attributes. For instance, if a classifier is specified by both a name and a resource attribute, then two events are mapped onto the same class if their names and resource attributes coincide. In this case, by equal events, we mean actions that compose an activity that has the same name and the same life cycle. The life cycle's attribute defines whether the event refers to the start or end of the related action (with the values *start* and *complete*, respectively). In Figure 3.2, an example of an XES log has been provided. In particular, the header provides information about the XES formalism and the classifiers used. The actions are defined as a sequence of event tags (e.g., wash hands) and grouped into traces. Each event provides information about the name of the action, when the action occurred, and its life cycle through its attributes. In Section 3.3.2, it is shown how a behavior pattern (like the one shown in Figure 3.1) can be translated using this XES notation.

Our simulator produces logs in the XES format. The motivation for this choice is twofold. In the first place, as part of the growing trend aiming at applying BPM and process mining, we want to make datasets easily analyzable by using tools from the community, which take as input XES files. In second place, XES is a standard for storing streaming data, which makes it particularly suitable for data coming from smart space sensors. However, an alternative format fully compatible with this simulator will be discussed in Chapter 4.

[1]cf. `https://xes-standard.org/`

```xml
<log xesversion="1.0" xesfeatures="nested-attributes" openxesversion="1.0RC7">
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <classifier name="Event Name" keys="concept:name"/>
    <classifier name="(Event Name AND Lifecycle transition)" keys="concept:name lifecycle:transition"/>
    <string key="concept:name" value="XES Event Log"/>
    <trace>
        <string key="concept:name" value="0"/>
        <event>
            <string key="concept:instance" value="0"/>
            <string key="concept:name" value="Wash hands"/>
            <string key="lifecycle:transition" value="start"/>
            <date key="time:timestamp" value="2010-11-04T05:40:44"/>
        </event>
        <event>
            <string key="concept:instance" value="0"/>
            <string key="concept:name" value="Wash hands"/>
            <string key="lifecycle:transition" value="complete"/>
            <date key="time:timestamp" value="2010-11-04T05:42:26"/>
        </event>
        <!--
        list of event tags
        -->
    </trace>
    <!--
    list of trace tags
    -->
</log>
```

Figure 3.2. An example of *event log* represented by using the XES formalism. This file contains a set of traces (defined through the `trace` tag), each one composed of a sequence of events (defined through the `event` tag).

3.3 Design and realization of the simulator

The simulator proposed in this chapter follows the framework described in [107], in which the authors propose a model to simulate daily activities. Section 3.3.1 introduces the main features of the original framework. We enriched the simulator with *(i)* the support for model-based simulation using context-adaptive behavior patterns, which allow to model branches and conditions that are typical of human activities (see Section 3.3.2), *(ii)* the possibility to simulate multiple users (see Section 3.3.3), and *(iii)* the possibility to export data as XES (see Section 3.3.4).

3.3.1 Original simulation framework

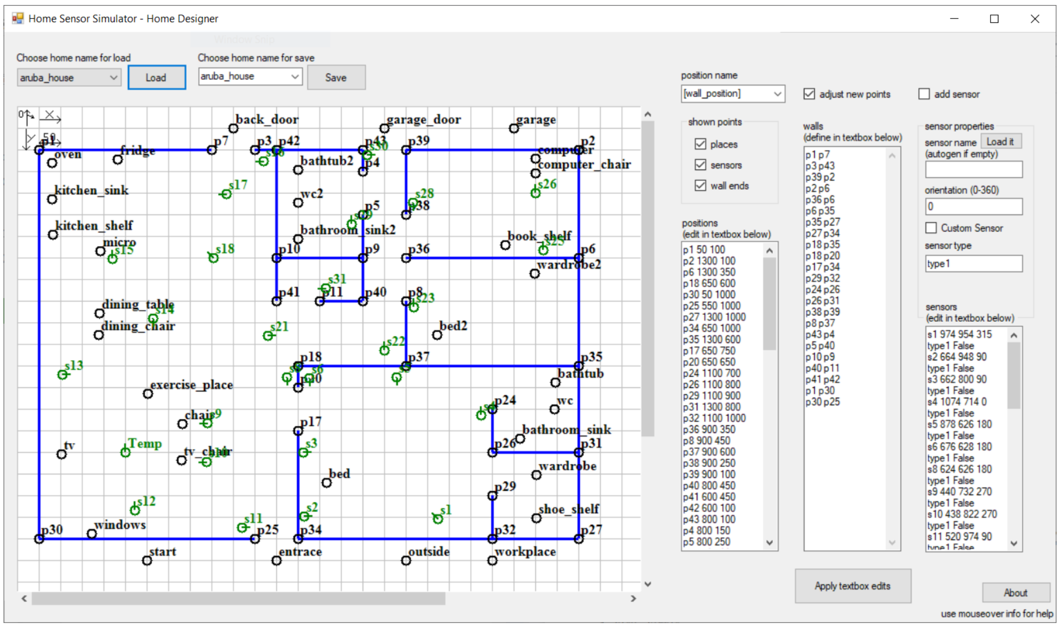
The original framework is based on three main components:

1. The *house model* models the environment in which the simulated inhabitant(s) live (in our case, a smart house). With such a model, it is possible to set the layout of the house. Walls, as in a real house, delimit areas of the virtual environment, thus limiting the possible movement trajectories of simulated human inhabitants. In Figure 3.3, a screenshot of the simulator's component used to design the layout of a house model has been provided. The position of walls is considered fixed, and the positions of walls and doors are therefore considered constant parameters. Next to this, variable parameters can also be defined (e.g., temperature, humidity, and whether the television is on or off). In addition, several *positions* can be defined within the already defined layout. Positions represent all the locations with which an inhabitant may interact during the day (e.g., the bed, the TV, the sink, the bookshelf). Finally, the positions of sensors can also be defined through this model. They are represented as a triple (x, y, α) where x and y are coordinates within the 2-dimensional layout of the house and α represents the orientation of the sensor in degrees, i.e., what the sensor is "looking at".

2. The *human model* is used to simulate different inhabitants with different behavior profiles. It consists of a set of constant parameters describing the character of each inhabitant, such as his or her speed or frequency of feeling different needs (e.g., tiredness, thirstiness). For instance, an inhabitant could need eight hours of sleep, while another would need much less. One could drink a lot of water every day; the other consumes less. These parameters, together with environmental parameters from the house model, describe the overall state of the current situation. This information is used to control the next event to be simulated. A detailed description of how events are handled by the simulator will be given later in this section.

3. Finally, the *acting model* provides the mathematical definition of the simulated events (described afterwards), and it is used to simulate the monitoring of an inhabitant in his or her smart environment. Simulated motion sensors log if they detect the inhabitant. The distance from the sensor and the angle of the inhabitant relative to the sensor's direction are also logged. The non-motion sensors (e.g., temperature, humidity, and light sensors) are implemented as environmental variables, e.g., the value of the temperature. Actions and other custom functions can modify them. For instance, by opening the tap, the variable related to water consumption will be increased by this action.

These three models together are used to simulate human daily activities by considering the environmental context in which the inhabitant(s) move(s) and interact(s), each of them with their behavioral characteristics and needs.



Choose home name for load: aruba_house

Choose home name for save: aruba_house

position name: [wall_position]

☑ adjust new points

☐ add sensor

shown points: ☑ places ☑ sensors ☑ wall ends

positions (edit in textbox below)

```
p1 50 100
p2 1300 100
p6 1300 350
p18 650 600
p30 50 1000
p25 550 1000
p27 1300 1000
p34 650 1000
p35 1300 600
p17 650 750
p20 650 650
p24 1100 700
p26 1100 800
p29 1100 900
p31 1300 800
p32 1100 1000
p36 900 350
p8 900 450
p37 900 600
p38 900 250
p39 900 100
p40 800 450
p41 600 450
p42 600 100
p43 800 100
p4 800 150
p5 800 250
```

walls (define in textbox below)

```
p1 p7
p3 p43
p39 p2
p2 p6
p36 p6
p6 p35
p35 p27
p27 p34
p18 p35
p18 p20
p17 p34
p29 p32
p24 p26
p26 p31
p38 p39
p8 p37
p43 p4
p5 p40
p10 p9
p40 p11
p41 p42
p1 p30
p30 p25
```

sensor properties

sensor name (autogen if empty):

orientation (0-360): 0

☐ Custom Sensor

sensor type: type1

sensors (edit in textbox below)

```
s1 974 954 315
type1 False
s2 664 948 90
type1 False
s3 662 800 90
type1 False
s4 1074 714 0
type1 False
s5 878 626 180
type1 False
s6 676 628 180
type1 False
s8 624 626 180
type1 False
s9 440 732 270
type1 False
s10 438 822 270
type1 False
s11 520 974 90
type1 False
```

Figure 3.3. A possible smart home layout that can be designed through the Home Designer tool. In particular, here we have recreated the smart environment used to extract the state-of-the-art testbed, called *aruba*, from the CASAS project (see Section 3.4.2). The original 2-dimensional map is provided by the authors of the testbed themselves.

The simulation is performed on the basis of a *virtual clock*, thus allowing to generate data for long periods of time with short computation time. At each simulation step (i.e., at each tick of the virtual clock), the values assigned to the parameters (defined in the three models already discussed) are reevaluated according to custom mathematical formulas. For instance, the tiredness of an inhabitant could be expressed as a formula that increases the related value (from the human model) at every simulation step. Such value could be increased more or less quickly, depending on how the formula is defined by the final user.

The simulator supports two different kinds of events:

- `main events`, i.e., what we have defined as human *activities* in Section 1. They represent a higher level of abstraction with respect to bottom events (see below). A main event contains a sequence of actions (i.e., bottom events) to be executed. For instance, the main event describing the activity `cook_and_eat` (see Figure 3.4) is composed by the following list of actions: *go_fridge, get_ingredients_from_fridge, go_kitchen_shelf,* and so on. This sequence of actions, in their entirety, composes the main event itself.

- `bottom events`, i.e., what we have defined as *actions* in Section 1. They represent a lower level of abstraction with respect to the main events. They are executed by the inhabitant in the smart environment, and they influence human and environmental variables during their execution. There is a sub-category of bottom events called *movement events*. Movement events are labeled with the prefix *go_*, and they trigger the movement of a person to a specific goal position from the list of *positions* already defined in the house model.

```
1 cook_and_eat MainEvent
2 Priority $min(@(hunger)*(0.5+@(c_f))+$attime(13,2)*20-25+@(warm_food_need)
      ,100)
3 Interrupt 80
4 Flow Simple
5 go_fridge get_ingredients_from_fridge go_kitchen_shelf
     get_ingredients_from_shelf go_oven use_oven go_dining_table
     eat_warm_meal go_oven pack_food go_fridge put_meal_to_fridge
     plate_to_sink
```

Figure 3.4. Example of main event describing the activity `cook_and_eat`. This activity is composed of a list of actions, also called bottom events. The priority of the event (see row 2) is provided by a formula taking into account the fact of being in a particular moment of the day (through the `$attime` built-in formula), the necessities (through the human variables `@(warm_food_need)` and `@(hunger)`), and the preferences of the specific inhabitant (through the human constant `@(c_f)`). The interrupt value (see row 3) defines the minimum priority another main event needs to interrupt this event. Finally, at row 5, the sequence of bottom events to be generated (i.e., actions to be executed by the human inhabitant to perform the activity) is reported. Among these bottom events, we have movement actions (e.g., `go_fridge` that are automatically translated in a movement of the inhabitant to the position named `fridge`) and other actions, such as `use_oven` that are further specified.

Each main event has a *priority* parameter. This value is used by the tool to control the sequence of main events (i.e., activities) that the inhabitant will perform through the simulation by determining which main event will be executed next. Like in real life, a person will drink if his/her priority in drinking is the highest (i.e., the person is thirsty), and he/she will take a rest when is tired and there is nothing more important to do.

Furthermore, all main events have an *interrupt* parameter. Main events can be interrupted by other main events. The original event will only resume when the interrupting one is concluded, if no other interruption happens. As happens in real life, a person could interrupt lunch to answer an urgent call. Only when the call is finished will he or she resume eating.

Then, at every simulation step the main event with the highest value of priority (among all the main events defined in the acting model) is executed. Only a main event with higher priority can interrupt it. In this case, the interrupted main event is paused and pushed in a queue. It will be restored only after that the interrupting main event is concluded. A detailed description of how it works can be found in [107]. Figure 3.4 and Figure 3.5 respectively show an example of a main event describing the activity `cook_and_eat`, and a bottom event describing the action `use_oven`.

3.3.2 Model-based simulation

The *main event* construct in the original framework only allows to model activities that are unconstrained flows of events [107]. We extended the simulator to support behavior patterns for describing and modeling more complex activities [168, 169].

```
use_oven BottomEvent
Duration 40*60
DurationDeviation 10*60
Results
cooked_food @(cooked_food)+$atstart()*(0)+$atend()*(2)+@(step)*(0)
power_use @(power_use)+$atstart()*(800)+$atend()*(-800)+@(step)*(0)
Home_Aired @(Home_Aired)-(3.0)*$atend()
```

Figure 3.5. Example of bottom event describing the action `use_oven`. In particular, the duration of each action (in seconds) is randomly chosen in the interval `Duration` $\pm$ `DurationDeviation` (in the figure 40 ± 10 minutes). During the execution of an action (always following the virtual clock), the value of the variable is continuously updated. For example, the `@(power_use)` variable is incremented by 800 at the beginning of the action and decreased by 800 at the end.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pros:TaskModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
↪  xmlns:pros="http://org/pros/BehaviourPatternModel.ecore">
    <behaviourPatterns ID="housekeeping" name="housekeeping"
    ↪  enabled="true" priority="30" interrupt="50">
            <!--
            list of actions (i.e., bottom events)
            -->
        <contextSituation condition="Home_Presence">
            <leftPart instanceName="Home_Presence"
            ↪  propertyName="value"/>
            <rightPart value="0"/>
        </contextSituation>
    </behaviourPatterns>

</pros:TaskModel>
```

Figure 3.6. Example of `task model` translated by using the XML notation. The task model includes a `behaviourPatterns` tag, which describes and models an activity (i.e., a main event), in this case `housekeeping`. It is composed of a list of actions (i.e., bottom events) defined through the `refinements` tag (see Figure 3.7). In this case, there is also a `contextSituation` tag: the entire activity can start only if the inhabitant is at home (i.e., the Home_Presence parameter is set to 0).

The behavior pattern introduces an XML notation (as described in [168]), which for our purposes has been extended to support some features of the simulator, such as priorities and action durations. The `behaviourPatterns` tag (inside the `taskModel` tag) models an activity. The tag has been enriched with two additional properties: `priority` and `interrupt` (see example in Figure 3.6). With these parameters, the simulator that parses the XML file can establish when the corresponding activity

```xml
<refinements order="1" type="Temporal">
    <refinementTo xsi:type="pros:CompositeTask" ID="housekeeping1"
    ↪  name="housekeeping1"
    ↪  refinedFrom="//@behaviourPatterns.0/@refinements.1">
        <refinements order="0" type="BottomEvent">
            <refinementTo xsi:type="pros:SystemTask" ID="clean_dust"
            ↪  name="clean_dust"
            ↪  refinedFrom="//@behaviourPatterns.0/@refinements.0"
            ↪  serviceName="clean_dust"
            ↪  serviceMethodName="clean_dust">
                <contextPrecondition condition="High Temp"
                ↪  operator="lower" explanation="High Temp">
                    <leftPart instanceName="Temp"
                    ↪  propertyName="value"/>
                    <rightPart value="20"/>
                </contextPrecondition>
            </refinementTo>
        </refinements>
        <refinements order="1" type="BottomEvent">
            <refinementTo xsi:type="pros:SystemTask" ID="wash_floor"
            ↪  name="wash_floor"
            ↪  refinedFrom="//@behaviourPatterns.0/@refinements.0"
            ↪  serviceName="wash_floor"
            ↪  serviceMethodName="wash_floor">
                <contextPrecondition condition="High Temp"
                ↪  operator="greater" explanation="High Temp">
                    <leftPart instanceName="Temp"
                    ↪  propertyName="value"/>
                    <rightPart value="20"/>
                </contextPrecondition>
            </refinementTo>
        </refinements>
    </refinementTo>
</refinements>
```

Figure 3.7. An example of `CompositeTask` represented using the XML notation. In this case, the task contains an exclusive branch: if the temperature registers a warm day, then the `BottomEvent` `wash_floor` will be executed. Otherwise, the `BottomEvent` `clean_dust` will be performed.

can start and, eventually, when it can be interrupted by another activity with the highest priority.

Within the `behaviourPatterns` tag, a series of `refinements` tags are defined, which represent the set of actions of which the activity is composed. Each refinement can be enriched with a `contextPrecondition`, expressed as a boolean formula over simulator variables, which determine whether the related action is executed or not.

For instance, if the temperature outside is greater than 20 degrees, then perform the action `wash_the_floor` (see example in Figure 3.7).

There exist three types of `refinements`, identified by the `type` property:

- `BottomEvent` maps to bottom events in the original simulator, thus modeling an action performed by the smart space inhabitant that directly involves an object or a service inside the smart environment (e.g., `clean_dust` in Figure 3.7).
- `moveevent` refinements model a movement action of the inhabitant to reach a specific point inside the smart house (e.g., `go_computer`);
- `CompositeTask` refinements are composed of a set of simple refinements (bottom events and/or movement events) and/or other nested composite tasks. This particular kind of refinement allows for the modeling of exclusive branches inside the action flow. Depending on the current environmental context, the simulator can make decisions about the next refinement to follow.

The `refinementsTo` tag contains the name of the refinement declared in the `refinements` father tag and is used as an identifier within the simulation. Finally, each `refinementTo` tag may contain a `temporalRelationship` tag that identifies specific order restrictions among the actions of the activity modeled in task-model language. For instance, as happens in the example provided in Figure 3.7, the action `wash_floor` is executed only if the value related to the temperature in the environment is greater than 20. Otherwise, the action `clean_dust` is executed.

Each behavior pattern optionally includes a `contextSituation` tag containing a condition in the form of a logical expression, which must be checked to verify whether the modeled activity can be executed or not, e.g., if it is a working day, then start the *work* activity (see example in Figure 3.6). This tag differs from the previously introduced `contextPrecondition`, which refers to the conditions required to start the entire activity and not a single composing action.

3.3.3 Multi-inhabitant simulation

The simulator described in [107] allows to define different behavior profiles through the human model. However, it is restricted to simulating the daily activity of a single human inhabitant. In order to make the simulator capable of replicating real-life scenarios, we added the possibility to instantiate more than one inhabitant in the simulated environment, each with its own human model, i.e., its own human behavioral profile.

In Section 3.3.1, we discussed how *priority* and *interrupt* parameters are handled during the simulation. That process allows for the management of the sequence of simulated activities performed by a single inhabitant.

To add the possibility to simulate multiple inhabitants within the same simulation, that process is replicated n times, where n is the number of virtual inhabitants. In this case, the simulator relies on common *house* and *acting* models (i.e., the same environment and the same set of available events) and several *human* models in order to differentiate the behavior of each simulated inhabitant.

Each instance of this process (i.e., each simulated inhabitant) writes on a common log and is labeled with a unique identifier. As a result, the simulator produces a single event log containing the interleaved series of activities performed by the n users, each with its own identifier.

3.3.4 Data export in XES format

Data generated in the context of the simulation is stored using the XES format described in Section 3.2. In particular, the simulator produces as an output three different logs: a segmented event log, an unsegmented event log, and a sensor log.

1. The *segmented event log* contains a different trace for each execution of an activity. Each trace includes two attributes, `traceId` and `name` denoting respectively the identifier of the trace and the name of the activity performed in that trace. The presence of the `name` attribute represents an important difference with respect to classical process mining event logs, where traces represent different executions of the very same process. Instead, we have different executions of different activities (the equivalent of processes in process mining). Traces contain events representing the execution of actions by virtual inhabitants. For each action, we can have one event in the case of immediate actions or two events in the case of actions with a duration. Each event included in a trace consists of *(i)* an `eventId` globally identifying the event, *(ii)* a `concept:name` attribute indicating the name of the action, *(iii)* a `time:timestamp` attribute indicating the occurrence time, in the virtual clock, of the event, *(iv)* a `lifecycle:transition` attribute that can assume the values `start` and `complete` denoting respectively the beginning of the action and the end (immediate events only have the `complete` event), and *(v)* a `org:resource` attribute indicating the identifier of the virtual inhabitant performing the action. The attributes containing a colon represent standard XES attributes identified by a name. Such a log easily allows for the application of supervised machine learning methods for activity recognition and modeling, as all of the occurrences of specific activities are labeled and encapsulated in the concept of XES trace.

2. The *unsegmented log* is made up of a single trace containing a sequence of events as described for the *segmented log file.* As such, events coming from multiple activities and multiple users are interleaved and not labeled with the corresponding activity. This log can be used for unsupervised learning techniques where labels are either not relevant or must be automatically found. In the latter case, the ground truth can be read from the *segmented log file.*

3. Finally, the *sensor log* file contains sensor measurements from the environment. This XES file, as the unsegmented log, contains a single trace. Each event in this trace represents a sensor measurement containing a set of attributes consisting of *(i)* a `concept:name` attribute representing the name of the sensor, *(ii)* a `org:resource` attribute whose value can be `SYSTEM` if the sensor measurement is from the environment or an inhabitant identifier if the sensor is inhabitant specific (e.g., the position sensor), *(iii)* a `time:timestamp` attribute indicating

the triggering time of the sensor measurement in the virtual clock, *(iv)* an `eventId` attribute indicating the identifier of the event, corresponding to those used in the segmented and unsegmented log files, that influences the sensor measurement, and *(v)* a `value` attribute containing the measured value for the sensor. The sensor log can be employed when the object of the analysis is the direct recognition of activities from sensor measurements or when the goal is to learn action models from sensor measurements.

The described logs are automatically obtained from the simulator by mapping main event repetitions to traces (which describe the execution of an activity), bottom events to events (which denote the execution of an action), and sensor measurements to virtual sensor output in the simulator as a consequence of a sensor event or of bottom events. The three log files can be integrated by means of event and trace identifiers, which allow for the mapping of sensor measurements to events and events to traces.

The simulator, with all its new features, can be downloaded at the link in the footnote[2]. The repository does not only include the executables needed to run the tool but also a set of models, including human models, house models, acting models, and behavior patterns, as described in this section. The goal of this repository is to provide a shared workspace that the community can contribute to. The creation of models is indeed an expensive task; thus, the availability of such a repository can speed up the definition of scenarios to produce workbenches.

3.4 Validation

Simulators are usually evaluated by comparing a real dataset from a smart space with a dataset generated by the simulator. Usual comparison-based approaches are:

- comparing the probability of each sensor event in both datasets [117];
- comparing the sequence order of sensor events in both datasets [117];
- applying activity recognition techniques to both datasets and comparing accuracy metrics [103];
- use data similarity measures for time series [52].

Such comparisons are possibly accompanied by usability questionnaires [53, 8]. As the simulation tool proposed in this chapter is fully configurable, we do not evaluate the features of the produced dataset, which vary according to the simulation parameters. Instead, we propose a two-fold evaluation. In Section 3.4.1, we first discuss which of the features available in free datasets are supported by our tool. Then, in Section 3.4.2, we demonstrate how we can apply a seminal activity recognition algorithm to a synthetically generated dataset, reproducing the features of the ones originally employed for validation and obtaining very similar results in performance evaluation, thus demonstrating how the synthetic dataset can be as good as a real

[2]`https://github.com/silvestroveneruso/smart_space_model_based_simulation`

one as a means to evaluate algorithms for smart spaces. Finally, in Section 3.4.3, we show the results obtained in the evaluation's phase.

3.4.1 Replicability of datasets available in literature

In order to assess the suitability of our tool to replicate the features of real and freely available datasets in Chapter 2, we will discuss how each of them is supported.

As a first aspect, the proposed tool produces sensor measurements in the form of *stream*. This is the most powerful of the sensor value representation modalities, as it is always possible to turn a stream of sensor measurements into *snapshots* or *summaries*, as these latter simply reduce the time granularity, whereas the opposite is in general not possible. At this moment, it is not possible to directly choose the last two modalities, which require performing a simple pre-processing of data.

Concerning the types of sensors supported, our tool can be used to easily introduce any kind of *environment-attached* sensor. It is also possible, though more difficult, to simulate body-worn sensors. Environment-attached sensors can provide information about both humans and the environment itself. Sensors can produce both *discrete* and *continuous* values.

Our tool can be configured to produce a dataset containing any number of simulated inhabitants and any number of activities. Additionally, actions in datasets and sensor measurements are precisely labeled with respect to the virtual inhabitant doing or triggering them and the activity performed at that time. The tool does not support collaborative activities between inhabitants that are found in some datasets (e.g., in the CASAS project repository).

The tool allows for the production of datasets of any length, which is a great advantage as real-life datasets are typically rather short (i.e., activities are not performed many times, which makes it difficult for algorithms to pick up patterns in the sensor log that correspond to specific activities). However, a long real dataset may show a variability arguably higher than a synthetic dataset. Even though it is possible to artificially add noise, as argued for example in [124], the description of human activities in the form of a business process (especially using so-called imperative modeling techniques, which behavior pattern formalism belongs to), cannot catch the entire freedom and variability a human has, thus producing a representation that is usually idealized.

One of the advantages of our simulator with respect to real datasets is that the dataset includes both action events and sensor measurements, which is unprecedented and allows for the analysis of not only activities but also actions.

3.4.2 Truthfulness evaluation

In order to evaluate the truthfulness of the datasets produced by our simulator and their suitability as an evaluation mean for algorithms in the smart space community, we have replicated as closely as possible two real-life scenario datasets from the CASAS project. The goal of this project is to treat environments as intelligent agents, where the status of the residents and their physical surroundings are perceived using sensors. On the basis of this information, a smart home can select and automate actions that meet the needs and preferences of its inhabitants, improving comfort, safety, and/or productivity. The reason for this choice relies on the major impact this project has had on the community and the availability of source code for algorithms that are often used as benchmarks [54, 72, 124, 50, 49, 98].

Among the several ones available at the link in the footnote[3], we selected two real scenario datasets:

1. a single-user dataset, namely *aruba* [45]. This dataset is a partially labeled sensor log that contains data collected in a real-life scenario. The inhabitant interacts with 4 temperature sensors, 31 motion sensors, and 4 door closure sensors. This dataset contains 6477 labeled activities. It includes eleven different labels: *Bed_to_toilet, Eating, Enter_home, Leave_home, Housekeeping, Meal_preparation, Relax, Resperate, Sleeping, Wash_dishes* and *Work*. We replicated the dataset by modeling a similar set of activities performed by the single inhabitant by using behavior pattern modeling formalism. We here only consider seven of them, skipping the infrequent ones. In addition, the *Other_activity* label is skipped as too generic. Additionally, we recreated the original layout of the *aruba*'s smart environment, provided by the authors of the testbed themselves, by using the Home Designer component of the simulator (see Figure 3.3), including the same list and positioning of sensors.

2. a multi-user dataset, namely *tulum* [42]. This dataset is a partially labeled sensor log that contains data collected in a real-life scenario by two inhabitants. The two volunteers interact with 5 temperature sensors and 31 motion sensors. This dataset contains 12637 labeled activities. It includes sixteen different labels: *R1_Sleeping_in_Bed, Personal_Hygiene, Bathing, Leave_Home, Enter_Home, Meal_Preparation, Watch_TV, Eating, Bed_Toilet_Transition, R2_Sleeping_in_Bed, Work_Table, Work_Bedroom_2, Yoga, Wash_Dishes, Work_LivingRoom* and *Work_Bedroom_1*. Labels' prefixes R1 and R2 denote the inhabitant that performed the *Sleeping_in_Bed* activity. We replicated the dataset by modeling a similar set of activities performed by the two inhabitants using the behavior pattern modeling formalism. We only consider eight of them here, skipping the infrequent ones. In addition, the *Other_activity* label is skipped as too generic. Additionally, we recreated the original layout of the *tulum*'s smart environment, provided by the authors of the testbed themselves, by using the Home Designer component of the simulator, including the same list and positioning of sensors.

Then, we evaluated the performance of a state-of-the-art human activity recognition (HAR) algorithm against both the real datasets (i.e., *aruba* and *tulum* from the CASAS project) and the synthetic datasets produced by our simulator "emulating" the original ones. Such datasets produced by our simulator can be found at the link in the footnote[4].

As a validation measure, we chose the *Activity Learning* (AL) software tool from the CASAS project itself [45], which is freely downloadable from `http://casas.wsu.edu/tools/`. AL contains a number of activity learning components,

[3]`http://casas.wsu.edu/datasets/`

[4]`https://drive.google.com/file/d/1DUe5MpOtIgVZnWNgEjqccZvhtyPUkf2m/view?usp=sharing`

which include activity modeling and recognition (AR), activity discovery (AD), and activity prediction (AP). In particular, among the different algorithms included in the repository, we used the AR component. Such a component learns models of activities from sensor data and can use these models to automatically label a sequence of sensor data with the corresponding activity. The AR algorithm is a sliding window-based approach that learns models of activities from sensor data and can use these models to automatically label a sequence of sensor data with the corresponding activity. For the classification task and for learning the activity models, AR adopts *support vector machines* (SVM). Further details are discussed in [111]. The rationale behind this choice is that this algorithm is one of the most representative examples of AR in the literature. In addition, AR is by far the most frequently tackled problem in the community.

3.4.3 Results

Firstly, we can notice a difference in terms of sensor activations produced by the simulation with respect to the dataset selected as the testbed. The *aruba* dataset proposes an average of 6676 sensor activations/day, while its synthetic emulated counterpart, produced by the simulator, has an average of 202 sensor activations/day. The *tulum* dataset proposes an average of 8311 sensor activations/day, while its synthetic emulated counterpart, produced by the simulator, has an average of 546 sensor activations/day.

These differences can be artificially filled by tuning up the simulator. For instance, by increasing or decreasing the value related to the *virtual clock*, i.e., the mechanism that manages the frequency of the simulation steps over time. Low values will increase the number of sensor activations generated over time; conversely, high values will decrease them. It is also possible to adjust the *noise* level of the produced synthetic log, further increasing or decreasing the number of sensor activations generated over time.

However, this discrepancy in terms of sensor activations per day does not imply differences in the behavior or in the quality of the synthetic datasets produced by the simulator. By running the AR algorithm, we computed the *activity recognition accuracy* score for each pair of datasets, the state-of-the-art dataset used as a benchmark and its synthetic emulated counterpart produced by the simulator.

The *aruba* dataset resulted in a score of **0.994607**, while its synthetic counterpart resulted in a score of **0.993711**. The related *confusion matrixes* are shown in Tables 3.1 and 3.2. The differences in terms of numbers in such tables are due to the discrepancy in sensor activations discussed at the beginning of this section.

In a similar way, the *tulum* dataset resulted in an *activity recognition accuracy* of **0.994997**, while its synthetic counterpart resulted in a score of **0.991672**. The related *confusion matrixes* are shown in Tables 3.3 and 3.4.

This shows that the synthetic logs produced by the simulator are comparable to their state-of-the-art counterparts (*aruba* and *tulum*) used as benchmarks, and therefore that the simulator is able to replicate a real behavior such as that collected in the datasets used as testbeds, whether single-user or multi-user.

Table 3.1. The *confusion matrix* obtained by running the AR algorithm on the *aruba* dataset.

	Sleeping	Relax	Work	Meal_preparation	Housekeeping	Bed_to_toilet	Wash_dishes
Sleeping	5695	0	0	0	0	0	0
Relax	0	19751	0	2	1	0	0
Work	0	0	3597	0	1	0	0
Meal_preparation	0	3	0	15343	0	0	0
Housekeeping	0	5	1	1	5803	0	0
Bed_to_toilet	0	0	0	0	0	249	0
Wash_dishes	0	0	0	7	2	0	2207

Table 3.2. The *confusion matrix* obtained in output by running the AR algorithm on the dataset produced by the simulator “emulating” the original *aruba* dataset.

	Sleeping	Relax	Work	Meal_preparation	Housekeeping	Bed_to_toilet	Wash_dishes
Sleeping	267	0	0	0	1	0	0
Relax	0	269	0	1	2	0	0
Work	0	1	396	1	1	0	0
Meal_preparation	0	0	0	673	1	0	0
Housekeeping	0	1	0	1	141	0	0
Bed_to_toilet	0	0	2	0	0	145	0
Wash_dishes	0	0	0	0	0	0	152

Table 3.3. The *confusion matrix* obtained by running the AR algorithm on the *tulum* dataset.

	R1_Sleeping	Personal_Hygiene	Bathing	Meal_Preparation	Watch_TV	Eating	R2_Sleeping	Wash_Dishes
R1_Sleeping	4655	7	3	0	3	0	28	0
Personal_Hygiene	6	20555	29	12	13	8	3	3
Bathing	5	31	12855	1	3	2	1	1
Meal_preparation	0	13	9	85121	44	99	1	8
Watch_TV	1	17	2	49	82782	66	0	8
Eating	0	13	3	135	106	60608	1	13
R2_Sleeping	13	6	3	0	0	1	33597	0
Wash_Dishes	1	4	0	20	16	20	0	7999

Table 3.4. The *confusion matrix* obtained in output by running the AR algorithm on the dataset produced by the simulator “emulating” the original *tulum* dataset.

	R1_Sleeping	Personal_Hygiene	Bathing	Meal_Preparation	Watch_TV	Eating	R2_Sleeping	Wash_Dishes
R1_Sleeping	207	1	1	0	1	3	6	0
Personal_Hygiene	3	281	3	2	2	0	3	1
Bathing	2	6	137	0	0	0	1	0
Meal_preparation	0	4	2	3659	21	36	0	1
Watch_TV	0	5	0	13	4673	7	2	2
Eating	0	2	0	32	12	2932	0	3
R2_Sleeping	6	1	1	1	1	0	940	0
Wash_Dishes	0	0	1	5	0	3	0	418

Chapter 4

Representing smart home data

As the utilization of Internet of Things (IoT) devices in support of business processes (BPs) becomes more frequent, there is a growing recognition of the potential to leverage the data collected by these devices for process mining (PM). Most current PM methods that can incorporate IoT data follow a similar approach: the IoT data are pre-processed with event abstraction and event-case correlation techniques to be translated into an event log in XES format [26, 97, 164, 171].

Although this is an interesting initial approach to integrating IoT data into PM and it allows for the application of existing control flow and data-aware techniques, this method does not fully exploit the potential of IoT data. Often, the resulting high-level event log lacks contextual information (i.e., properties that can influence process execution, as explained in [26, 166]) that could be derived from the IoT data, or it has limited capability to incorporate such context information. Furthermore, by separating the abstraction phase from the analysis phase, the true potential of advanced algorithms to optimize both abstraction and model discovery together cannot be harnessed. For instance, the development of an IoT-enhanced decision mining algorithm requires direct access to lower-level IoT data to learn the most relevant features directly from the source data instead of relying on an error-prone event abstraction step that might leave important information behind at a lower granularity level [27].

In this chapter, we present a new data format based on the conceptual model described in [26]. We then describe an event log created following this format and show that this new format greatly facilitates more in-depth analyses of IoT-enhanced BPs.

4.1 Existing standards for event logs

When applying PM techniques to smart space data, the common approach is to pre-process sensor data with event abstraction and event-case correlation techniques and translate it into an event log in XES format [26, 97, 164, 171].

XES [86], the current standard event log model, is an XML-based model that mainly consists of the notions of event, case, and log (see example in Figure 3.2). The XES format has already been introduced and discussed in Section 3.2.

Recently, the increase in maturity of the PM field has increased the urge to create alternative models. Multiple propositions that relax some assumptions of XES and allow for more flexibility in event data storage have been presented, e.g., in [151, 81]. Among them, OCEL [81] was designed to be more suitable for storing event logs extracted from relational databases and is widely considered the main challenger of XES today. It introduces the concept of object, which generalizes the notion of case by allowing one event to be linked with multiple objects instead of a single case. This removes the necessity to "flatten" the event log by picking one case notion from the several potential case notions that often coexist in real-life processes.

Although these are interesting initial approaches to integrating IoT data into PM and allow for the application of existing control flow and data-aware techniques, these standard formats do not fully exploit the potential of IoT data. Often, the resulting high-level event log lacks contextual information that could be derived from the IoT data, or it has limited capability to incorporate such context information [26, 166]. In [27], the authors listed ten requirements for the storage of IoT-enhanced event logs and showed that both XES and OCEL failed to meet more than half of these requirements. The definition of an alternative format suitable for integrating sensor and event data is crucial in this context.

4.1.1 Process mining using IoT data

Challenges related to the application of PM to IoT data, and in particular to how event logs are represented, are reported in [27, 124]. The vast majority of the PM literature involving IoT data has focused on mining high-level events of the process from low-level IoT data to create XES event logs. Various frameworks to extract high-level logs from IoT data have been presented [207, 188, 164, 189, 110, 108, 97, 59, 125]. Traditional PM techniques can then be applied to these event logs to, e.g., discover control-flow models of the processes. A recent survey of process discovery in smart spaces, which represents a large chunk of the literature on PM applied to IoT, is provided in [24].

Although most of the existing literature is on IoT event abstraction, some other possible techniques have also been investigated. Banham et al. [19] propose to perform data-aware process discovery with IoT-based attributes. The framework proposed requires abstracting the IoT data to integrate it into an XES event log. A second work is proposed by Rodriguez-Fernandez et al. [160], who present an approach for IoT-enhanced deviation detection in the time series data directly (in a so-called *time-series log*). Remark that all these papers bumped into the limitations of traditional event logs and had to abstract the data first or use the raw sensor data.

Another important research direction is represented by the automated segmentation of logs. Here, statistics-based techniques have been proposed [121], as well as techniques based on the structure of possibly mined processes [71], as discussed in Chapter 5.

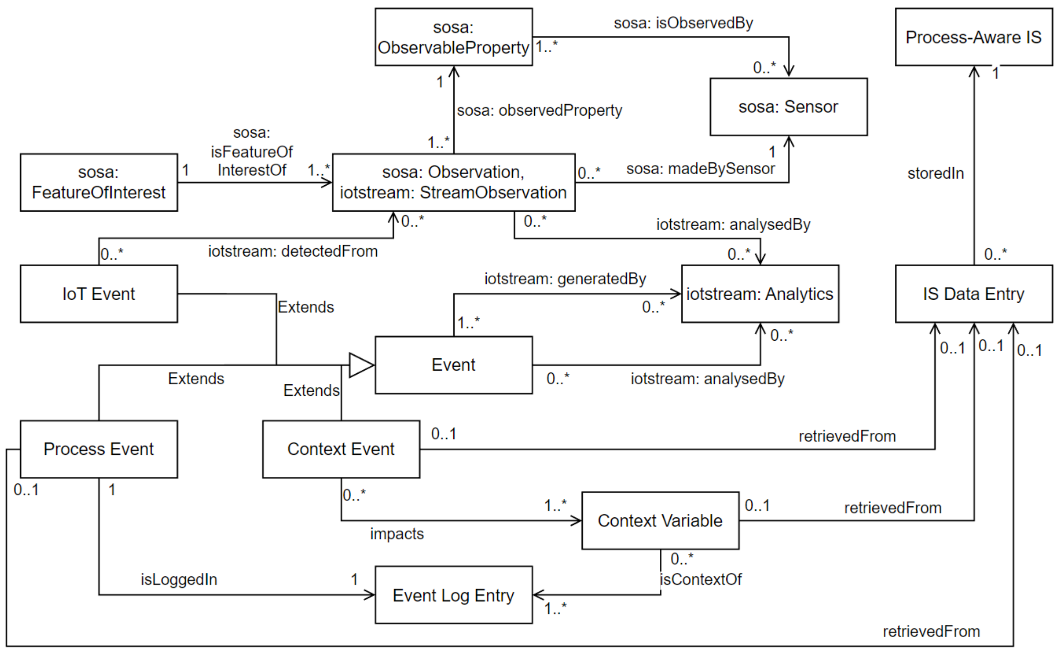


Figure 4.1. Core of the bridging model for IoT and PM (picture taken from [26]).

4.1.2 Existing models bridging IoT and process mining

A first model for bridging IoT and PM was presented in [26] (see Figure 4.1). This model integrates IoT ontologies (like SOSA [96]) with typical PM concepts around the central notion of event, which is decomposed into three types of events: *(i)* IoT events, which correspond to events as understood in IoT systems; *(ii)* Process events, which are events as understood in PM; *(iii)* Context events, which correspond to changes in a parameter of the context of the process. For more information, see Section 4.2.1 or [26].

4.2 Format specification

In this section, we outline the format we propose for IoT-enhanced event logs and explain how we operationalized it to generate XML-based NICE logs. It builds further upon the conceptual model presented by the authors in [26], which aimed at integrating IoT ontologies and event log models. To do so, multiple event types were defined to bridge the gap between the (physical) IoT world and the (mostly digital) PM world. In this work, in line with design science research [91], we adopted an iterative approach, moving between design and evaluation through prototyping in order to continuously improve the quality of the event log format. The evaluation step verified *(i)* that all constructs and relationships of the meta-model were transcribed correctly in the XML and *(ii)* that the format was capable of representing event logs from various types of IoT-enhanced BPs.

4.2.1 Meta-model

At its core, our data format consists of lists of *Data Sources*, *Objects*, and *Events* (see Figure 4.2a). An *Event* is related to one or several *Objects*, which can be digital (defined in the format as Digital Object, e.g., an order) or physical (defined as Feature of Interest, e.g., a fridge), or both (e.g., a parcel). All *Objects* can have a collection of *Properties*, which can also be digital or physical (e.g., the total amount of the order, the temperature in the fridge, the weight, and the value of the parcel), and represent context parameters of the process. *Events* are derived from one or several data entries or lower-level events that are logged by a given data source, which can either be an information system or a sensor. We distinguish between three types of events, which follow the hierarchy shown in Figure 4.2b:

- `IoT Event`: an instantaneous change in a real-world phenomenon that is monitored by a sensor or derived from lower-level IoT events. For instance, the temperature in a fridge is decreasing.

- `Process Event`: an instantaneous change of state in the transactional lifecycle of an activity (corresponding to the usual notion of event in PM). This type of event is deduced from one or more IoT events or taken from an IS data entry. For instance, a product is taken from a fridge and loaded into a truck.

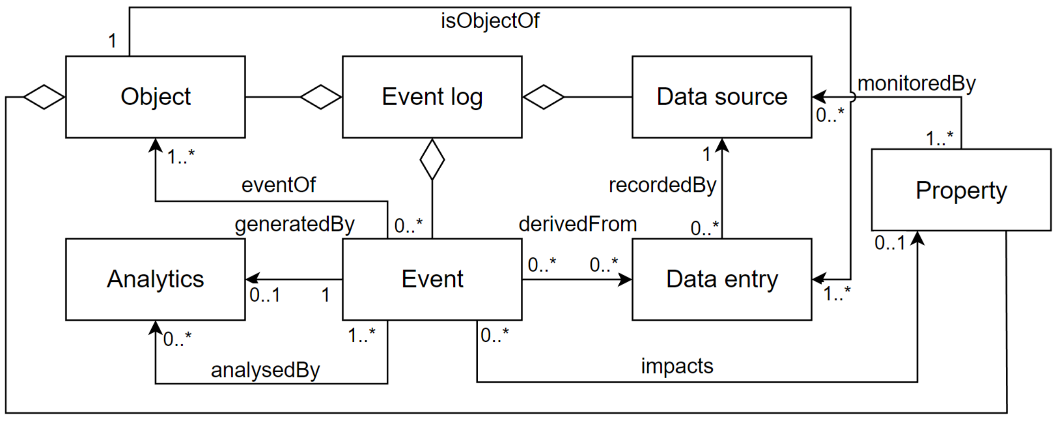
- `Context Event`: an instantaneous change in a real-world phenomenon or a digital property that has an impact on the execution of a specific process instance (i.e., it impacts a property of an Object) but does not change its control-flow state. Such an event typically influences the path followed in a process instance, how an activity is executed, etc. This type of event is deduced from one or more IoT events or taken from an IS data entry. For instance, the pressure in a tank exceeds a maximum threshold, prompting a human intervention in a chemical production process.

Higher-level events can be derived from lower-level events via the notion of *Analytics*, which represents any technique used for event abstraction (e.g., rule-based reasoning techniques like CEP and data-driven models). IoT events can cascade until a deduced event has a direct relationship with the process, i.e., it is a Process Event or a Context Event.

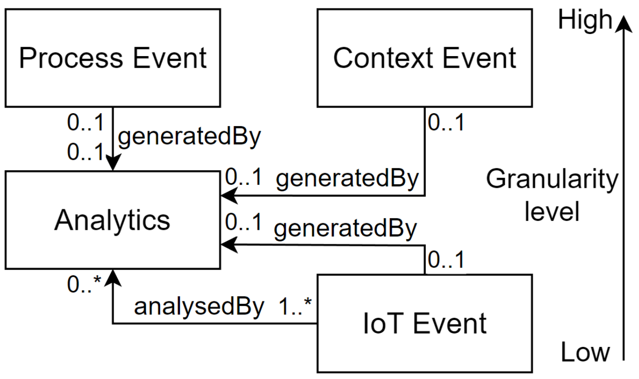
4.2.2 Implementation

In this section, we describe how we translated the model in Figure 4.2a into an XML format for NICE logs. XML was chosen for its flexibility and popularity, though the model could also be transcribed in other languages such as JSON or YAML. The XML schema of the processed log is structured as follows: the `EventLog` tag is the root and contains three main elements, identified respectively by the tags:

1. `ObjectList`: this element contains a list of objects. In IoT-enhanced BPs, different objects interact, and an event can involve a combination of objects (e.g., users, locations). The properties of objects are represented as attributes, which can be updated by Context Events.



(a)



(b)

Figure 4.2. Figure (a) shows the core of the NICE meta-model. Figure (b) shows the hierarchy of events in NICE logs (picture taken from [26]).

2. `DataSourceList`: this element contains a list of data sources available within the observed environment. We distinguish between two types of data sources: *(i)* physical data sources, typically sensors, that monitor physical properties, e.g., the opening of a door or the temperature; and *(ii)* digital data sources, like process-aware information systems (PAISs), that record digital properties and Process Events.

3. `EventList`: this element contains a list of events. There are three types of events: `IoTEvent`, `ContextEvent`, and `ProcessEvent`. `IoTEvent` is used to represent low-level types of events, such as raw sensor measurements, and, if relevant, mixed-level types of events, such as the aggregation of lower-level `IoTEvents` into *actions*, i.e., atomic interactions with the environment or a part of it (e.g., sitting on a chair, opening the fridge). A `ContextEvent` represents a change in the value of a property of an object that impacts the execution of the process (e.g., influences a decision). It is related to an object and the property it updates, and it contains this property's new value. Finally, `ProcessEvent` is used to represent high-level types of events, e.g., groups of human atomic interactions with the environment that are performed with a common goal. It

represents the execution of a business activity (e.g., prepare dinner, register an order). Each event is associated with an identifier, a timestamp, and one or more objectIDs. An `IoTEvent` has an additional child element called `Observation` that contains the raw sensor value or the action, if available. While a `ProcessEvent` has two child elements that refer to the name of the related `Activity` and its `LifeCyclePhase` (e.g., start, complete).

These three levels of events are connected through the `Analytics` element that references the lower-level events that generate a higher level of event and the `Method` used. For instance, the mixed-level `IotEvent` called "open the fridge" in its `Analytics` has references to the lower-level events related to the opening of the fridge door and the triggered motion sensors near the fridge. The higher-level `ProcessEvent` related to the activity "cook", in its `Analytics`, has references to the mixed-level events that all together compose the activity "cook", including the mixed-level `IotEvent` "open the fridge". Figure 4.3 shows examples of `Analytics` elements used to derive mixed-level IoT events and process events.

For our experiments, the simulator presented in Chapter 3 [195], which was originally designed to produce logs in the XES standard format, has been extended to generate NICE logs. Using a Python script, the synthetic log produced was then processed to fit the meta-model proposed in this article, following the XML structure described above. The Python script and the resulting log are available in this repository: `https://github.com/silvestroveneruso/NiceParsingTool.git`.

4.3 Format validation

In this section, we evaluate our new format from two angles: *(i)* theoretically, we compare it with the list of requirements outlined in [27]; *(ii)* in practice, by showing how it can be used to gain a better understanding of a process.

4.3.1 Theoretical requirements fulfilment

In this section, we confront our model with the requirements outlined in [27]. These requirements were structured around four challenges for IoT-enhanced logs: data granularity (C1), control-flow - context perspective convolution (C2), scope of relevance (C3), and data dynamicity (C4):

- **R1** (Store high-level events, C1): obtained with process and context events;
- **R2** (Store low-level events, C1): obtained with IoT events;
- **R3** (Store intermediary events, C1): obtained with IoT events derived from others;
- **R4** (Enable traceability between high-level and low-level events, C1): achieved with the concept of Analytics, which links derived events with the events they are derived from;

```
    ...
    <IoTEvent id="66" timestamp="2020-01-01T00:31:58" objectID="objID_1,objID_2">
  <Observation id="obs_66" resultTime="2020-01-01T00:31:58" value="Go_bathroom_sink"
       featureOfInterest="objID_24"/>
  <Analytics itGenerates="eventID_66" itAnalysesEvents="eventID_62,eventID_65">
              <Methods> <Method name="CEP"/> </Methods>
        </Analytics>
    </IoTEvent>
    ...
    <IoTEvent id="86" timestamp="2020-01-01T05:36:18" caseID="objID_1,objID_2">
        <Observation id="obs_86" resultTime="2020-01-01T05:36:18" value="Go_bed"
             featureOfInterest="objID_26"/>
        <Analytics itGenerates="eventID_86" itAnalysesEvents="eventID_68,eventID_70,eventID_75,
             eventID_78,eventID_80,eventID_85">
            <Methods> <Method name="CEP"/> </Methods>
        </Analytics>
    </IoTEvent>
    ...
    <IoTEvent id="88" timestamp="2020-01-01T05:36:22" objectID="objID_1,objID_2,objID_23">
        <Observation id="obs_88" resultTime="2020-01-01T05:36:22" value="OFF" sensor="s3"
             featureOfInterest="objID_23"/>
    </IoTEvent>
    <IoTEvent id="89" timestamp="2020-01-01T05:40:52" objectID="objID_1,objID_2,objID_23">
        <Observation id="obs_89" resultTime="2020-01-01T05:40:52" value="ON" sensor="s3"
             featureOfInterest="objID_23"/>
    </IoTEvent>
    <IoTEvent id="90" timestamp="2020-01-01T05:40:52" objectID="objID_1,objID_2">
        <Observation id="obs_90" resultTime="2020-01-01T05:40:52" value="Go_in_bed"
             featureOfInterest="objID_23"/>
        <Analytics itGenerates="90" itAnalysesEvents="88,89">
            <Methods><Method name="CEP"/></Methods>
        </Analytics>
    </IoTEvent>
    <ProcessEvent id="91" timestamp="2020-01-01T05:40:52" objectID="objID_1,objID_2">
        <Activity value="Sleeping"/>
        <LifecyclePhase value="complete"/>
        <Analytics itGenerates="91" itAnalysesEvents="66,86,90">
            <Methods><Method name="CEP"/></Methods>
        </Analytics>
    </ProcessEvent>
    ...
```

Figure 4.3. An example of events connected by the `Analytics` element. In particular, in this portion of the log, there are five `IoTEvents` and one `ProcessEvent`. The mixed-level `IoTEvent` with ID 90, which represents the atomic action *Go_in_bed*, is generated by the analysis of the low-level `IoTEvents` with IDs 88 and 89, with the `Method` called CEP. The high-level `ProcessEvent` with ID 90, which completes the Event *Sleeping*, is generated by the analysis of the mixed-level `IoTEvents` with IDs 66, 86, and 90, with the `Method` called CEP. The derivation of events with IDs 66 and 86 follows the same principle, but the lower-level `IoTEvents` from which they are derived are not shown for brevity.

- **R5** (Represent context at event level, C3): done with the properties of the objects linked to an event;
- **R6** (Represent context at activity level, C3): done with properties of the objects linked to the events of an activity; an additional activity object could link events together for convenience (link the events to the activity object and

the context to the activity object);

- **R7** (Represent context at case/object level, C3): context is represented at the object level with the object properties; case is a special object type;
- **R8** (Represent context at process level, C3): this requirement can be completed with a process-wide object (e.g., the house in a smart home log);
- **R9** (Update context parameters independently from process events, C2): context parameters (i.e., properties) are updated by context events and not by process events;
- **R10** (Update context parameters at a higher frequency (than process events), C4): this requirement is achieved together with R9, as context events can happen at any rate, distinguished from process events.

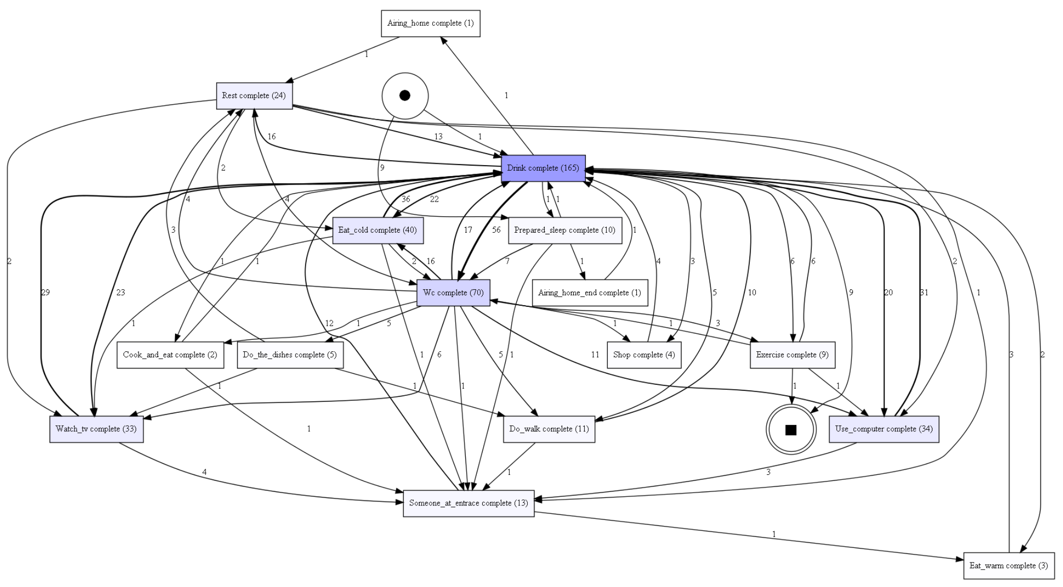
4.3.2 Log description

To showcase the usability of the data format, we generated a log simulating the behavior of two users living in a smart home for four weeks, executing activities such as *Cook_and_eat, Do_the_dishes, Drink, Eat_cold, Eat_warm, Exercise, Go_work, Sleep, Rest, Shop, Use_Computer, Watch_tv* and *Wc*, recorded as process events, and interacting with 30 motion sensors and a temperature sensor (each change in sensor value being recorded as an IoT event). The simulator (described in Chapter 3) was programmed so that during one week, the users show a different behavior, i.e., they stay at home the whole day instead of going to work in the morning and coming back home in the evening during weekdays. This change in behavior is due to extremely high outside temperatures during that week, which are tracked by temperature sensors and logged as IoT events.

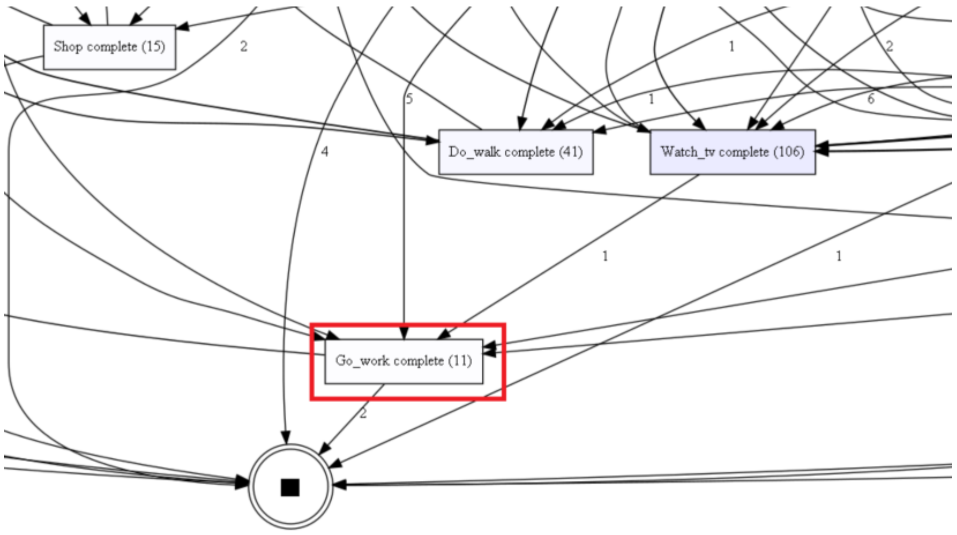
4.3.3 Log Analysis

Based on these IoT events, a context event is derived when the temperature exceeds the maximum acceptable temperature for work (set to 32.5 degrees for office workers in Belgium) and sets the property "Temperature suitable for work" to false. When the temperature decreases below the threshold again, another context event is generated, which sets "Temperature suitable for work" back to true. Figure 4.4 contrasts directly-follows graphs (DFGs), while the "Temperature suitable for work" property was false (see Figure 4.4a) and over the whole log (see Figure 4.4b). From the figure, we can see that the behavior of the users differs in both circumstances, as the activity `Go_work` is not performed during the heatwave period and more activities are recorded at home.

These context events make the change in the behavior of the users easy to explain, as the days of anomalous behavior are flagged with context events and characterized by a value of "Temperature suitable for work" equal to true. This way, all relevant information is stored in the log and is easily traceable, as lower-level IoT events are still present. Moreover, this approach makes it possible to set or mine different thresholds for "Temperature suitable for work", e.g., for users working in different sectors, and to create one context parameter for each user with different rules, which



(a) DFG of the behavior of the users during the heatwave.



(b) Zoom in on the `Go_work` activity in the DFG of the behaviour of the users over the whole log.

Figure 4.4. DFGs of the behavior of the users in the simulated log.

would be much more difficult in traditional event log formats. Once clearly identified with the context events, deviating behavior can be removed from the log or treated separately simply by removing the events recorded between the two context events. Finally, storing the low-level data in the log enables the use of, e.g., decision mining or trace clustering techniques to mine the finer-grained rules behind the break from work from the low-level data directly, which is not possible in a XES or OCEL log.

4.3.4 Comparison with the state-of-the-art

In the last years, several data formats have been adapted to IoT-enhanced event logs [82, 136, 195, 160]. In this section, we present these new alternatives and compare them to our data format with respect to the requirements identified in [27].

Of these models, the least tailored one to IoT-enhanced logs is D-OCEL [82]. It is an extension of OCEL [81] which introduces dynamic attributes, thereby solving one of the main issues faced by OCEL for IoT-enhanced event logs.

In [136], authors present the XES DataStream extension, which aims at facilitating the storage of IoT data in XES logs. DataStream introduces, among others, the notions of *point* and *datastream*, which respectively represent individual sensor observations and group together points that belong to the same event or trace. This enables the storage of all IoT data in the log, together with the impacted events or traces.

The simulator presented in Chapter 3 introduces another format based on XES. It generates multiple files: a segmented high-level event log, an unsegmented high-level event log, and a sensor log. The IoT-enhanced logs generated by this smart home simulator have been used and properly extended to support the NICE format.

Finally, the TS-log, a model that is completely independent from existing standards like XES and OCEL, was presented in [160]. This model is designed exclusively for TS data that is collected around a process and can be used to derive the execution of activities (i.e., process events). A TS-log therefore contains a set of variable names, a domain function for each variable, an index, and a function recording the value of each variable at each timestamp.

Comparing our model to related works, the NICE log format is the most flexible and balanced and the only one fulfilling all requirements expressed in [27] (see Table 4.1). OCEL does neither allow data attribute updates nor traceability and only has one event type. XES and D-OCEL have the same limitations, except that they allow for data attribute updates, but only with process events. XES with DataStream does much better but still requires all sensor data points to be linked to one process event, and the support for mixed-level events is uncertain. The simulator [195] does not support all context representation and enables only limited traceability (and information is scattered in multiple files). Finally, TS logs focus fully on low-level data and cannot store traditional process data.

Compared to the conceptual model presented in [26], the format of the NICE logs differs in several aspects. Primarily, NICE logs rely on the concept of object instead of the concept of case. This is because objects offer more flexibility and can be used to precisely define the scope of context parameters as object properties. Then, while the original conceptual model gave a more detailed description of IoT concepts than of process concepts, NICE logs harmonize both perspectives with overarching concepts.

		XES [86]	OCEL [81]	D - OCEL [82]	DS [136]	Simu-lator [195]	TS log [160]	NICE log
Requirements	R1: Store high-level events	✓	✓	✓	✓	✓		✓
	R2: Store low-level events				✓	✓	✓	✓
	R3: Store intermediary/mixed-level events					✓		✓
	R4: Enable traceability between high-level and low-level events				✓			✓
	R5: Represent context at event level	✓	✓	✓	✓	✓		✓
	R6: Represent context at activity level			✓	✓			✓
	R7: Represent context at case/object level	✓	✓	✓	✓			✓
	R8: Represent context at process level	✓		✓		✓		✓
	R9: Update context parameters independently from process events					✓	✓	✓
	R10: Update context parameters at a higher frequency				✓	✓	✓	✓

Table 4.1. Comparison between the different solutions with respect to the requirements described in [27].

Chapter 5

Unsupervised discovery of human habits

It has been argued that business process formalisms can be used as models for activities and habits [124]. In order to acquire such models, process mining (PM) techniques can be employed. When PM techniques such as *process discovery* are applied to data gathered from smart spaces, it is possible to model and visualize human habits and activities as business processes.

In this chapter, we first introduce a state-of-the-art methodology [71] allowing, given a sensor log, to automatically segment *human habits* by applying process mining techniques. Such methodology relies on a bottom-up discretization strategy for the timestamp attribute. Such discretization algorithms find the best division of a continuous attribute by iteratively merging contiguous sub-ranges (also called "bins") following a quality evaluation heuristic. In this case, the heuristic is based on quality measures computed on the process models automatically mined, through *process discovery*, from the intermediate bins.

The structural behavior of a Petri net mined through process discovery can be analyzed using several different quality measures (see Section 1.2.1). In the approach shown in this chapter, we drive the discretization targeting process models with high *simplicity* and low *structuredness*:

- *Structuredness* [114] is a measure obtained by disassembling the observed process model into small sub-models, assigning a score to each of them, and combining these scores. This metric gives higher penalties to sub-models that have many layers of components that are embedded in other components, which in general makes the whole net more difficult to understand, and also to sub-models in which the number of joins and the number of splits components do not match.

  Considering the patterns in Figure 5.1, the *sequence* pattern is considered the simplest one, so it has the lowest penalty value associated, which is obtained by summing the weights of all the transitions that are involved. The most complex component is the *unstructured* one, which thus has the biggest penalty value because we do not know which behavioral pattern it represents.

- *Simplicity* is a metric depending only on the size and structure of the model

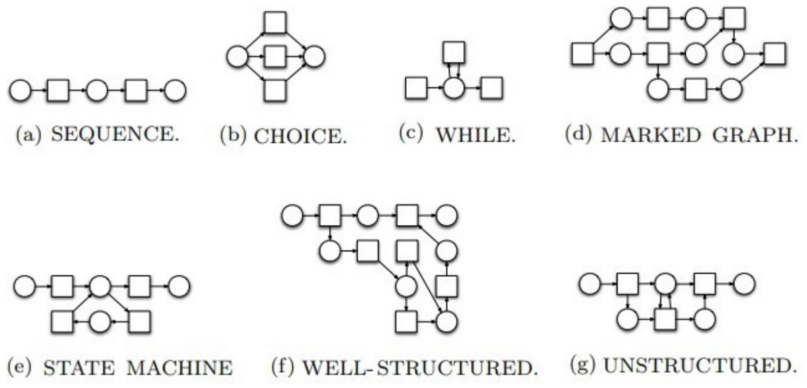


Figure 5.1. Examples of patterns that are considered when the original process model is iteratively disassembled into sub-models. Picture taken from [114].

without considering its behavior. Given the formal definition of a Petri net provided in Definition 1.1, where $|F|$ is the number of arcs, $|P|$ is the number of places, and $|T|$ is the number of transitions inside the model, we can define this metric as $\frac{|P|+|T|}{|F|}$.

A higher value of *simplicity* means that the Petri net is "simpler to understand". Conversely, if the equation returns a low value, we expect that the Petri net has a complex structure. Lower values of simplicity may happen instead when the number of arcs is much bigger than the number of nodes in the net, so in general, this leads to Petri nets that are not easy to read. Each obtained bin then represents a time range in which the human is supposed to perform activities following a clearly identifiable human process.

5.1 Proposed approach

In this section, we present step-by-step the approach implemented in this work. Figure 6.1 shows an overview of such an approach.

5.1.1 Data acquisition

A smart space produces data in the form of a *sensor log* $\mathcal{S}$, i.e., an ordered sequence of raw measurements. Measurements can be produced by a sensor within the smart space on a periodic basis (e.g., humidity measurements) or whenever a specific event is detected, such as the opening of a closet. In this work, we only consider measurements obtained from Presence InfraRed sensors (PIRs).

PIRs are widely used in smart environments as they provide a low-cost, low-power, small, and lightweight alternative in many application scenarios to other devices [144]. With respect, for example, to cameras, PIRs *(i)* are cheaper and consume less power; *(ii)* do not require sophisticated image processing algorithms; *(iii)* are not perceived as a privacy threat by the final user; and *(iv)* are easier to

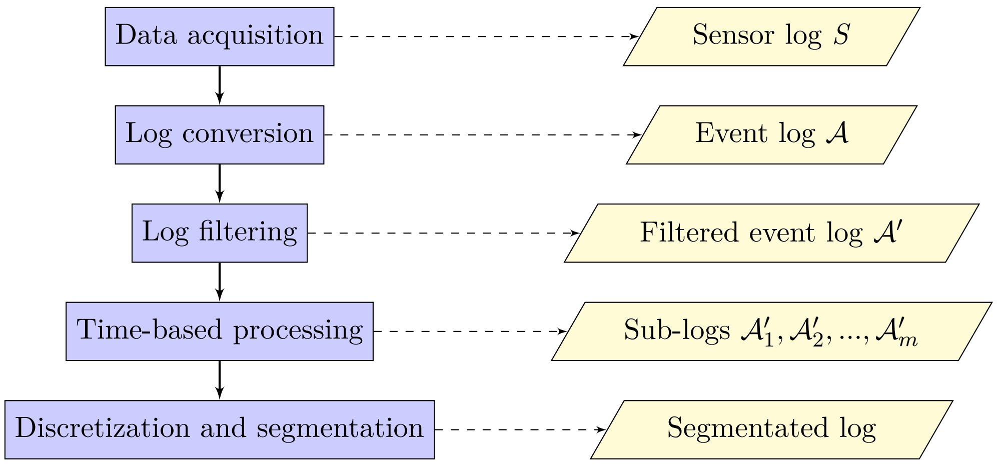

Data acquisition
Sensor log S
Log conversion
Event log $\mathcal{A}$
Log filtering
Filtered event log $\mathcal{A}'$
Time-based processing
Sub-logs $\mathcal{A}'_1, \mathcal{A}'_2, ..., \mathcal{A}'_m$
Discretization and segmentation
Segmentated log

Figure 5.2. Overview of the proposed approach. Blue boxes represent the sequence of steps of the unsupervised methodology described in this chapter. Yellow boxes represent the output of each of these steps.

setup. Similar considerations can be drawn about other positioning technologies (e.g., radio signal and audio-based positioning).

5.1.2 Log conversion

As pointed out in Chapter 1, a major requirement for applying process mining to smart spaces is to convert the sensor log into a segmented and labeled event log. To achieve this aim, we can employ the technique proposed in [125, 129]. Such a technique revolve around the TRACLUS algorithm [116]. TRACLUS is a trajectory clustering algorithm, originally devised for describing trajectories of hurricanes, consisting of two phases: *(i)* a trajectory partitioning technique using the minimum description length (MDL) principle, and *(ii)* a density-based line-segment clustering algorithm. The first part regarding the separation of the trajectory into sub-trajectories can be seen as an unsupervised segmentation technique: by considering the entire log as a long, complex trajectory, a sub-trajectory properly identified can define portions of the log relative to the same action.

The Visual Process Maps (VPM) system [129] exploits the partitions found by the first part of the TRACLUS algorithm to segment the log into sub-trajectories with a homogenous velocity. Each of these trajectory is then classified into three categories with labels `MOVEMENT, AREA`, and `STAY` by considering information features such as duration, velocity, and heterogeneity. In particular:

- `MOVEMENT`, if the sub-trajectory contains a “fast” movement from a sensor to another one;
- `AREA`, if the action involves many sensor measurements from the very area of the house;
- `STAY`, if the user stays under a given sensor “long” enough.

The classification allows for the replacement of sequences of measurements with human actions consisting of a category and a location, the latter inferred by the position of the corresponding sensors in the house. For example, the `<AREA Bathroom>` action indicates that the human was wandering around the bathroom, whereas `<STAY Kitchen_table>` represents the fact that the inhabitant remained for a while sitting at the kitchen table. At this point, once the sensor log has been turned into an action log, traces are defined by splitting the log on a daily basis. The obtained action log $\mathcal{A}$, which is a special case of an event log, is compatible with process discovery tools, e.g., the *inductive miner* to mine Petri nets.
The action log $\mathcal{A}$ is a timestamped sequence of tuples $\langle \mathtt{d}, \mathtt{s}, \mathtt{e}, \mathtt{a} \rangle$ where:

- `d` is the day in the log;
- `s` and `e` are, respectively, the timestamps at which the action starts and ends. Tuples follow a chronological order according to `s`;
- `a` is the result of the sensor aggregation. It is labeled as `STOP`, `AREA` or `MOVEMENT` and it is followed by the identifier of the position.

5.1.3 Log filtering

The event log $\mathcal{A}$, obtained from the previous conversion step, is further processed. Consecutive repetitions of the same action within the log are merged together. Specifically, given two consecutive tuples (representing consecutive actions) denoted as $T_1 = \langle Day_1, Start\_TS_1, End\_TS_1, Action_1 \rangle$ and $T_2 = \langle Day_2, Start\_TS_2, End\_TS_2, Action_2 \rangle$, the following conditions must hold: $Day_1 = Day_2$, $Action_1 = Action_2$, and $End\_TS_1 = Start\_TS_2$. If these conditions are satisfied, $T1$ and $T2$ are merged into a single action, defined as $\langle Day_1, Start\_TS_1, End\_TS_2, Action_2 \rangle$. This operation can be iteratively applied to a chain of multiple subsequent tuples. Refer to Table 5.1 for an illustration of this merging process.

5.1.4 Time-based processing of the log

Once the filtering task is completed, we can then proceed to the segmentation phase. For this purpose, we apply a classical bottom-up discretization technique based on the attribute referring to the time of day. Just to capture the underlying principle behind our approach, we can refer to the well known *Chi-merge*'s [132] strategy: this algorithm starts by dividing the entire range of an attribute at the finest level of granularity possible. Then, adjacent bins/intervals that met a condition based on χ^2 (i.e., the statistical measure *Chi*) are iteratively merged. If this condition is not met, the algorithm stops, and those bins/intervals remain separated.

We start our segmentation by dividing the entire range of the *time-of-the-day* attribute (i.e., 00:00 - 24:00) into bins of constant width. For instance, if 15 minutes is chosen as the minimum bin width, the *time-of-the-day* attribute will be divided into $24 * (60/15) = 96$ bins. Each bin is associated with a correspondent sub-log (of the unsegmented event log) where we have a case for each day in the dataset only containing actions with start_timestamp included in the bin (e.g., all the actions in a specific day happening since 00:00 to 00:15).

Day	Start_TS	End_TS	Action
		...	
1	11:34:00	11:54:00	AREA Living
1	11:54:00	11:56:32	STAY Kitchen_table
1	11:56:32	12:01:21	STAY Kitchen_table
1	12:01:21	12:02:04	STAY Kitchen_table
1	12:02:04	12:02:19	STAY Kitchen_table
1	12:02:19	12:05:05	STAY Dining_chair
		...	

(a)

Day	Start_TS	End_TS	Action
		...	
1	11:34:00	11:54:00	AREA Living
1	11:54:00	12:02:19	STAY Kitchen_table
1	12:02:19	12:05:05	STAY Dining_chair
		...	

(b)

Table 5.1. In this example, the portion of the action log shown in Table (b) represents the output of the merging of events from Table (a). In particular, four consecutive `STAY Kitchen_table` events are merged into a single one.

Definition 5.1. *Given an action log $\mathcal{A}$, we define eventLog$(\mathcal{A}, [a, b])$ as the event log having the day as a case identifier and containing, for each case, the actions performed between time a and b during the day associated with the case.*

5.1.5 Discretization and segmentation

Once this initial segmentation of the log is provided, the Algorithm 5.1 is executed. The algorithm takes as input *(i)* a finite set `intervals` of chronologically ordered intervals/bins (96 in the previous example), *(ii)* a parameter `minN` denoting the minimum number of intervals to be returned by the algorithm, and *(iii)* a parameter `minScore` representing the stop criterion.

The algorithm finds the best possible segmentation of the event log in habits, producing no less than `minN` intervals/bins (see row 1). At each iteration (see rows 2 to 16), the algorithm iterates on all the intervals/bins, and for each pair of adjacent intervals/bins (see row 5), it applies the *inductive miner* (see row 6) to the event log obtained by merging those intervals/bins. In order to also consider possible habit intervals that cover two consecutive days (e.g., a time interval that lasts from 23:00 to 01:00 of the following day), the merging of two adjacent intervals/bins is circular, so that the last interval/bin in the array is merged with the first.

For each couple of adjacent intervals/bins, the inductive miner produces a Petri net *pn*. For each of these process models, we compute a score obtained starting from the *simplicity* and *structuredness* measures introduced above. This equation allows

Algorithm 5.1: Discretization algorithm.

```
Data: A, intervals, minN, minScore
while len(intervals) > minN do
    maxScore = 0;
    index = null;
    for i in [0, len(intervals) - 2] do
        pair = concat(intervals[i], intervals[i+1]);
        pn = inductiveMiner(eventLog(A, pair));
        score = 100 $\times$ pn.simplicity - pn.structuredness;
        if score > maxScore then
            maxScore = score;
            index = i;
        end
    end
    if maxScore < minScore then
        return intervals;
    end
    intervals = merge(intervals, index, index+1);
end
return intervals;
```

us to find the pair of adjacent intervals/bins whose correspondent Petri net is the most readable one. In fact, in general, the Petri nets that are more simple and easy to understand have a high value for *simplicity* and a low value for *structuredness*. Thus, in each iteration, the couple of adjacent intervals/bins that may actually be merged are the ones that maximize the value obtained from this equation. *Simplicity* in particular is multiplied by a factor of 100, which has been empirically chosen in order to uniformize the two quality measures since *simplicity* values are always less than 1. We keep track of the maximum score computed and of the corresponding couple of adjacent bins (see rows 8 to 11).

Once all adjacent bins have been considered, if the maximum score computed is above the `minScore` threshold, the `intervals` array is updated by merging the adjacent bins corresponding to the maximum score. Otherwise, the algorithm ends, as any additional merging would not be convenient.

Noticeably, the algorithm always terminates. The merging phase stops either if it is not convenient to keep merging bins or if a minimum number of bins is reached (note that iteration by iteration, the number of bins is always decreased by one). After the algorithm terminates, the `intervals` variable contains a set of intervals/bins each corresponding to a habit. Here the rationale is that bins will be merged only if the resulting Petri net results are simpler and less structured, meaning that the process model of the underlying habit is easy to read.

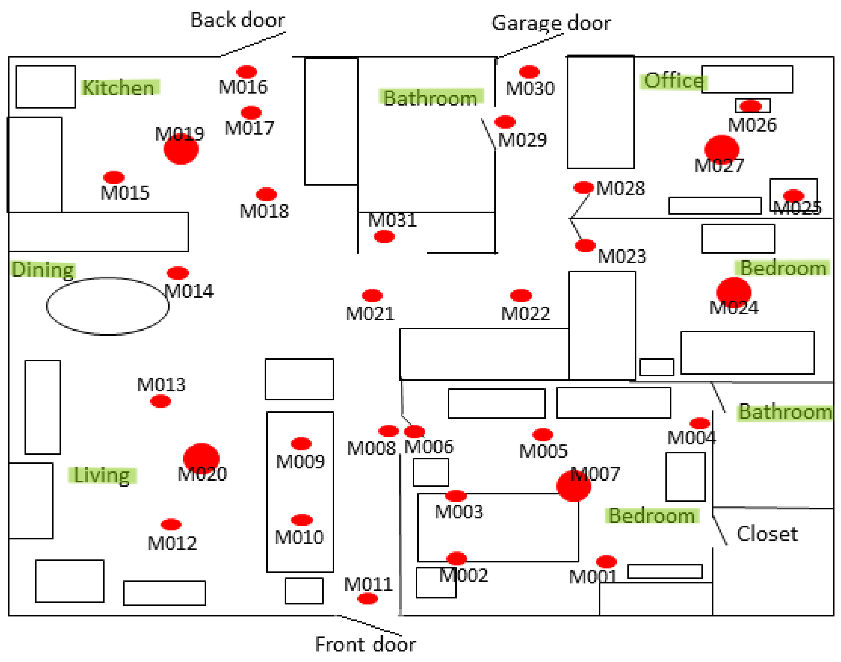


Figure 5.3. The layout of the real smart house used within the context of the state-of-the-art dataset *aruba* from the CASAS project.

5.2 Validation and Results

The approach described in Section 5.1 is validated against the *aruba* dataset from the CASAS project (`http://casas.wsu.edu/datasets/`). The reason for this choice relies on the major impact this project has had on the community and the availability as source code of algorithms that are often used as benchmarks [54, 124, 98, 127, 195].

The *aruba* dataset is a partially labeled sensor log that contains data collected in a real-life scenario for 220 days. It contains 6477 labeled activities, with the start and end markers of activities performed by the resident, meaning that for a subset of measurements, we know the activity correspondent to those activations of sensors. The activities available in the dataset are the following ones: *meal preparation*, *relax*, *eating*, *work*, *sleeping*, *wash dishes*, *bed to toilet*, *enter home*, *leave home*, and *housekeeping*. The inhabitant interacts, among others, with 31 Presence InfraRed (PIR) sensors that trigger as soon as a person enters their detection area, producing discrete measures that can be easily associated with actions. The layout of the *aruba* map with the arrangement of the sensors in the environment is shown in Figure 5.3.

Once obtained the action log $\mathcal{A}$ from the raw sensor log, Algorithm 5.1 is executed, passing as arguments *(i)* the pre-processed action log, *(ii)* an array of 96 initial bins obtained by segmenting the *time-of-the-day* attribute in equal-width bins of 15 minutes, *(iii)* a minimum number of bins equal to 2, and *(iv)* a minimum score chosen empirically. The six best intervals from our discretization algorithm are the following ones: *05:15-07:00*, *07:00-13:45*, *13:45-18:15*, *18:15-21:45*, *21:45-23:00*, and *23:00-05:15*. Each of these temporal intervals is considered a human *habit*.

In order to evaluate the result obtained, a simple statistical analysis of the dataset has been performed. In particular, each habit was mapped to a set of activities to

Activity	Most frequent intervals
Bed to toilet	00:30-00:45 03:30-06:15
Eating	08:00-08:15 09:15-10:45 12:00-12:15 13:45-14:15 18:00-19:00 19:15-19:45
Enter home	11:15-11:30 13:15-14:15 14:30-15:15 15:45-16:00
Housekeeping	10:30-11:45 14:30-15:15
Leave home	11:30-11:45 14:45-15:00
Meal preparation	06:00-06:30 06:45-10:45 12:00-12:30 13:15-14:45 15:15-20:15
Relax	08:45-09:30 13:30-00:00
Sleeping	23:30-07:30
Wash dishes	09:45-10:30 11:00-11:15 17:45-18:00 19:30-20:00 20:45-21:00
Work	12:00-12:30 14:15-14:30 15:45-16:45

Table 5.2. Most frequent temporal intervals for each activity.

Habit	Activities
05:15-07:00	sleeping, bed to toilet, meal preparation
07:00-13:45	sleeping, meal preparation, eating, relax, wash dishes, housekeeping, enter home, leave home, work
13:45-18:15	enter home, meal preparation, relax, eating, work, housekeeping, leave home, wash dishes
18:15-21:45	meal preparation, eating, wash dishes, relax
21:45-23:00	relax
23:00-05:15	relax, sleeping, bed to toilet

Table 5.3. Association between habit intervals and the activities occurring more frequently in the same time interval.

show its plausibility. As already stated, these activities have been manually labeled in the original sensor log $\mathcal{S}$. According to the activity label available in the raw log, we computed the temporal intervals in which each activity occurs more frequently (see Table 5.2).

Table 5.3 introduces another kind of association obtained by comparing the most frequent intervals for each activity with respect to the habits found by our discretization algorithm. This association shows in which habit a certain activity occurs more frequently. For instance, the activity *sleeping* occurs more frequently during the time interval "23:30-07:30" and so can be associated with both the habit intervals "23:00-05:15" and "05:15-07:00".

In addition, for each of the six identified habit ranges, the percentage of time spent in each of the related activities shown in Table 5.3 was calculated. Results

Habit	Percentage of time spent in each habit for relevant activities		
05:15-07:00	80.1% sleeping 4.1% other	10.7% meal preparation	5.1% bed to toilet
07:00-13:45	35% work 8.90% relax 5% eating 2% leave home	18.06% meal preparation 8.64% other 4% sleeping	10% housekeeping 5.4% wash dishes 3% enter home
13:45-18:15	25% work 13.52% other 5% wash dishes	20.69% meal preparation 10% eating 3% leave home	13.79% relax 7% housekeeping 2% enter home
18:15-21:45	25% eating 20.74% other	23.01% meal preparation 10% wash dishes	21.25% relax
21:45-23:00	75.57% relax	24.43% other	
23:00-05:15	85% sleeping 1.85% other	10.13% relax	3.02% bed to toilet

Table 5.4. The percentage of events related, for each habit interval, to the activities occurring in the same time interval.

are shown in Table 5.4. For example, the habit "05:15-07:00" has a total duration in the log of $105 * 60 * 220 = 13200$ seconds, which are mostly distributed among the activities *sleeping*, *bed to toilet* and *meal preparation*. Another example: during the habit "05:15-07:00", the sleeping activity is performed for 10567 seconds; thus, the sleeping activity represents about 80% of that habit. The shown percentages are helpful to understand how much a habit is linked to a specific activity. As we can note from the table, only a part of the habit is related to specific activities, while the rest of the time is either unlabeled or contains spurious activities and is denoted with the class *other*. Noteworthy, these results are obtained by considering the manual labeling of the human inhabitant in the original *aruba* log as the ground truth. Obviously, this labeling may be subject to imprecision and noise.

Chapter 6

Unsupervised discovery of human activities

Human activities have always been considered first-class citizens in research about smart home automation. The term activity denotes a set of actions performed to obtain a specific goal (e.g., cleaning the house, cooking). Being able to recognize the onset and end of an activity performed by a single human or by a group of human users can be helpful for different purposes. Examples of applications include the definition of automation rules or the detection of potentially harmful deviations from usual behavior.

Unfortunately, the practical applicability of techniques proposed in the literature is limited by the effort required by the final user to manually label smart home logs to train activity recognition models.

First, many approaches (e.g., [112]) require to manually specify window lengths or other kinds of numerical thresholds (e.g., number of events, minimum distance between two events). The selection of such parameters is hard and does not take into account the peculiarities of the different activity types. In second place, the vast majority of proposed solutions are directly applied to raw sensor measurements. This strategy does not exploit the meaning of a sequence of sensor measurements, only highlighting the statistical distribution of occurrences and co-occurrences of sensor events. Finally, in many cases, automatic segmentation techniques are only used to complement manual segmentation and are not intended to segment the full log (e.g., [46]).

In this chapter, we introduce a fully automated log segmentation technique able to mark the beginning and end of each activity repetition in a sensor log. In order to obtain this result, the proposed technique employs the information about human position in the log to extract high-level actions (e.g., standing still or operating in a specific area of the house). Then, inactivity periods are analyzed in order to perform the first segmentation. Finally, clustering is employed to identify classes of segments representing activities.

The rationale behind the employment of position information to abstract human actions is the frequent availability of sensors, such as Presence InfraRed sensors (PIRs), that provide, at a different level of granularity, measurements reporting the position of humans in the environment. Additionally, human positions provide

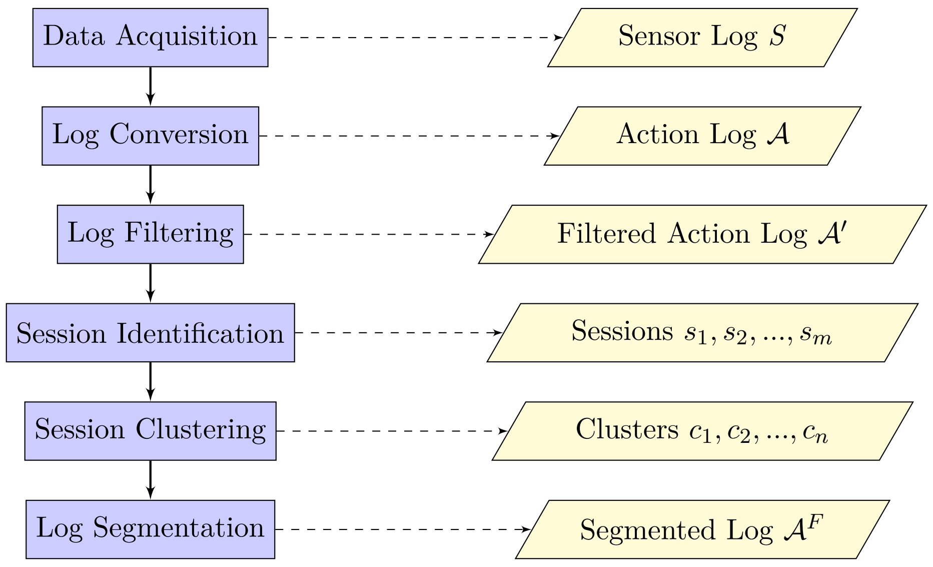


Figure 6.1. Overview of the proposed approach. Blue boxes represent the sequence of steps of the unsupervised methodology described in this chapter. Yellow boxes represent the output of each of these steps.

useful insights about the duration of human actions. Finally, the proposed technique is evaluated by comparing the automatic segmentation obtained by applying our method to the manual segmentation of logs available in the literature.

6.1 Proposed approach

In this section, we present step-by-step the data processing pipeline implemented in this work. Figure 6.1 shows an overview of such an approach.

6.1.1 Data acquisition

A smart space produces data in the form of a *sensor log* $\mathcal{S}$, i.e., an ordered sequence of raw measurements. Measurements can be produced by a sensor within the smart space on a periodic basis (e.g., humidity measurements) or whenever a specific event is detected, such as the opening of a closet. In this work, we only consider measurements obtained from Presence InfraRed sensors (PIRs), i.e., motion sensors that are activated whenever a human crosses their detection cone. For this type of sensor the same considerations made in Section 5.1.1 apply.

Position information, if a proper spatial resolution is granted, is very valuable as a proxy for human actions, provided that an association between positions and furniture is available. Noticeably, the approach described in this chapter can be easily extended to other categories of sensors, provided that a proper log conversion step (see Section 6.1.2) is implemented.

6.1.2 Log conversion

As highlighted in Chapter 1, directly analyzing the raw sensor log $\mathcal{S}$ may be counterproductive as measurements are too fine-grained, hiding behind variety and noise the actual behavior of the human user. In order to address this issue, in this work, we employ the technique we previously discussed in Section 5.1.2 to turn the raw unsegmented sensor log $\mathcal{S}$ into an unsegmented action log $\mathcal{A}$ [125].

The classification allows for the replacement of sequences of raw sensor measurements with human actions consisting of a category and a location. The location is deduced based on the positions of the associated sensors situated within the smart house. The obtained action log $\mathcal{A}$ is a sequence of tuples $\langle \texttt{Day}, \texttt{Start\_TS}, \texttt{End\_TS}, \texttt{Action} \rangle$ where:

- `Day` represents the specific day in the log.
- `Start_TS` and `End_TS` denote the timestamps indicating the start and end of each action. The tuples are organized in chronological order based on `Start_TS`.
- `Action` is the outcome of the classification that replaces sequences of raw sensor measurements. It is assigned one of these labels: `STAY`, `AREA`, or `MOVEMENT`, followed by the identifier of the position.

6.1.3 Log filtering

The action log $\mathcal{A}$, obtained from the previous conversion step, is further processed. Consecutive repetitions of the same action within the log are merged together. In order to address this task, we employ the same technique we previously discussed in Section 5.1.3 Refer to Table 5.1 for an illustration of this merging process. The resulting output of this merging step is referred to as $\mathcal{A}'$.

6.1.4 Session identification

The action log $\mathcal{A}'$ produced at the previous step is a continuous sequence of events where the end timestamp of an event corresponds to the start event of the following one. Some of the events, though, are represented by `MOVEMENT` actions, i.e., actions representing the human moving between two positions in the environment to perform more added-value actions, represented in the log through `STAY` and `AREA`. For example, once the user has finished lunch, he/she could move to the living room to watch TV. As a consequence, we remove the `MOVEMENT` events, thus obtaining a further filtered action log that we will denote as $\mathcal{A}''$. With respect to $\mathcal{A}'$, $\mathcal{A}''$ is discontinuous, i.e., it presents empty slots where a `MOVEMENT` action was previously present.

We now introduce the concept of *session threshold* Δ. In particular, we define Δ as the mathematical *median* of the duration of the empty slots left by removing the `MOVEMENT` actions. Intuitively, the value of Δ represents the usual duration of movements, allowing one to identify which movements are likely to represent a change of scenario, i.e., of activity.

After computing Δ, the action log $\mathcal{A}''$ is segmented into *sessions*. A session $s_i = \langle e_{i1}, ..., e_{in} \rangle$ in $\mathcal{A}''$ is defined as a sub-sequence of events where the time difference between the timestamp of the last event in the session and the timestamp of the first event in the following session is larger than the threshold value Δ.

As an example, we consider a sample portion $\mathcal{A}''_{sample} = \langle a_1, b_3, c_5, d_{11}, e_{13} \rangle$ of an action log $\mathcal{A}''$, where the letter indicates the action name and the subscript is the timestamp in which the action occurred (e.g., the action c occurred at time 5). Let us assume a session threshold $\Delta = 5$. From this example, we can see that the time difference between the action c and the action d is computed as $11 - 5 = 6$, which is greater than Δ. Then, we can derive two sessions $\langle s_1, s_2 \rangle$ from $\mathcal{A}''^{sample}$ where $s_1 = \langle a_1, b_3, c_5 \rangle$ and $s_2 = \langle d_{11}, e_{13} \rangle$. Notice that their concatenation results in $\mathcal{A}''^{sample}$.

At the end of this step, a sequence of m sessions $s_1, s_2, ..., s_m$ are identified, and the action log can be represented as a concatenation of all of them, i.e., $\mathcal{A}'' = \langle s_1, s_2, ..., s_m \rangle$.

6.1.5 Session clustering

Different sessions may represent several different ways of performing the same activity. In order to group all the possible ways to execute an activity, session clustering is performed. In this step, sessions are encoded as vectors to be subsequently fed into the clustering algorithm. Several encoding techniques can be applied. We introduce two of them, which are of general applicability and were implemented and used in the evaluation of this approach (see later in Section 6.2).

Both methodologies generate vectors with one dimension for each different action in the action log $\mathcal{A}''$ (e.g., `AREA Living` and `STAY Kitchen_table`). The complete set of actions that can be found within $\mathcal{A}''$ is denoted as $\{x_1, ..., x_n\}$.

- *Frequency-based encoding*: it is useful when we want to cluster on the basis of the frequency of the occurrence of actions in sessions. Each session s_k is encoded in a vector $v_k = \langle v_{x_1}, ..., v_{x_n} \rangle$ where the single component v_{x_i} is the number of occurrences of the action x_i in session s_k.

- *Duration-based encoding*: each session s_k is encoded in a vector $v_k = \langle v_{x_1}, ..., v_{x_n} \rangle$ where the single component v_{x_i} is the duration of instances of action x_i in the session s_k. The duration of an action is trivially computed as the time difference between the related end and start timestamps. If x_i does not occur in s_k, then $v_{x_i} = 0$.

The output of the clustering phase is a set of clusters $c_1, c_2, ..., c_n$, each cluster grouping together a set of sessions representing different ways of performing an activity.

6.1.6 Log segmentation

As we have seen in previous steps, the action log $\mathcal{A}''$ can be expressed as a sequence of sessions, and each session is a sub-sequence of actions $\langle e_{i1}, ..., e_{in} \rangle$. Additionally, in the clustering step, we mapped every session to a cluster c_i. As a consequence, we can label every action of a session with the cluster label of the session itself. For example, if the session $s_1 = \langle e_1, e_2, e_3 \rangle$ is clustered as c_3. The sequence of actions within the session $\langle e_1, e_2, e_3 \rangle$ can be labeled as c_3.

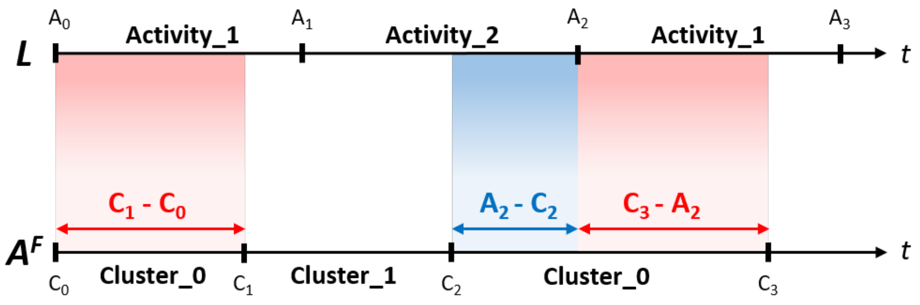


Figure 6.2. An example of the comparison between the labeled log L used as a benchmark and the log $\mathcal{A}^F$ generated by our approach. The red area represents the time intervals in which the $Cluster\_0$ coincides with $Activity\_1$, while the blue area represents the time interval in which the $Cluster\_0$ coincides with $Activity\_2$.

As a consequence, we can define the final segmented log $\mathcal{A}^F$ by adding to all the events in $\mathcal{A}''$ the label of the cluster to which they belong.

6.2 Experimental evaluation

The technique introduced in this work has been implemented as a Python script[1]. The implementation features several algorithms for clustering, including *K-means, spectral clustering, DBSCAN* and *OPTICS*.

To assess the validity of our approach, we compare the segmentation obtained by using our tool to the labeling provided with the dataset used as a benchmark. In Section 6.2.1, we discuss the criteria behind the selection of the dataset employed for the evaluation. In Section 6.2.3 we present the results of the evaluation using a state-of-the-art technique as a reference.

We will denote with L the labeled dataset selected for the evaluation and with $\mathcal{A}^F$ the log produced by applying our approach to L itself, ignoring the activity labels (see Section 6.1). The labeling provided by L represents our *ground truth* for the comparison.

L can be seen as a sequence of labeled n activities $\langle Activity\_1, ..., Activity\_n\rangle$. For each $Activity\_i \in L$ we have a starting timestamp A_{start} and an ending timestamp A_{end}, with $A_{start} < A_{end}$.

Similarly, $\mathcal{A}^F$ can be seen as a sequence of m clusters $\langle Cluster\_1, ..., Cluster\_m\rangle$ identified by our approach. For any $Cluster\_i \in \mathcal{A}^F$ we have a starting timestamp C_{start} and an ending timestamp C_{end}, with $C_{start} < C_{end}$.

The start and the end of the $Cluster\_i$ do not necessarily coincide with the start and the end of the $Activity\_i$. For instance, let us consider $Cluster\_0$ and $Activity\_0$ from Figure 6.2: their initial timestamps (A_0 and C_0) match, but the final ones (A_1 and C_1) do not.

For each identified $Cluster\_i \in \mathcal{A}^F$, we establish how much it coincides with each $Activity\_j \in L$. Let us consider again the example provided in Figure 6.2 where two small portions of L and $\mathcal{A}^F$ are provided. In particular, the example

[1]See: https://github.com/DIAG-Sapienza-BPM-Smart-Spaces/Activity-Segmentation.git

highlights the sequence of activities $\langle Activity\_1, Activity\_0, Activity\_1, ...\rangle$ and the sequence of clusters $\langle Cluster\_0, Cluster\_1, Cluster\_0, ...\rangle$, respectively from L and $\mathcal{A}^F$.

Considering only the portion of logs highlighted by the example, then $Cluster\_0$ coincides with $Activity\_1$ for a time span expressed as $\frac{(C_1-C_0)+(C_3-A_2)}{(A_1-A_0)+(A_3-A_2)}$. In a similar way, $Cluster\_0$ coincides with $Activity\_2$ for a time span expressed as $\frac{(A_2-C_2)}{(A_2-A_1)}$.

Still considering the example provided in Figure 6.2, let us specify some values for the constants: $A_0 = 0$, $A_1 = 5$, $A_2 = 10$, $A_3 = 15$ and $C_0 = 0$, $C_1 = 4$, $C_2 = 8$, $C_3 = 14$. The relationship between $Cluster\_0$ and $Activity\_1$ is $\frac{(C_1-C_0)+(C_3-A_2)}{(A_1-A_0)+(A_3-A_2)} = \frac{(4-0)+(14-10)}{(5-0)+(15-10)} = \frac{8}{10} = 0,8$. Then $Cluster\_0$ coincides with $Activity\_1$ for the 80% of the time in this portion of logs highlighted by this example. Similarly, the relationship between $Cluster\_0$ and $Activity\_2$ is $\frac{(A_2-C_2)}{(A_2-A_1)} = \frac{(10-8)}{(10-5)} = \frac{2}{5} = 0,4$. Then $Cluster\_0$ coincides with $Activity\_2$ for the 40% of the time in this portion of logs highlighted by this example.

This is repeated for every cluster/activity pair. Finally, each $Activity\_i \in L$ is mapped to a $Cluster\_j \in \mathcal{A}^F$ that has the highest matching percentage among all the identified clusters.

The remainder of this section presents *(i)* in Section 6.2.1 the process we followed to select the datasets included in the evaluation, and *(ii)* in Section 6.2.2 the results achieved.

6.2.1 Dataset selection

This section reports a preliminary analysis of the datasets widely employed by the smart space community. The list of the datasets that have been taken into account during this analysis is shown in Table 6.1. We considered only labeled datasets. Having activities labeled allows for activity-specific tasks such as recognition and prediction. In our case, labeling is used to make a comparison between the clusters identified by our methodology and the activities performed within the dataset. Furthermore, we are interested in the following features:

- *Sensors*: the kind of sensors used in the log.
- *Inhabitants*: the number of residents participating in the dataset. It is not always possible to establish which inhabitant is responsible for triggering a specific sensor.
- *Length*: the log size measured in terms of days of acquisition/number of sensor measurements influences the robustness of the clustering task.

From the list provided in Table 6.1, we excluded:

1. datasets from multi-user environments, i.e., datasets in which more than one inhabitant is involved (datasets 3, 4, 5, 7 and 10);
2. datasets that can be considered “small" in terms of days of acquisition/number of sensor measurements. In particular, we excluded those shorter than 3 months;

ID	Dataset	Sensors	Inhabitants	Length
1	[148]	3 PIR, 4 door, 1 flush, 2 pressure, 2 electric	1	14 days
2a	[176]	77 activation/deactivation sensors	1	16 days
2b	[176]	84 activation/deactivation sensors	1	16 days
3	[7]	6 photocells, 3 force, 3 contact, 6 distance, 1 IR, 1 temperature	2	30 days
4	[80]	6 PIR, 5 door, 1 temperature, 6 microphones, 5 cameras and accelerometers (for each subject)	10	-
5	[161]	IMUs and accelerometers attached to subjects (3 each), objects and devices	4	-
6	[194]	14 switch sensors on devices and doors	1	30 days
7	[13]	Smartphone accelerometers	30	-
8	[51]	About 200 device and environmental sensors	1	28 days
9	[42]	30 PIR, 4 door switches, 3 temperature	1	2 years
10	[170]	50 PIR plus device and environmental sensors	2	-

Table 6.1. List of the available and most widely used datasets for the smart space community. They have been identified with an increasing ID from 1 to 10 (datasets 2a and 2b have the same prefix because they are from the same project). For more information on these datasets see Table 2.2.

3. datasets with a poor distribution of PIR sensors within the environment.

With these considerations, from the list in Table 6.1, we choose the dataset identified as 10: the *aruba* dataset from CASAS project [44]. It is a state-of-the-art sensor log freely available at `http://casas.wsu.edu/datasets/`. It is a partially labeled sensor log containing real-life data from a smart home. Specifically, the smart home was inhabited by an adult woman. This resident moves in an environment with 31 motion sensors, 4 temperature sensors, and 4 sensors that detect and monitor the closing of doors. This dataset contains 6438 labeled activities. The layout of the *aruba* map can be seen in Figure 5.3.

The rationale behind this decision is also based on the significant influence that this project has had on the community as well as the accessibility of source code for algorithms frequently employed as benchmarks [54, 71, 125, 49, 98].

The *aruba* dataset includes eleven different labels: *Enter_home, Leave_home, Housekeeping, Meal_preparation, Eating, Wash_dishes, Relax, Resperate, Bed_to_toilet, Sleeping*, and *Work*. The *other* label is ignored as too generic.

We considered only measurements from PIR sensors in order to be able to use the VPM system [125, 129] to perform the *Log conversion* task (see Section 6.1.2).

Furthermore, we also decided to evaluate our approach on a synthetic dataset produced by using the smart space simulator [195] presented in Chapter 3. In particular, we produced the synthetic dataset by simulating a series of activities conducted by a single human, namely: *Cook_&_eat, Do_dishes, Drink, Eat_cold,* *Eat_warm, Exercise, Go_work, Sleep, Rest, Shop, Use_PC, Watch_tv* and *Wc.* We replicated the configuration of the *aruba* smart environment shown in Figure 5.3, including the same list of sensors and the same positioning, by using the design tools provided within the same simulator.

Other three datasets have also been considered for the evaluation, i.e., 2a, 2b, and 6 in Table 6.1, but results are not included as they did not suit our approach (see Section 6.2.4 for details).

6.2.2 Results

By following the evaluation methodology discussed in Section 6.2, we applied our approach to the *aruba* dataset from the CASAS project and to a synthetic dataset produced by our simulator (see Chapter 3). We repeated the evaluation using four different clustering algorithms: Spectral clustering, K-means, DBSCAN, and OPTICS.

First, we calculated the *session threshold* value Δ, which turned out to be 7 seconds for the CASAS dataset (with 8355 identified sessions) and 6 seconds for the synthetic one (with 946 identified sessions). Then, we performed a preliminary *hyperparameter* analysis. In particular, we selected, for each algorithm, the set of optimal parameters whose values are used to control the clustering process:

- **Spectral**: *random_state*=0, *assign_labels*=cluster_qr and *eigen_solver*=amg;
- **K-means**: *init*=k-means++, *algorithm*=elkan;
- **DBSCAN**: *eps* was set to 0.3 for the CASAS dataset and 0.24 for the synthetic dataset. *minPts*=2*dimensionality of the encoded log. Such dimensionality was 35 for the CASAS dataset and 18 for the synthetic one;
- **OPTICS**: *min_samples*=50, *xi*=0.05 and *min_cluster_size*=0.05.

All four algorithms performed better with the *duration-based encoding* (see Section 6.1.5). Then, the feature vector is normalized to ensure that the maximum absolute value of each feature in the training set becomes 1.0.

While with DBSCAN and OPTICS there is no need to choose the number of clusters a priori, for Spectral and K-means an estimation of the number of clusters must be provided. In these latter cases, we employed the *elbow method* to determine an optimal number of clusters to produce [101]. The estimation provided by the elbow method balances the number of clusters versus the error within the clusters. The error, in this context, refers to the average distances of points within a cluster from their respective centroids. We applied the elbow method and determined the result based on the knee of the elbow to define the number of clusters to create, which is explicitly required by all algorithms.

Note that the number of clusters does not need to be equal to the number of activities in the dataset used as a benchmark. From our experiments, we noticed that they could also be more with respect to the number of activities, with some of them remaining almost empty.

Finally, we computed a mapping between the activity labels in the dataset used as a benchmark and the clusters identified by our approach. Table 6.2a presents

CASAS Dataset Results

Activities	Spectral	K-means	DBSCAN	OPTICS
Bed_to_toilet	5	0	0	0
Eating	1	3	0	0
Enter_home	16	3	2	1
Leave_home	16	3	1	1
Housekeeping	9	3	1	0
Meal_prep.	19	3	1	0
Relax	9	3	0	1
Resperate	12	27	0	1
Sleeping	5	0	1	1
Wash_dishes	18	3	0	1
Work	16	23	0	0

(a)

Synthetic Dataset Results

Activities	Spectral	K-means	DBSCAN	OPTICS
Cook_&_eat	5	1	1	2
Do_dishes	0	7	3	3
Drink	5	1	1	2
Eat_cold	0	3	0	3
Eat_warm	0	1	1	2
Exercise	4	5	0	3
Go_work	8	1	0	2
Sleep	3	4	0	3
Rest	1	3	0	3
Shop	7	2	0	3
Use_PC	0	1	0	3
Watch_TV	4	5	0	3
WC	3	5	0	3

(b)

Table 6.2. Table (a) shows the clusters identified by applying the approach described in this paper to the CASAS *aruba* dataset. Table (b) shows the clusters identified by applying the approach described in this paper to the synthetic dataset produced by the simulator. In the first column, all the labeled activities available in the related dataset are listed, while on the first row is the list of algorithms used for the clustering task.

the results obtained for the *aruba* CASAS dataset: each column shows the clusters identified for each activity by each clustering algorithm. Similarly, Table 6.2b presents the results obtained for the synthetic dataset.

	CASAS	Synthetic
Our approach	98% (9h)	90% (4h)
[46]	81% (37h)	91% (13h)
[46] + pre-processing	96% (42h)	97% (16h)

Table 6.3. Accuracy results obtained by *(i)* the approach described in this work (first line), *(ii)* the state-of-the-art segmentation method, *(iii)* the same state-of-the-art method applied to the pre-processed log. In brackets, the time spent by each method in terms of hours.

6.2.3 Comparison with the state-of-the-art

To further evaluate our approach, we performed an empirical comparison with the state-of-the-art log segmentation method developed by [46] already introduced in Section 3.4.2. Their *Activity Learning* (AL) software tool (which can be freely downloaded from `http://casas.wsu.edu/tools/`) contains various dedicated components such as activity modeling and recognition (AR), activity discovery (AD), and activity prediction (AP). For our comparison, we used the AD component, which finds sequential patterns in time-ordered sensor data and computes a segmentation of the input dataset. Noteworthy, with respect to the state-of-the-art presented in Section 1.1, the comparison with [46] was the only option as other available methods have different inputs or outputs.

In addition, to simply evaluating the proposed approach, we were interested to see whether or not the pre-processing step from our approach (see Sections 6.1.2 and 6.1.3) could improve the final segmentation of the state-of-the-art method itself.

Therefore, we applied [46] and *pre-processed* [46] to the same datasets used to validate our approach: the *aruba* dataset from the CASAS project and the synthetic dataset produced by a smart home simulator (see Section 6.2.1). Then, considering the labeling provided by the benchmark datasets as our ground truth, we calculated the percentage of accuracy as the number of labels correctly assigned by each segmentation method out of the total number of assigned labels. Additionally, we kept track of the total time spent on each segmentation method. The experiments have been carried out on a Dell XPS 15 9500 notebook with an Intel Core i7-10750H and 16 GB of physical memory.

Comparison results are shown in Table 6.3 and will be discussed in the next section.

6.2.4 Discussion

The previous section reported the results obtained by applying the unsupervised approach described in this chapter to five different datasets. Such datasets are identified with IDs 2A, 2B, 6, and 10 in Table 6.1, plus a synthetic dataset produced using the simulation tool described in Chapter 3. As already mentioned in Section 6.2.1, three of the five datasets (namely 2A, 2B, and 6) are found to be unsuitable for our approach. For datasets 2A and 2B, the critical point is that PIR sensors are not homogeneously placed within the domestic environment and are mostly concentrated in the kitchen: 47 out of 77 total sensors for dataset 2A and 42 out of 84 for dataset 2B. This distribution does not allow for correctly abstracting the actions that take place in other areas of the house in the *Log Conversion* phase.

The dataset 6 has few sensor measurements: only 2180 labeled entries compared to the more than 792k of the *aruba* dataset. Consequently, the *Creation of Sessions*' phase of our approach generated few sessions, i.e., the number of sessions was less than the number of clusters suggested by the elbow method, making it impossible to apply a clustering algorithm in an optimal way.

Results in Table 6.2 show that the algorithm that performed best is *Spectral Clustering*. K-means does not always result in the best clustering results when pairs of clusters are not linearly separable. In human action clustering, this aspect has been highlighted in [100, 146, 205] where authors use spectral clustering and natural language techniques rather than employing compactness approaches such as K-means.

Analyzing the identified clusters, we can draw very interesting conclusions on the results obtained (see Table 6.2) through spectral clustering:

- The activities *Sleeping* and *Bed_to_toilet* are clustered together in Cluster 5. This result can be explained by looking at the layout of the house: the bathroom used during *Bed_to_toilet* is inside the bedroom used for *Sleeping*. In addition, there is only a single sensor in the bathroom (M004), which just registers the entrance and exit of the inhabitant from the bathroom itself. Then, no other specific action within the bathroom can be inferred, making both activities comparable in terms of actions.

- The activities *Enter_home, Leave_home*, and *Work* are clustered together in Cluster 16. This is justified by the fact that the inhabitant's workplace is located outside his house, so the movements registered by motion sensors to go to work are comparable to the ones related to entering and leaving the house for any other reason not specified in the labeling provided.

- The activities *Housekeeping* and *Relax* are clustered together in Cluster 9. It is plausible to think that the actions performed within the general activity *Housekeeping* cover most of the areas that the inhabitant uses to relax, e.g., the living room and the bedroom.

- All the other activities are clustered in well-separated clusters. The clusters not included in the mapping activity/cluster are mostly empty.

The results from spectral clustering for the synthetic dataset (shown in Table 6.2) also show very interesting characteristics. We still take into account the layout of the *aruba* dataset, conveniently recreated using the simulator's tools:

- The activities *Cook_and_eat* and *Drink* are clustered together in Cluster 5. Both activities take place in the same environment (i.e., the kitchen), making the movements required to perform them comparable.

- The activities *Do_the_dishes, Eat_cold*, and *Eat_warm* are clustered together in Cluster 0. Such activities take place in the same environment (i.e., the kitchen), and in particular, the kitchen sink is located in the middle between the oven, fridge, and microwave, all essential elements for the activities related to eating.

- The activity *Use_computer* is also clustered into Cluster 0. This is the only ambiguous result obtained from clustering since it cannot be associated with the other activities in the same cluster from the previous point.

- The activities *Exercise* and *Watch_TV* are clustered together in Cluster 4. As can be seen from the map, the exercise spot has been designed inside the living room near the chair used to watch the TV, making the activities very similar from the point of view of the motion sensors triggered during the simulation.

- All the other activities are clustered in well-separated clusters. The clusters not included in the mapping activity/cluster are mostly empty.

Regarding the comparison made with the state-of-the-art method (see Section 6.2.3), some interesting results emerge from Table 6.3.

The accuracy of our approach is superior, or highly comparable, to the results obtained by employing [46]. In particular, for the *aruba* dataset from the CASAS project, we reach a percentage of 98% against the 81% achieved by [46], while for the synthetic dataset, we reach a percentage of 90% against the 91% achieved by [46]. Concerning the time required by each method to complete the segmentation, our approach turns out to be four times faster, while for the synthetic dataset, our approach is three times faster.

Finally, it is also interesting to note how the application of our pre-processing step (see Sections 6.1.2 and 6.1.3) to the state-of-the-art methodology improved its final segmentation in terms of accuracy: for the *aruba* dataset, from 81% to 96% accuracy, while for the synthetic dataset, from 91% to 97% accuracy.

Chapter 7

User study

Performing a qualitative analysis alongside quantitative results in scientific research work is crucial for providing a comprehensive and nuanced understanding of the phenomena under investigation. While quantitative data offers measurable and numerical insights, qualitative analysis adds depth, context, and a more holistic perspective to the research findings. Here are several key reasons highlighting the importance of incorporating qualitative analysis in scientific research:

- qualitative analysis helps contextualize quantitative results by providing a deeper understanding of the factors influencing the observed patterns or trends;
- qualitative methods, such as interviews, open-ended surveys, or content analysis, enable researchers to capture rich and detailed information that may not be easily quantifiable;
- qualitative findings can be used to validate or challenge quantitative results, providing a more robust and trustworthy interpretation of the study.

In summary, the integration of qualitative analysis with quantitative results enhances the rigor and completeness of scientific research.

In [197], starting from the results obtained by the unsupervised segmentation methodologies introduced in Chapters 5 and 6, we mine the related process models and analyze them throughout a user study.

Process mining provides several mining approaches, each offering unique insights into the representation of underlying processes. In this user study we are interested in three *discovery* algorithms: *(i)* the inductive miner, *(ii)* the heuristic miner, and *(iii)* the fuzzy miner. Inductive process mining excels in capturing implicit knowledge, heuristic approaches leverage predefined rules and domain knowledge, while fuzzy mining accommodates uncertainty and imprecision in the data [84].

7.1 Mining process models through process discovery

In order to mine the habit and activity models from the segmentation results respectively obtained by applying the approaches described in Chapters 5 and 6, we have used the ProM software tool.

ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plugins. It is platform-independent as it is implemented in Java and can be downloaded free of charge (link in the footnote[1]). It provides the latest implementations of the known process mining algorithms, not only for the *discovery* of process models but also for checking the conformance of a model and/or for enhancing it [191].

For each habit interval identified by the segmentation methodology described in Chapter 5, the ProM framework was used to discover the related process models. In particular, four algorithms were used: *(i)* the inductive miner, *(ii)* the heuristic miner, *(iii)* the fuzzy miner, and *(iv)* the alpha miner. The models discovered with the alpha miner were excluded from the analysis because they were not relevant.

Similarly, for each activity identified by the segmentation methodology described in Chapter 6, the ProM framework was used to discover the related process models. The same four discovery algorithms were used. The models discovered with the alpha miner were excluded from the analysis because they were not relevant.

In Appendix, we propose some of the most relevant models mined from the ProM tool. All the other models are available in the repository in the footnote[2].

7.2 Designing the user study

Procedure. The user study was conducted following a questionnaire-based approach. The questionnaire was designed, created, and distributed to participants using the Google Forms platform.

Participants. Overall, a total of 20 different participants were involved in the user study. The age range was (on average) between 18 and 34 years and involved people with bachelor's degrees up to post-doc.

No previous knowledge was required to complete the questionnaire. However, before starting, it was recommended to read a brief handbook, which introduces the basic concepts and terminology for tackling the questionnaire. The handbook is available at the link in the footnote[3].

Questionnaire design. The questionnaire included 38 sections organized as follows:

- Section 1 contains a recommendation for reading the handbook before starting. The link to the handbook was provided.
- Section 2 were about user profiling, i.e., age and current position.
- Sections 3 to 20 are designed to provide feedback on the process models mined over the activity-based segmentation results from the approach described in Chapter 6. In particular, for each relevant activity, the related process model was mined by using three different *discovery* algorithms, i.e., *(i)* the inductive miner, *(ii)* the heuristic miner, and *(iii)* the fuzzy miner. Each output has been

[1]see: https://promtools.org/

[2]see: https://drive.google.com/drive/folders/1-nj4jSWd3I1LC1vnT1cYqKh5Yhnxj-vE?usp=sharing

[3]see: https://drive.google.com/file/d/1szyCCN_bM_OGo2iTI4IPQP0InbeL52-B/view

evaluated by visually inspecting the specific process model and by answering three questions:

- **Question 1**: "*How well do you think this model reflects the activity x?*", where x was the activity under analysis in that specific section. It was rated on a Likert scale ranging from 1 ("too generic") to 10 ("too specific"). Here, we wanted to have a high-level feedback on the model in its entirety.
- **Question 2**: "*Do the single actions in the model make sense with respect to a possible activity x?*", where x was the activity under analysis in that specific section. It was rated on a Likert scale ranging from 1 ("not at all") to 10 ("completely suitable"). Here, with respect to the previous question, we wanted to have a low-level feedback on the individual nodes (i.e., actions) of the model.
- **Question 3**: "*Do you have any comments about this model?*". It was an optional, open-ended question to collect further feedback on the model under observation.

- Sections 21 to 38 are designed to provide feedback on the process models mined over the habit-based segmentation results from the approach described in Chapter 5. In particular, for each habit, the related process model was mined by using three different *discovery* algorithms, i.e., *(i)* the inductive miner, *(ii)* the heuristic miner, and *(iii)* the fuzzy miner. Each output has been evaluated by visually inspecting the specific process model and by answering three questions:
  - **Question 1**: "*How well do you think this model reflects a possible daily human routine covering the time between START_TIME to END_TIME?*", where the time range was related to the habit under analysis in that specific section (e.g., from 05:15 AM to 7:00 AM). It was rated on a Likert scale ranging from 1 ("too generic") to 10 ("too specific"). Here, we wanted to have a high-level feedback on the model in its entirety.
  - **Question 2**: "*Do the single actions in the model make sense with respect to a possible daily human routine covering the time between START_TIME to END_TIME?*", where the time range was related to the habit under analysis in that specific section (e.g., from 05:15 AM to 7:00 AM). It was rated on a Likert scale ranging from 1 ("not at all") to 10 ("completely suitable"). Here, with respect to the previous question, we wanted to have a low-level feedback on the individual nodes (i.e., actions) of the model.
  - **Question 3**: "*Do you have any comments about this model?*". It was an optional, open-ended question to collect further feedback on the model under observation.

Questionnaire results. The feedback was collected in an Excel file and then analyzed with ad hoc statistical tools.

7.3 Statistical tools for qualitative analysis

Statistical tools play a fundamental role in the analysis and interpretation of data collected through a user study. Two widely used tools for comparing means and exploring group differences are the t-test and *analysis of variance* (ANOVA). These methods are fundamental in hypothesis testing and are essential in various fields, from experimental sciences to social research. In particular:

7.3.1 t-test

t-test is a statistical method employed for assessing the significance of differences between the means of two groups. It encompasses several variants, with the independent two-sample t-test being a common choice. For comparing the means (μ_1 and μ_2) of two independent groups with sample sizes n_1 and n_2, the t-statistic is computed as:

$$t = \frac{(\bar{X}_1 - \bar{X}_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{7.1}$$

where $\bar{X}_1$ and $\bar{X}_2$ are the sample means, s_1 and s_2 are the sample standard deviations, and n_1 and n_2 are the respective sample sizes.

The t-test assesses the null hypothesis that the means of the two groups are equal, and the test statistic follows a t-distribution with $n_1 + n_2 - 2$ degrees of freedom.

7.3.2 Analysys of variance (ANOVA)

ANOVA is a statistical method widely employed in scientific research to investigate the impact of multiple independent variables on a dependent variable. It is designed to partition the total variability observed in a dataset into distinct components attributable to different sources.

For a 1-way ANOVA, considering k groups, the total sum of squares (SST) can be expressed as:

$$SST = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (X_{ij} - \bar{X})^2 \tag{7.2}$$

where X_{ij} represents the j-th observation in the i-th group, $\bar{X}$ is the overall mean, and n_i is the sample size of the i-th group.

ANOVA then decomposes SST into two components: the sum of squares between groups (SSB) and the sum of squares within groups (SSW). The calculations are as follows:

$$SSB = \sum_{i=1}^{k} n_i (\bar{X}_i - \bar{X})^2 \tag{7.3}$$

$$SSW = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^2 \tag{7.4}$$

The mean square between groups (MSB) and mean square within groups (MSW) are obtained by dividing SSB and SSW by their respective degrees of freedom, leading to the F-ratio:

$$F = \frac{MSB}{MSW} \tag{7.5}$$

ANOVA tests the null hypothesis that all group means are equal, using the F-statistic. A significant F-value implies that at least one group mean differs from the others.

7.3.3 Choice of statistical tool by design

Both the t-test and ANOVA share the common objective of assessing differences in means, but they are applied in different contexts. The t-test is suitable for comparing two groups directly, while ANOVA extends the analysis to multiple groups simultaneously, making it advantageous in experimental designs with more complex structures. Additionally, ANOVA assesses overall group differences, but it does not pinpoint which specific groups are different. In contrast, the t-test directly informs about the difference between two specified groups.

In summary, the choice between ANOVA and the t-test hinges on the experimental design: ANOVA for multiple group comparisons and the t-test for focused examination of differences between two groups. Understanding the nuances of these statistical tools empowers researchers to make informed decisions in designing and interpreting their experiments.

In our case study, we want to analyze the process models obtained from different mining algorithms, i.e., *inductive*, *heuristic*, and *fuzzy* miners. For each algorithm, we identify an observation group. Therefore, having more than two groups, our choice falls under 1-way ANOVA.

7.4 Results and discussion

As described in Section 7.2, for each process model, two questions were asked: the first relating to the global model (high-level analysis) and the second specific to the nodes (i.e., human actions) included in the model (low-level analysis). Then, we ran two separate ANOVA tests, one for each question. The results are respectively shown in Table 7.1 and Table 7.2.

These results revealed that there is a 100% chance that at least one algorithm has a significant difference in mean scores. In addition, post-hoc tests have been conducted to explore pairwise differences between the three discovery algorithms under observation, i.e., fuzzy, heuristic, and inductive.

These additional pairwise tests show that process models mined using the *fuzzy* algorithm are considered more suitable for this type of modeling of human behavior, both at a high-level (i.e., question 1) and at a low-level (i.e., question 2). The mean scores obtained from the questionnaire, shown in the bar charts in Figure 7.1, further highlight the participants' preferences. Human behavior is flexible by nature, and this characteristic is considered by fuzzy mining, which takes into account the uncertainty and imprecision of the data [84].

Source	df	SS	MS	F	p-value
Factor (Between Groups)	2	1417.143	708.571	198.90	9.335E-69
Error (Within Groups)	681	2,426	4		
Total	683	3843.169			
F critical Value					
3.008949291					

(a)

	Fuzzy	Heuristic	Inductive	Total
Mean	7.557	4.109	5.192	5.619
Standard Deviation	1.791	2.144	1.696	2.372
Variance	3.208	4.600	2.878	5.626
t-critical	1.970	1.970	1.970	
Margin	1.283	2.084	1.467	
Grand Mean	5.619			
SS Total	3,843.2			
Sum of Squares Factor	855.567	520.023	41.551	
SS Factor	1417.143			
SS Error	2,426.03			

(b)

Table 7.1. Table (a) shows the results obtained by performing the 1-way ANOVA test on the feedback relating to the activity or habit model in its entirety (i.e., question 1). Table (b) shows the relevant calculations made to calculate these results.

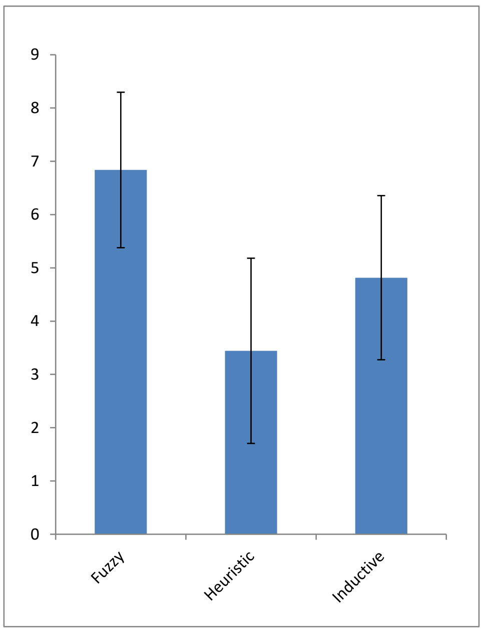
Source	df	SS	MS	F	p-value
Factor (Between Groups)	2	1329.774	664.887	212.82	1.58914E-72
Error (Within Groups)	681	2,128	3		
Total	683	3457.292			
F critical Value 3.008949291					

(a)

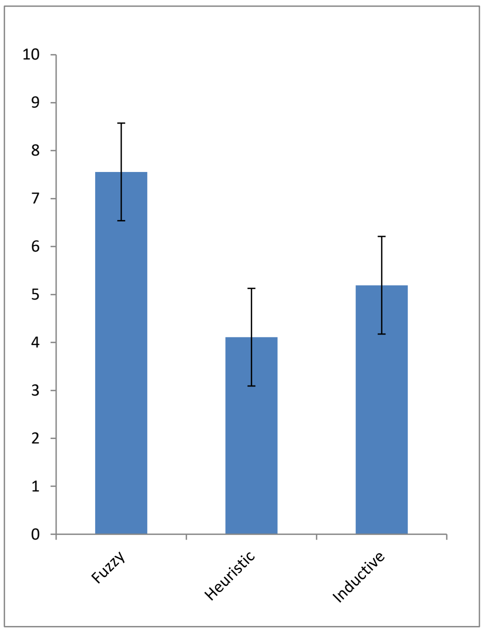
	Fuzzy	Heuristic	Inductive	Total
Mean	7.557	4.109	5.192	5.619
Standard Deviation	1.791	2.144	1.696	2.372
Variance	3.208	4.600	2.878	5.626
t-critical	1.970	1.970	1.970	
Margin	1.283	2.084	1.467	
Grand Mean	5.619			
SS Total	3,843.2			
Sum of Squares Factor	855.567	520.023	41.551	
SS Factor	1417.143			
SS Error	2,426.03			

(b)

Table 7.2. Table (a) shows the results obtained by performing the 1-way ANOVA test on the feedback related to the specific actions included in the activity or habit model under observation (i.e., question 2). Table (b) shows the relevant calculations made to calculate these results.



(a)



(b)

Figure 7.1. Bar charts showing the mean scores of the responses to the various algorithms under analysis. Figure (a) shows the scores for question 1, while Figure (b) shows the scores for question 2.

Chapter 8

Conclusions and future work

In this thesis, we discussed the application of process discovery techniques to smart space data, focusing on smart home environments, and how to handle the main identified challenges in the field [124]: *(i)* generating data in an appropriate format to be able to apply PM techniques; *(ii)* abstracting the gap between sensor and event logs; *(iii)* choosing or designing the proper modeling formalism for representing human behavior; *(iv)* segmenting logs into traces to be able to apply PM techniques; *(v)* dealing with multi-user environments; and *(vi)* addressing the continuous evolution of human behavior.

- In Chapter 2, we have conducted a literature review on the application of process discovery techniques to smart space data. A total of 25 studies were included and analyzed in the survey. The results showed that there are already some potential solutions for these challenges, ranging from sensor measurements to activities and sometimes going a step further by identifying habits.

  However, some important issues still need to be addressed, such as the selection of an appropriate modeling formalism for human behavior mining, the exploitation of context information, the generalisability of the developed techniques, the identification of user individual and collaborative activities in multi-user environments, and discovery techniques that can deal with user behavior evolution without being intrusive for the users. In this thesis, we dealt with some of these main challenges.

- In Chapter 3, we presented a simulator for smart homes based on process models of human activities expressed using the behavior pattern formalism. In addition, we propose a common way to represent produced datasets by leveraging a standard successfully employed in the field of business process logs, i.e., the XES format. Such a format is then further extended with the work described in Chapter 4.

  The simulator allows to generate data for multiple independent virtual inhabitants. The proposed tool can be configured to support any of the features of freely available datasets and can be used to define challenges and benchmarks addressing different sets of sensors and environmental layout and conditions. In the evaluation section, we proved that the tool was able to produce datasets with characteristics similar to those of freely available datasets and that the produced datasets can be used to evaluate state-of-the-art techniques producing results similar to those obtained with a real dataset. Additionally, our simulator can produce a dataset including both the sensor level and the action level, which is something unmatched in available datasets.

At the current stage, the simulator is not able to simulate collaborative activities between different inhabitants, but they are only addressed in a very limited number of works in the scientific literature. In addition, the simulator is only able to model sensors attached to the environment because the modeling cost of body-worn sensors is too high. This can be considered a minor limitation, as these latter are mainly employed to recognize particular human movements (e.g., falling), and replicating experiments is relatively simple with respect to those requiring a whole environment.

While modeling human activities using behavior patterns is simple and immediate, modeling sensors and devices inside the simulator may not be straightforward. One of our aims is to promote the employment of the simulator public repository in the community as a facility to collaborate by proposing new models that can then be combined in order to ease the generation of freely available datasets.

- In Chapter 4, we presented a new format for IoT-enhanced event logs. This format combines the different event types presented in [26] with the notion of object from OCEL to form a very flexible format capable of integrating IoT data with process data with minimal information loss. The format was evaluated, and we provided evidence of its theoretical robustness and practical usefulness. Finally, we also compared our new format with existing alternatives and showed that it is the only one fulfilling the requirements in [27]. Moreover, a first practical application demonstrated how generating a log following our format enables process analyses integrating IoT data. Such a log has been produced with the smart home simulator introduced in Chapter 3.

  Some limitations of our work include the still-experimental implementation so far and the potential amplifying effect of multiple event abstraction steps on sensor data quality issues. To cope with these limitations, in future works, we first plan to further develop the format by creating more tool support to generate and manipulate logs. Next to this, we would like to design new analysis techniques taking advantage of the capabilities of the new format, e.g., for IoT-enhanced trace clustering and IoT-enhanced predictive process monitoring. As such, looking into the compatibility of NICE logs with the OCEL 2.0 format could help leverage existing techniques and tools.

- In Chapter 5, we have introduced an approach to automatically segmenting a sensor log into human habits by defining a discretization strategy based on metrics from the Petri nets automatically discovered at each merging step. Then, the final segmentation can be used to mine process models describing human habits that can, in turn, be employed to make decisions about the environment.

  Furthermore, in Chapter 6, we have introduced another unsupervised approach, but this time we focus on activities instead of habits, which allows for finer-grained control over human routines. We proposed a clustering-based strategy that groups together actions performed by humans in the context of a smart space. The approach has been evaluated against five state-of-the-art datasets, four out of five based on real scenarios, and an additional synthetic dataset produced by the smart home simulator introduced in Chapter 3. The approach was also compared with a state-of-the-art log segmentation method, obtaining interesting results in terms of accuracy and execution time.

  At their current stage, these two proposed solutions have some limitations. They need a sensor log to be properly converted into an action log $\mathcal{A}$. In order to perform such a conversion task, we applied the technique described in [125], which only supports Presence InfraRed sensors (PIRs), a frequent option as discussed in Section 2.1.1. By only using this kind of sensor, we limit the actions that can be recognized. For instance, in the smart house used within the context of the dataset *aruba*, only a sensor is placed on the bathroom, specifically on the door (namely M004, as shown in Figure 5.3). From that sensor, we can only gather information about the inhabitant entering or exiting the bathroom; it is not possible to get the details of the actions performed inside it (e.g., wash hands, use the toilet, get a shower).

  However, despite this limitation, it is still possible to obtain equivalent information (i.e., the detection of a subject in a certain area) from other types of sensors, making the approach scalable and applicable to a wider range of cases. For example, a video camera or a laser detection sensor (LIDAR) can derive the same information, i.e., the presence of a subject within their range of vision.

  Furthermore, our approaches have been evaluated solely against single-human datasets. As discussed in Section 2.1.5, this is a quite common limitation in the field that limits the relevancy of solutions. In multi-human environments, it is far harder to associate every measure within the log with the correspondent resident that has triggered it, especially if we only consider PIRs, and thus it is even more difficult to segment the activities performed by every single user within the smart environment. Future work will concentrate efforts in this direction.

- Finally, in Chapter 7, the quantitative results already obtained in Chapters 5 and 6 have been enhanced with a qualitative analysis with an ad hoc statistical tool, i.e., the analysis of variance (ANOVA) technique. The results show that the fuzzy miner is considered the most suitable discovery algorithm for modeling human behavior (i.e., activities and habits), and this difference compared to other algorithms is statistically significant.

However, some important issues still need to be addressed in future work.

Research agenda and connection with process mining

Strictly related to the results achieved in the survey introduced in Section 2.1, interesting points and future directions arise. First of all, the study of the best modeling formalism for human behavior is to be continued [58], as many different languages are used and some languages showing potentially useful characteristics have seldom been used yet. For instance, in the survey, declarative modeling formalisms, which often provide richer constructs, have only been used by S8 [175], who specifically highlighted the added value of DECLARE models for unstructured or flexible behavior. Other declare models, such as CMMN [2], could be as well studied, as it forms a standardized specification providing the possibility to specify control flow and corresponding rule-based constraints [202] useful for indicating context-dependent behavior. The choice of formalism may need to be adapted to the specific application, and transformations between formalisms may also be a viable option to meet diverse needs (understandability, actionability, expressiveness, flexibility, etc.). In addition, the use of contextual information to create more meaningful models should be further explored, as it largely remains unexplored.

Models specifically proposed for representing IoT-enhanced process models (see [187]) could also be explored, although the mining of these models is still a big challenge. The NICE format introduced in Chapter 4 can be used as a basis for further developments in this direction. This also raises the question of the algorithm that should be used to discover models of human behavior. A benchmark could also help to assess to what extent existing algorithms are able to address the challenges of human behavior and whether new algorithms are required. With the user study discussed in Chapter 7, we investigated in this direction.

As highlighted by the studies included in the survey in Chapter 2, another relevant issue is the frequent employment of a restricted list of datasets. In the first place, the scarce availability of datasets in other domains led researchers to mainly focus on the home environment. In second place, a large portion of them is evaluated against a single dataset from the smart space community (see Table 2.1), and in several cases, important information on the datasets is not even provided (see Table 2.2). While the use of a common dataset makes it easier to compare the different methods, it might make some of the techniques less generalizable to other data and other environments. In addition, to properly evaluate and compare the different approaches, the datasets used should be properly described.

Generalization remains one of the main issues to be solved in future research. In particular, general approaches are needed to enable their application independently of the type of sensors used, or at least not sensitive to the specific dataset. As a consequence, the introduction of validated benchmarks could be of benefit to the entire community, as happened in other communities [66]. In this sense, simulators such as the one presented in Chapter 3 can also be employed to generate datasets that can be used to develop and validate process mining techniques for different kinds of smart spaces and types of sensors. Validation through both real and synthetic data can generate interesting results and insights.

Concerning the gap between sensor logs and event logs, some progress has been made. However, in the vast majority of cases, proposed approaches are not generic, being instead sensitive to the specific sensors employed. In addition, the right level of granularity at which sensor measurements should be aggregated into events [207], is still a hot source of debate [56] and a huge research challenge to tackle [20]. The research community, in particular, is currently focusing on *(i)* the specific goals of event abstraction, *(ii)* the consequent loss of information, *(iii)* the difficulty, or even impossibility, of creating general purpose techniques, and *(iv)* on the kind of technology needed.

Automated segmentation of logs is also an area where many research efforts are needed. Techniques usually employed usually focus on classifying specific point in time as segmentation points, whereas data mining techniques could be employed at the sequence level as proposed in [46, 120]. The methodologies discussed in Chapters 5 and 6 push in this direction.

Support for multiple users without the employment of body-worn devices is currently a neglected issue in research. This applies not only to the smart space but also to the community research area at the intersection between process mining and IoT [75]. If the employment of active methods for tracking (e.g., wearing a beacon bracelet) can be considered acceptable in working scenarios, it is usually considered annoying or unfeasible in other scenarios (e.g., smart houses, public spaces). In order to track multiple users in a completely passive way, though, more complex devices are needed (e.g., cameras, grids of presence infrared sensors), making setups expensive or raising concerns with respect to privacy issues.

As human behavior is inherently mutable over time, future research should focus more on detecting deviations from discovered human processes, for example, by employing conformance checking techniques and acquiring new behaviors if deviations are frequent [157].

Notably, the efforts needed to solve the proposed research agenda are not isolated from research in process discovery and, more generally, process mining. In the following, we will discuss this aspect with respect to the research questions identified in Section 2.1:

- **RQ-1**: a review of the different available process modeling formalisms and, more specifically, of the processes where the human component is fundamental is reviewed in [57, 6]. Even though, as witnessed by recent publications, the introduction of new process modeling formalisms is not a priority for the community, the automatic discovery of such processes is a very active field [15]. Particularly of interest for the world of smart spaces is the ability to mine decisions [138] in order to make models suitable for automated enactment.

- **RQ-2**: the problem of bridging the gap between sensor events and process events is strongly connected with the problem of providing a view of the processes at multiple levels, as introduced in [20]. As a consequence, the problem can be seen as general for the process mining community. The review proposed in [207] describes the various techniques for event abstraction proposed in the PM literature so far. This being said, unsupervised event abstraction techniques for PM were recently shown to have limitations [192]. This thesis corroborates the importance given to user knowledge for event abstraction in IoT PM and might indicate that future research should focus more on supervised event abstraction techniques. A study to evaluate the performance of unsupervised event abstraction specifically on sensor logs could help orient further research.

- **RQ-3**: the problem of splitting the logs into traces is quite specific for the smart space world, where processes are derived from raw sensor data [109]. While classical process mining deals with logs with well-defined traces, the problem of deriving traces from sensor data can be connected to the research on trace clustering [70]. Another related line of research is case correlation, which consists of trying to group together process events belonging to the same instance when no case ID is logged at runtime. For instance, in [152], authors propose an approach for case correlation based on how frequently activities follow each other and how close they are temporally. Such techniques may also be useful to find a logical case notion in sensor logs.

- **RQ-4**: as for **RQ-3**, process mining usually deals with logs where resources are first-class citizens, and therefore the problem of associating events to specific resources is usually neglected. Nevertheless, a field of PM research that could be relevant for this challenge is organizational mining, which looks into how work is passed between different actors in a process (e.g., in [10]).

- **RQ-5**: process enhancement and adaptation in the long term is a classical task of process mining. This is very much related to research in conformance checking [69] and concept drift [162].

Applying deep learning to smart space data: overview and opportunities

Most of the studies that rely on machine learning techniques for tasks like human activity recognition (HAR) heavily depend on statistical methods [31], symbolic representation [131], and time-frequency transformation [93] for feature extraction. The features extracted are carefully pre-processed and engineered. To efficiently collect and capture the flexible features of human behavior, there were no uniform or systematic extraction techniques.

In recent years, deep learning (DL) techniques have had remarkable success in modeling high-level abstractions from complex data across a wide range of domains, including computer vision, speech processing, and natural language processing (NLP) [153]. Studies such as [87, 113, 204] investigated the efficacy of deep learning applied to smart space contexts.

One of the appealing promises of DL is replacing the manually selected features with efficient unsupervised or semi-supervised feature learning and hierarchical feature extraction algorithms. Deep models architectures enable scalable learning from basic to abstract features. Furthermore, deep models have the capacity to learn descriptive characteristics from complex data thanks to modern computing resources like GPUs. The activity recognition system's exceptional learning capacity also makes it possible for it to thoroughly examine multi-modal sensory data in order to provide precise recognition.

Deep neural networks have a variety of architectures that encode characteristics in different ways. *Convolutional neural networks* (CNNs), for instance, are capable of capturing the local connections of multi-modal sensory data, and accurate recognition is achieved through the translational invariance that locality introduces [89]. *Recurrent neural networks* (RNNs) are suitable for processing streaming sensory data in the identification of human activity because they extract temporal dependencies and gradually learn information across time intervals.

Given their detachability and flexibility in composition, deep neural networks can be unified into networks with a single optimization function. This allows for the integration of various deep learning techniques, such as deep transfer learning [5], deep active learning [83], deep attention mechanisms [143], and other non-systematic but nonetheless efficient solutions [94, 139]. These methods have been used in works that address a variety of deep learning difficulties [39].

There are several research directions that merit further consideration in order to develop the full potential of deep learning applied to smart space environments:

- *Unsupervised learning*: deep learning models used for human activity recognition (HAR) are mainly used to extract features but are unable to classify activities when there is no ground truth provided. HAR requires an amount of labeled data as ground truth to train the models. Unsupervised learning methodologies can mitigate this fundamental requirement.

  One potential method for inferring labels in an unsupervised manner is to search for other knowledge. Once a lot of effort has been put into developing a model on a supervised and controlled dataset, *unsupervised transfer learning* (UTL) [21] approaches can be employed to make the known model appliable to as many other applications as possible. Another solution could be to use methods based on structured data such as ontology [159].

- *Standardization of the state-of-the-art*: as pointed out in [39], although several works are available in the field of deep learning applied to sensor-based environments, a true standardization of the state-of-the-art for a fair comparison is lacking. Evaluation metrics vary from study to study. Dividing data into training and test sets is a fundamental aspect of deep learning and influences the final result. Thus, defining a standard represents an urgent challenge in the field. A literature review that extends the work already done and summarizes the current situation can act as a guideline for future developments in the field.

- *Multi-user environments*: unlike single-user environments, many real-life environments consist of multiple users performing activities, in some cases even concurrently or collaboratively. In this context, it becomes even harder to carry out tasks such as HAR and activity prediction. Activity prediction is an extension of HAR in which the system aims to predict the next activity in advance. In [104], authors use a *long short-term memory* (LSTM) network to perform activity prediction. Such an approach relies on a *word-embedding* technique that is typically used for natural language processing (NLP) to map words into a vector space. By using a similar embedding approach in a multi-user smart space, they map activities to vector coordinates in a vector space. Furthermore, algorithms of *intention recognition* based on brain signals [209] can assist the prediction task.

  The authors of [104] conclude that by considering other contextual information, such as location information, the accuracy of the results could improve. In the approaches described in Chapters 5 and 6, we heavily rely on user location information. Combining these methodologies could lead to interesting results. It will be considered in our future work.

Bibliography

[1] Mysphera enterprise, rtls sphera indoor positioning system, http://mysphera.com (Cited on page 25)

[2] Omg: Case management model and notation 1.1 (2016) (Cited on page 90)

[3] van der Aalst, W.e.a.: Process mining manifesto. In: BPM'11. pp. 169–194 (2011) (Cited on page 8)

[4] van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, 2 edn. (2016) (Cited on pages vi and 8)

[5] Akbari, A., Jafari, R.: Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. In: Proceedings of the 18th International Conference on Information Processing in Sensor Networks. pp. 85–96 (2019) (Cited on page 93)

[6] Aldin, L., de Cesare, S.: A literature review on business process modelling: new frontiers of reusability. Enterprise Information Systems 5(3), 359–383 (2011) (Cited on page 91)

[7] Alemdar, H., Ertan, H., Incel, O.D., Ersoy, C.: Aras human activity datasets in multiple homes with multiple residents. In: 2013 7th Intl. Conf. on Pervasive Computing Technologies for Healthcare and Workshops. pp. 232–235. IEEE (2013) (Cited on page 73)

[8] Alshammari, N., Alshammari, T., Sedky, M., Champion, J., Bauer, C.: Openshs: Open smart home simulator. Sensors 17(5), 1003 (2017) (Cited on page 42)

[9] Alsheikh, M.A., Selim, A., Niyato, D., Doyle, L., Lin, S., Tan, H.P.: Deep activity recognition models with triaxial accelerometers. In: Workshops at the Thirtieth AAAI Conference on Artificial Intelligence (2016) (Cited on page 6)

[10] Alvarez, C., Rojas, E., Arias, M., Munoz-Gama, J., Sepúlveda, M., Herskovic, V., Capurro, D.: Discovering role interaction models in the emergency room using process mining. Journal of biomedical informatics 78, 60–77 (2018) (Cited on page 92)

[11] America, S.: Is there a reproducibility crisis in science? Nature Video, 28 May 2016, https://www.scientificamerican.com/video/

is-there-a-reproducibility-crisis-in-science/ (Cited on pages vi and 30)

[12] Ampatzoglou, A., Bibi, S., Avgeriou, P., Chatzigeorgiou, A.: Guidelines for managing threats to validity of secondary studies in software engineering. In: Contemporary Empirical Methods in Software Engineering, pp. 415–441 (2020) (Cited on page 29)

[13] Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: Esann. vol. 3, p. 3 (2013) (Cited on page 73)

[14] Annett, J.: Hierarchical task analysis. Handbook of cognitive task design 2, 17–35 (2003) (Cited on page 31)

[15] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: review and benchmark. IEEE transactions on knowledge and data engineering 31(4), 686–705 (2018) (Cited on page 91)

[16] Augusto, J.C., Nugent, C.D.: The use of temporal reasoning and management of complex events in smart homes. In: ECAI. vol. 16, p. 778 (2004) (Cited on page 3)

[17] Aztiria, A., Augusto, J.C., Basagoiti, R., Izaguirre, A., Cook, D.J.: Discovering frequent user–environment interactions in intelligent environments. Personal and Ubiquitous Computing 16(1), 91–103 (2012) (Cited on pages 3 and 7)

[18] Aztiria, A., Izaguirre, A., Basagoiti, R., Augusto, J.C., Cook, D.J.: Automatic modeling of frequent user behaviours in intelligent environments. In: 2010 IE. pp. 7–12 (2010) (Cited on page 27)

[19] Banham, A., Leemans, S.J., Wynn, M.T., Andrews, R., Laupland, K.B., Shinners, L.: xpm: Enhancing exogenous data visibility. AI in medicine 133, 102409 (2022) (Cited on page 48)

[20] Beerepoot, I., Di Ciccio, C., Reijers, H.A., Rinderle-Ma, S., Bandara, W., Burattin, A., Calvanese, D., Chen, T., Cohen, I., Depaire, B., et al.: The biggest business process management problems to solve before we die. Computers in Industry 146, 103837 (2023) (Cited on pages vii, 12, 90, and 91)

[21] Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. In: Proceedings of ICML workshop on unsupervised and transfer learning. pp. 17–36. JMLR Workshop and Conference Proceedings (2012) (Cited on page 93)

[22] Bernasconi, E., Boccuzzi, M., Briasco, L., Catarci, T., Ghignoli, A., Leotta, F., Mecella, M., Anna, M., Nina, S., Veneruso, S.V., et al.: Notae: Not a written word but graphic symbols. In: CEUR WORKSHOP PROCEEDINGS. vol. 3144, pp. 1–7 (2022) (Cited on page xiv)

[23] Bernasconi, E., Boccuzzi, M., Catarci, T., Ceriani, M., Ghignoli, A., Leotta, F., Mecella, M., Monte, A., Sietis, N., Veneruso, S., et al.: Exploring the historical context of graphic symbols: the notae knowledge graph and its visual interface. In: CEUR WORKSHOP PROCEEDINGS. vol. 2816, pp. 147–154. Dennis Dosso, Stefano Ferilli, Paolo Manghi, Antonella Poggi, Giuseppe Serra ... (2021) (Cited on page xiv)

[24] Bertrand, Y., Van den Abbeele, B., Veneruso, S., Leotta, F., Mecella, M., Serral, E.: A survey on the application of process discovery techniques to smart spaces data. EAAI 126, 106748 (2023) (Cited on pages vii, viii, xi, and 48)

[25] Bertrand, Y., Van den Abbeele, B., Veneruso, S., Leotta, F., Mecella, M., Serral Asensio, E.: A survey on the application of process mining to smart spaces data. Lecture Notes in Business Information Processing (2022) (Cited on page xi)

[26] Bertrand, Y., De Weerdt, J., Serral, E.: A bridging model for process mining and iot. In: International Conference on Process Mining. pp. 98–110 (2021) (Cited on pages vii, 47, 48, 49, 51, 56, and 88)

[27] Bertrand, Y., De Weerdt, J., Serral, E.: Assessing the suitability of traditional event log standards for iot-enhanced event logs. In: International Conference on Business Process Management. pp. 63–75 (2022) (Cited on pages vii, 47, 48, 52, 56, 57, and 88)

[28] Bertrand, Y., Veneruso, S., Leotta, F., Mecella, M., Serral Asensio, E.: Nice: The native iot-centric event log model for process mining. Lecture Notes in Business Information Processing (Cited on pages viii and xi)

[29] Bono-Nuez, A., Blasco, R., Casas, R., Martín-del Brío, B.: Ambient intelligence for quality of life assessment. Journal of Ambient Intelligence and Smart Environments 6(1), 57–70 (2014) (Cited on page 1)

[30] Bose, R.J.C., Verbeek, E.H., van der Aalst, W.M.: Discovering hierarchical process models using prom. In: IS Olympics: Information Systems in a Diverse World: CAiSE Forum 2011, London, UK, June 20-24, 2011, Selected Extended Papers 23. pp. 33–48. Springer (2012) (Cited on pages 18 and 19)

[31] Brophy, E., Veiga, J.J.D., Wang, Z., Ward, T.E.: A machine vision approach to human activity recognition using photoplethysmograph sensor data. In: 2018 29th Irish Signals and Systems Conference (ISSC). pp. 1–6. IEEE (2018) (Cited on page 92)

[32] Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., Zaccaria, R.: Analysis of human behavior recognition algorithms based on acceleration data. In: 2013 IEEE International Conference on Robotics and Automation. pp. 1602–1607. IEEE (2013) (Cited on pages 18 and 19)

[33] Brzychczy, E., Trzcionkowska, A.: Process-oriented approach for analysis of sensor data from longwall monitoring system. In: International Conference on

Intelligent Systems in Production Engineering and Maintenance. pp. 611–621. Springer (2018) (Cited on page 9)

[34] Cameranesi, M., Diamantini, C., Mircoli, A., Potena, D., Storti, E.: Extraction of user daily behavior from home sensors through process discovery. IEEE IoT Journal 7(9), 8440–8450 (2020) (Cited on page 18)

[35] Cameranesi, M., Diamantini, C., Potena, D.: Discovering process models of activities of daily living from sensors. In: BPM'17. pp. 285–297 (2017) (Cited on page 18)

[36] Carolis, B.D., Ferilli, S., Mallardi, G.: Learning and recognizing routines and activities in sofia. In: European Conference on Ambient Intelligence. pp. 191–204 (2014) (Cited on pages 18 and 19)

[37] Carolis, B.D., Ferilli, S., Redavid, D.: Incremental learning of daily routines as workflows in a smart home environment. ACM TiiS 4(4), 1–23 (2015) (Cited on page 18)

[38] Chan, H., Perrig, A.: Security and privacy in sensor networks. computer 36(10), 103–105 (2003) (Cited on page 28)

[39] Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., Liu, Y.: Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. ACM CSUR 54(4), 1–40 (2021) (Cited on pages 6 and 93)

[40] Chen, L., Hoey, J., Nugent, C.D., Cook, D.J., Yu, Z.: Sensor-based activity recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42(6), 790–808 (2012) (Cited on page 5)

[41] Cook, D., Das, S.: Smart Environments: Technology, Protocols and Applications. Wiley-Interscience (2005) (Cited on page 1)

[42] Cook, D.J.: Learning setting-generalized activity models for smart spaces. IEEE intelligent systems 2010(99), 1 (2010) (Cited on pages 44 and 73)

[43] Cook, D.J.: Learning setting-generalized activity models for smart spaces. IEEE intelligent systems 27(1), 32 (2012) (Cited on pages 18, 19, and 21)

[44] Cook, D.J., Crandall, A.S., Thomas, B.L., Krishnan, N.C.: Casas: A smart home in a box. Computer 46(7), 62–69 (2012) (Cited on pages 18, 19, 22, 26, and 73)

[45] Cook, D.J., Crandall, A.S., Thomas, B.L., Krishnan, N.C.: Casas: A smart home in a box. Computer 46(7), 62–69 (2013) (Cited on page 44)

[46] Cook, D.J., Krishnan, N.C., Rashidi, P.: Activity Discovery and Activity Recognition: A New Partnership. IEEE Transactions on Cybernetics 43(3), 820–828 (2013) (Cited on pages ix, 3, 4, 5, 26, 67, 76, 78, and 91)

[47] Cook, D.J., Schmitter-Edgecombe, M.: Assessing the quality of activities in a smart environment. Methods of information in medicine 48(05), 480–485 (2009) (Cited on pages 18 and 19)

[48] Cook, D., Augusto, J., Jakkula, V.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4), 277–298 (2009) (Cited on page v)

[49] Crandall, A., Cook, D.J.: Learning activity models for multiple agents in a smart space. In: Handbook of Ambient Intelligence and Smart Environments, pp. 751–769. Springer (2010) (Cited on pages 44 and 73)

[50] Crandall, A.S., Cook, D.J.: Coping with multiple residents in a smart environment. Journal of Ambient Intelligence and Smart Environments 1(4), 323–334 (2009) (Cited on page 44)

[51] Cumin, J., Lefebvre, G., Ramparany, F., Crowley, J.L.: A dataset of routine daily activities in an instrumented home. In: Intl. Conf. on Ubiquitous Computing and Ambient Intelligence. pp. 413–425. Springer (2017) (Cited on page 73)

[52] Dahmen, J., Cook, D.: Synsys: A synthetic data generation system for healthcare applications. Sensors 19(5), 1181 (2019) (Cited on page 42)

[53] Davies, M., Callaghan, V.: iworlds: Generating artificial control systems for simulated humans using virtual worlds and intelligent environments. Journal of Ambient Intelligence and Smart Environments 4(1), 5–27 (2012) (Cited on page 42)

[54] De-La-Hoz-Franco, E., Ariza-Colpas, P., Quero, J.M., Espinilla, M.: Sensor-based datasets for human activity recognition–a systematic review of literature. IEEE Access 6, 59192–59210 (2018) (Cited on pages 44, 64, and 73)

[55] Degeler, V., Lazovik, A., Leotta, F., Mecella, M.: Itemset-based mining of constraints for enacting smart environments. In: 2014 PERCOM WORKSHOPS. pp. 41–46. IEEE (2014) (Cited on page 4)

[56] Depaire, B., Fahland, D., Leotta, F., Lu, X.: Third international workshop on event data and behavioral analytics (edba'22) (Cited on page 90)

[57] Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. Journal on Data Semantics 4, 29–57 (2015) (Cited on page 91)

[58] Di Federico, G., Burattin, A., Montali, M.: Human behavior as a process model: Which language to use? In: ITBPM@ BPM. pp. 18–25 (2021) (Cited on pages 13, 15, and 90)

[59] Dimaggio, M., Leotta, F., Mecella, M., Sora, D.: Process-based habit mining: Experiments and techniques. In: UIC 2016. pp. 145–152. IEEE (2016) (Cited on pages 15, 18, 19, and 48)

[60] Dogan, O.: Discovering customer paths from location data with process mining. EJEST 3(1), 139–145 (2020) (Cited on pages 18, 19, and 28)

[61] Dogan, O., Akkol, E., Olucoglu, M.: Understanding patient activity patterns in smart homes with process mining. In: Knowledge Graphs and Semantic Web: 4th Iberoamerican Conference and third Indo-American Conference, KGSWC 2022, Madrid, Spain, November 21–23, 2022, Proceedings. pp. 298–311. Springer (2022) (Cited on pages 18 and 19)

[62] Dogan, O., Bayo-Monton, J.L., Fernandez-Llatas, C., Oztaysi, B.: Analyzing of gender behaviors from paths using process mining: A shopping mall application. Sensors 19(3), 557 (2019) (Cited on pages 18 and 19)

[63] Dogan, O., Martinez-Millana, A., Rojas, E., Sepúlveda, M., Munoz-Gama, J., Traver, V., Fernandez-Llatas, C.: Individual behavior modeling with sensors using process mining. Electronics 8(7), 766 (2019) (Cited on pages 18 and 19)

[64] Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., Schreier, G.: The internet of things for ambient assisted living. In: ITNG'10. pp. 804–809 (2010) (Cited on page 2)

[65] Douskos, C.: Habit and intention. Philosophia 45(3), 1129–1148 (2017) (Cited on page 13)

[66] Duchateau, F., Bellahsene, Z.: Designing a benchmark for the assessment of schema matching tools. Open Journal of Databases 1(1), 3–25 (2014) (Cited on page 90)

[67] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013) (Cited on pages vi and 8)

[68] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013) (Cited on page 13)

[69] Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: a state-of-the-art literature review. In: Proceedings of the 11th international conference on subject-oriented business process management. pp. 1–10 (2019) (Cited on page 92)

[70] Ekanayake, C.C., Dumas, M., García-Bañuelos, L., La Rosa, M.: Slice, mine and dice: Complexity-aware automated discovery of business process models. In: Business Process Management: 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings. pp. 49–64. Springer (2013) (Cited on page 92)

[71] Esposito, L., Leotta, F., Mecella, M., Veneruso, S.: Unsupervised segmentation of smart home logs for human habit discovery. In: IE'22. pp. 1–8. IEEE (2022) (Cited on pages x, xi, 5, 18, 48, 58, and 73)

[72] Esposito, L., Leotta, F., Mecella, M., Veneruso, S.: Unsupervised segmentation of smart home logs for human habit discovery. In: 2022 18th International Conference on Intelligent Environments (IE). pp. 1–8 (2022) (Cited on page 44)

[73] Esposito, L., Veneruso, S.V., Leotta, F., Monti, F., Mathew, J.G., Mecella, M.: Unsupervised segmentation of human habits in smart home logs through process discovery. In: ITBPM@ BPM. pp. 56–61 (2021) (Cited on pages x and xi)

[74] Fanelli, D.: How many scientists fabricate and falsify research? a systematic review and meta-analysis of survey data. PLOS ONE 4(5) (2009) (Cited on pages vi and 30)

[75] Farahsari, P.S., Farahzadi, A., Rezazadeh, J., Bagheri, A.: A survey on indoor positioning systems for iot-based applications. IEEE Internet of Things Journal 9(10), 7680–7699 (2022) (Cited on page 91)

[76] Fernández-Llatas, C., Benedi, J.M., García-Gómez, J.M., Traver, V.: Process mining for individualized behavior modeling using wireless tracking in nursing homes. Sensors 13(11), 15434–15451 (2013) (Cited on pages 18, 19, and 29)

[77] Fernández-Llatas, C., Meneu, T., Benedi, J.M., Traver, V.: Activity-based process mining for clinical pathways computer aided design. In: 2010 annual international conference of the IEEE engineering in medicine and biology. pp. 6178–6181. IEEE (2010) (Cited on page 25)

[78] Fernandez-Llatas, C., Pileggi, S.F., Traver, V., Benedi, J.M.: Timed parallel automaton: A mathematical tool for defining highly expressive formal workflows. In: Fifth Asia Modell. Symposium. pp. 56–61 (2011) (Cited on page 21)

[79] Ferro, L.S., Marrella, A., Veneruso, S.V., Mecella, M., Catarci, T., et al.: An interactive learning experience for cybersecurity related issues. In: Proceedings of the International Workshop on Human-Centered Cybersecurity (In conjunction with CHITALY 2019) (2019) (Cited on page xiii)

[80] Fleury, A., Vacher, M., Noury, N.: Svm-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results. IEEE transactions on information technology in biomedicine 14(2), 274–283 (2009) (Cited on page 73)

[81] Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: A Standard for Object-Centric Event Logs, CCIS, vol. 1450, p. 169–175 (2021) (Cited on pages vii, 48, 56, and 57)

[82] Goossens, A., De Smedt, J., Vanthienen, J., van der Aalst, W.M.: Enhancing data-awareness of object-centric event logs. In: ICPM 2022. pp. 18–30. Springer (2022) (Cited on pages 56 and 57)

[83] Gudur, G.K., Sundaramoorthy, P., Umaashankar, V.: Activeharnet: Towards on-device deep bayesian active learning for human activity recognition. In: The 3rd international workshop on deep learning for mobile systems and applications. pp. 7–12 (2019) (Cited on page 93)

[84] Günther, C.W.: Process mining in flexible environments (2009) (Cited on pages 79 and 83)

[85] Günther, C.W., Van Der Aalst, W.M.: Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In: BPM. pp. 328–343 (2007) (Cited on pages 21 and 27)

[86] Günther, C.W., Verbeek, E.: Xes standard definition (Mar 2014) (Cited on pages vii, 47, and 57)

[87] Ha, S., Yun, J.M., Choi, S.: Multi-modal convolutional neural networks for activity recognition. In: 2015 IEEE International conference on systems, man, and cybernetics. pp. 3017–3022. IEEE (2015) (Cited on page 92)

[88] Hagras, H.: Toward human-understandable, explainable ai. Computer 51(9), 28–36 (2018) (Cited on page 2)

[89] Hammerla, N.Y., Halloran, S., Plötz, T.: Deep, convolutional, and recurrent models for human activity recognition using wearables. arXiv preprint arXiv:1604.08880 (2016) (Cited on page 92)

[90] Haresamudram, H., Essa, I., Plötz, T.: Assessing the state of self-supervised human activity recognition using wearables. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 6(3), 1–47 (2022) (Cited on pages ix, 3, and 29)

[91] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly pp. 75–105 (2004) (Cited on page 49)

[92] Hiremath, S.K., Nishimura, Y., Chernova, S., Plötz, T.: Bootstrapping human activity recognition systems for smart homes from scratch. Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 6(3), 1–27 (2022) (Cited on page 4)

[93] Huynh, T., Schiele, B.: Analyzing features for activity recognition. In: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies. pp. 159–163 (2005) (Cited on page 92)

[94] Ito, C., Cao, X., Shuzo, M., Maeda, E.: Application of cnn for human activity recognition with fft spectrogram of acceleration and gyro sensors. In: Proceedings of the 2018 ACM international joint conference and 2018 international symposium on pervasive and ubiquitous computing and wearable computers. pp. 1503–1510 (2018) (Cited on page 93)

[95] Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Fortino, G., Gal, A., Kannengiesser, U., Leotta, F., et al.: The internet of things meets business process management: a manifesto. IEEE Systems, Man, and Cybernetics Magazine 6(4), 34–44 (2020) (Cited on pages vii and 30)

[96] Janowicz, K., Haller, A., Cox, S.J.D., Le Phuoc, D., Lefrançois, M.: Sosa: A lightweight ontology for sensors, observations, samples, and actuators. Journal of Web Semantics 56, 1–10 (May 2019) (Cited on page 49)

[97] Janssen, D., Mannhardt, F., Koschmider, A., van Zelst, S.J.: Process model discovery from sensor event data. In: ICPM 2020. pp. 69–81 (2020) (Cited on pages vii, 47, and 48)

[98] Janssen, D., Mannhardt, F., Koschmider, A., Zelst, S.J.v.: Process model discovery from sensor event data. In: International Conference on Process Mining. pp. 69–81. Springer (2020) (Cited on pages 15, 18, 44, 64, and 73)

[99] Jobanputra, C., Bavishi, J., Doshi, N.: Human activity recognition: A survey. Procedia Computer Science 155, 698–703 (2019) (Cited on page 1)

[100] Jones, S., Shao, L.: Unsupervised spectral dual assignment clustering of human actions in context. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. pp. 604–611 (2014) (Cited on page 77)

[101] Ketchen, D.J., Shook, C.L.: The application of cluster analysis in strategic management research: an analysis and critique. Strategic management journal 17(6), 441–458 (1996) (Cited on page 74)

[102] Khattak, A.M., Truc, P.T.H., Hung, L.X., Vinh, L.T., Dang, V.H., Guan, D., Pervez, Z., Han, M., Lee, S., Lee, Y.K.: Towards smart homes using low level sensory data. Sensors 11(12), 11581–11604 (2011) (Cited on page 15)

[103] Kim, E., Helal, S., Lee, J., Hossain, S.: The making of a dataset for smart spaces. In: Intl. Conf. on ubiquitous intelligence and computing. pp. 110–124. Springer (2010) (Cited on page 42)

[104] Kim, Y., An, J., Lee, M., Lee, Y.: An activity-embedding approach for next-activity prediction in a multi-user smart space. In: 2017 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 1–6. IEEE (2017) (Cited on page 93)

[105] Kitchenham, B.: Procedures for performing systematic reviews. Keele, UK, Keele University 33(2004), 1–26 (2004) (Cited on page 14)

[106] Klinkmüller, C., Ponomarev, A., Tran, A.B., Weber, I., Aalst, W.v.d.: Mining blockchain processes: extracting process mining data from blockchain applications. In: International Conference on Business Process Management. pp. 71–86. Springer (2019) (Cited on page 9)

[107] Kormányos, B., Pataki, B.: Multilevel simulation of daily activities: Why and how? In: 2013 IEEE Intl. Conf. on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA). pp. 1–6. IEEE (2013) (Cited on pages 34, 37, and 40)

[108] Koschmider, A., Janssen, D., Mannhardt, F.: Framework for process discovery from sensor data. In: EMISA. p. 8 (2020) (Cited on page 48)

[109] Koschmider, A., Janssen, D., Mannhardt, F.: Framework for process discovery from sensor data. In: 10th Intl. Workshop on Enterprise Modeling and Information Systems Architectures (EMISA). vol. 2627, pp. 32–38 (2020) (Cited on page 91)

[110] Koschmider, A., Mannhardt, F., Heuser, T.: On the Contextualization of Event-Activity Mappings, LNBIP, vol. 342, p. 445–457. Springer (2019) (Cited on page 48)

[111] Krishnan, N., Cook, D.: Activity recognition on streaming sensor data. Pervasive and mobile computing 10, 138–154 (02 2014) (Cited on page 45)

[112] Krishnan, N.C., Cook, D.J.: Activity recognition on streaming sensor data (2014) (Cited on pages ix, 3, 4, 6, and 67)

[113] Lane, N.D., Georgiev, P.: Can deep learning revolutionize mobile sensing? In: Proceedings of the 16th international workshop on mobile computing systems and applications. pp. 117–122 (2015) (Cited on page 92)

[114] Lassen, K.B., van der Aalst, W.M.: Complexity metrics for workflow nets (2009) (Cited on pages 58 and 59)

[115] Lee, E.A.: Cyber physical systems: Design challenges. In: 2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC). pp. 363–369. IEEE (2008) (Cited on pages viii, 11, and 30)

[116] Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: Proceedings of SIGMOD'07. pp. 593–604 (2007) (Cited on pages 22 and 60)

[117] Lee, J.W., Cho, S., Liu, S., Cho, K., Helal, S.: Persim 3d: Context-driven simulation and modeling of human activities in smart spaces. IEEE Transactions on Automation Science and Eng. 12(4), 1243–1256 (2015) (Cited on page 42)

[118] Leemans, S.J., Fahland, D., Van Der Aalst, W.M.: Discovering block-structured process models from event logs containing infrequent behaviour. In: BPM'13. pp. 66–78. Springer (2013) (Cited on page 21)

[119] 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. pp. 36–44 (2020) (Cited on page 5)

[120] de Leoni, M., Pellattiero, L.: The benefits of sensor-measurement aggregation in discovering iot process models: A smart-house case study. In: BPM'21. pp. 403–415 (2021) (Cited on pages 18 and 91)

[121] de Leoni, M., Pellattiero, L.: The benefits of sensor-measurement aggregation in discovering iot process models: a smart-house case study. In: BPM 2021. pp. 403–415 (2021) (Cited on page 48)

[122] de Leoni, M., Pellattiero, L.: The benefits of sensor-measurement aggregation in discovering iot process models: A smart-house case study. In: Marrella, A., Weber, B. (eds.) Business Process Management Workshops. pp. 403–415. Springer International Publishing, Cham (2022) (Cited on page 5)

[123] Leotta, F., Mecella, M.: Plathea: a marker-less people localization and tracking system for home automation. Software: Practice and Experience 45(6), 801–835 (2015) (Cited on page 13)

[124] Leotta, F., Mecella, M., Mendling, J.: Applying process mining to smart spaces: Perspectives and research challenges. In: CAiSE 2015 Workshops. pp. 298–304. Springer (2015) (Cited on pages vi, viii, 8, 11, 13, 14, 15, 27, 28, 43, 44, 48, 58, 64, and 87)

[125] Leotta, F., Mecella, M., Sora, D.: Visual process maps: A visualization tool for discovering habits in smart homes. JAIHC 11(5), 1997–2025 (2020) (Cited on pages 5, 18, 48, 60, 69, 73, and 89)

[126] Leotta, F., Mecella, M., Sora, D., Catarci, T.: Surveying human habit modeling and mining techniques in smart spaces. Future Internet 11(1), 23 (2019) (Cited on pages viii, 1, and 15)

[127] Leotta, F., Mecella, M., Veneruso, S.: Unsupervised segmentation of smart home position logs for human activity analysis. In: 2023 19th International Conference on Intelligent Environments (IE). pp. 1–4 (2023) (Cited on pages x, xi, 5, and 64)

[128] Leotta, F., Mecella, M., Veneruso, S.: Discovering Human Habits through Process Mining: State of the Art and Research Challenges. Springer Cham (2024) (Cited on pages x and xii)

[129] Leotta, F., Veneruso, S.: Vpm: Analyzing human daily habits through process discovery. In: BPM (PhD/Demos). pp. 156–160 (2021) (Cited on pages viii, xi, 60, and 73)

[130] Li, C.Y., van Zelst, S.J., van der Aalst, W.M.: An activity instance based hierarchical framework for event abstraction. In: 2021 3rd International Conference on Process Mining (ICPM). pp. 160–167 (2021) (Cited on page 5)

[131] Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. pp. 2–11 (2003) (Cited on page 92)

[132] Liu, H., Hussain, F., Tan, C.L., Dash, M.: Discretization: An enabling technique (2002) (Cited on page 61)

[133] Lull, J.J., Bayo, J.L., Shirali, M., Ghassemian, M., Fernandez-Llatas, C.: Interactive process mining in iot and human behaviour modelling. In: Interactive Process Mining in Healthcare, pp. 217–231 (2021) (Cited on pages 18 and 19)

[134] Ma, H., Zhang, Z., Li, W., Lu, S.: Unsupervised human activity representation learning with multi-task deep clustering. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 5(1), 1–25 (2021) (Cited on page 4)

[135] Ma'arif, M.R.: Revealing daily human activity pattern using process mining approach. In: EECSI 2017. pp. 1–5 (2017) (Cited on page 18)

[136] Mangler, J., Grüger, J., Malburg, L., Ehrendorfer, M., Bertrand, Y., Benzin, J.V., Rinderle-Ma, S., Serral Asensio, E., Bergmann, R.: Datastream xes extension: embedding iot sensor data into extensible event stream logs. Future Internet 15(3), 109 (2023) (Cited on pages 56 and 57)

[137] Mannhardt, F., Bovo, R., Oliveira, M.F., Julier, S.: A taxonomy for combining activity recognition and process discovery in industrial environments. In: IDEAL 2018. pp. 84–93 (2018) (Cited on page 1)

[138] Mannhardt, F., De Leoni, M., Reijers, H.A., Van Der Aalst, W.M.: Decision mining revisited-discovering overlapping rules. In: Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings 28. pp. 377–392. Springer (2016) (Cited on page 91)

[139] Mathur, A., Zhang, T., Bhattacharya, S., Velickovic, P., Joffe, L., Lane, N.D., Kawsar, F., Lió, P.: Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices. In: 2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). pp. 200–211. IEEE (2018) (Cited on page 93)

[140] McCURDY, T., Glen, G., Smith, L., Lakkadi, Y.: The national exposure research laboratory's consolidated human activity database. Journal of Exposure Science & Environmental Epidemiology 10(6), 566–578 (2000) (Cited on page 18)

[141] Mohamed, A.r., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. IEEE transactions on audio, speech, and language processing 20(1), 14–22 (2011) (Cited on page 6)

[142] Munoz-Gama, J., Martin, N., Fernandez-Llatas, C., Johnson, O.A., Sepúlveda, M., Helm, E., Galvez-Yanjari, V., Rojas, E., Martinez-Millana, A., Aloini, D., et al.: Process mining for healthcare: Characteristics and challenges. Journal of Biomedical Informatics 127, 103994 (2022) (Cited on page 9)

[143] Murahari, V.S., Plötz, T.: On attention models for human activity recognition. In: Proceedings of the 2018 ACM international symposium on wearable computers. pp. 100–103 (2018) (Cited on page 93)

[144] Narayana, S., Prasad, R.V., Rao, V.S., Prabhakar, T.V., Kowshik, S.S., Iyer, M.S.: Pir sensors: Characterization and novel localization technique. In: ISPN 2015. pp. 142–153 (2015) (Cited on pages 26 and 59)

[145] Neal, D.T., Wood, W.: Automaticity in situ and in the lab: The nature of habit in daily life. Oxford handbook of human action pp. 442–457 (2009) (Cited on page 13)

[146] Niebles, J.C., Wang, H., Fei-Fei, L.: Unsupervised learning of human action categories using spatial-temporal words. International journal of computer vision 79(3), 299–318 (2008) (Cited on page 77)

[147] Oosterlinck, D., Benoit, D.F., Baecke, P., Van de Weghe, N.: Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits. Applied geography 78, 55–65 (2017) (Cited on page 28)

[148] Ordóñez, F., De Toledo, P., Sanchis, A., et al.: Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. Sensors 13(5), 5460–5477 (2013) (Cited on pages 18, 19, 22, and 73)

[149] Peeva, V., Mannel, L., Aalst, W.: From place nets to local process models (04 2022) (Cited on page 5)

[150] Plötz, T., Hammerla, N.Y., Olivier, P.L.: Feature learning for activity recognition in ubiquitous computing. In: Twenty-second international joint conference on artificial intelligence (2011) (Cited on page 6)

[151] Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. arXiv:1303.2554 [cs] (Mar 2013) (Cited on page 48)

[152] Pourmirza, S., Dijkman, R., Grefen, P.: Correlation miner: mining business process models and event correlations without case identifiers. International Journal of Cooperative Information Systems 26(02), 1742002 (2017) (Cited on page 92)

[153] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.L., Chen, S.C., Iyengar, S.S.: A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys (CSUR) 51(5), 1–36 (2018) (Cited on page 92)

[154] Prathama, F., Yahya, B.N., Lee, S.L.: A multi-case perspective analytical framework for discovering human daily behavior from sensors using process mining. In: COMPSAC 2021. pp. 638–644 (2021) (Cited on page 18)

[155] Quinde, M., Giménez-Manuel, J., Oguego, C.L., Augusto, J.C.: Achieving multi-user capabilities through an indoor positioning system based on ble beacons. In: 2020 16th International Conference on Intelligent Environments (IE). pp. 13–20. IEEE (2020) (Cited on page 13)

[156] Ramasamy Ramamurthy, S., Roy, N.: Recent trends in machine learning for human activity recognition—a survey. WIREs Data Mining and Knowledge Discovery 8(4), e1254 (2018) (Cited on page 7)

[157] Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems: challenges, methods, technologies, vol. 54. Springer (2012) (Cited on page 91)

[158] Reinkemeyer, L.: Process mining in a nutshell. In: Process Mining in Action, pp. 3–10 (2020) (Cited on pages viii, 12, and 23)

[159] Riboni, D., Pareschi, L., Radaelli, L., Bettini, C.: Is ontology-based activity recognition really effective? In: 2011 IEEE international conference on pervasive computing and communications workshops (PERCOM workshops). pp. 427–431. IEEE (2011) (Cited on page 93)

[160] Rodriguez-Fernandez, V., Trzcionkowska, A., Gonzalez-Pardo, A., Brzychczy, E., Nalepa, G.J., Camacho, D.: Conformance checking for time-series-aware processes. IEEE TII 17(2), 871–881 (2021) (Cited on pages 48, 56, and 57)

[161] Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al.: Collecting complex activity datasets in highly rich networked sensor environments. In: 2010 Seventh Intl. Conf. on networked sensing systems (INSS). pp. 233–240. IEEE (2010) (Cited on page 73)

[162] Sato, D.M.V., De Freitas, S.C., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM CSUR 54(9), 1–38 (2021) (Cited on page 92)

[163] Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal communications 8(4), 10–17 (2001) (Cited on page 28)

[164] Seiger, R., Zerbato, F., Burattin, A., Garcia-Banuelos, L., Weber, B.: Towards iot-driven process event log generation for conformance checking in smart factories. In: EDOCW. p. 20–26 (Oct 2020) (Cited on pages vii, 47, and 48)

[165] Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A.: The road from sensor data to process instances via interaction mining. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) Advanced Information Systems Engineering. pp. 257–273. Springer International Publishing, Cham (2016) (Cited on page 5)

[166] Serral, E., De Smedt, J., Vanthienen, J.: Making business environments smarter: a context-adaptive petri net approach. In: UIC 2014. pp. 343–348 (2014) (Cited on pages vii, 47, and 48)

[167] Serral, E., Schuster, D., Bertrand, Y.: Supporting users in the continuous evolution of automated routines in their smart spaces. In: BPM 2021. pp. 391–402 (2021) (Cited on page 18)

[168] Serral, E., Valderas, P., Pelechano, V.: Context-adaptive coordination of pervasive services by interpreting models during runtime. The Computer Journal 56(1), 87–114 (2013) (Cited on pages 15, 31, 32, 37, and 38)

[169] Serral, E., Valderas, P., Pelechano, V.: Addressing the evolution of automated user behaviour patterns by runtime model interpretation. Software & Systems Modeling 14(4), 1387–1420 (2015) (Cited on pages 31, 33, and 37)

[170] Singla, G., Cook, D.J., Schmitter Edgecombe, M.: Recognizing independent and joint activities among multiple residents in smart environments. Journal of Ambient Intelligence and Humanized Computing 1(1), 57–63 (2010) (Cited on page 73)

[171] Soffer, P., et al.: From event streams to process models and back: Challenges and opportunities. Information Systems 81, 181–200 (Mar 2019) (Cited on pages vii and 47)

[172] Sora, D., Leotta, F., Mecella, M.: An habit is a process: a bpm-based approach for smart spaces. In: International Conference on Business Process Management. pp. 298–309. Springer (2017) (Cited on pages 9 and 13)

[173] Sora, D., Leotta, F., Mecella, M.: Addressing multi-users open challenge in habit mining for a process mining-based approach. In: Integrating Research Agendas and Devising Joint Challenges. pp. 266–273 (2018) (Cited on page 18)

[174] Sztyler, T., Carmona, J.J.: Activities of daily living of several individuals (11 2015) (Cited on pages 18, 19, and 22)

[175] Sztyler, T., Carmona, J., Völker, J., Stuckenschmidt, H.: Self-tracking reloaded: applying process mining to personalized health care from labeled sensor data. Transactions on Petri nets and other models of concurrency XI pp. 160–180 (2016) (Cited on pages 18, 19, and 90)

[176] Tapia, E.M., Intille, S.S., Larson, K.: Activity recognition in the home using simple and ubiquitous sensors. In: Intl. Conf. on pervasive computing. pp. 158–175. Springer (2004) (Cited on pages 18, 19, 22, and 73)

[177] Tax, N.: Human activity prediction in smart home environments with lstm neural networks. In: 2018 14th IE. pp. 40–47 (2018) (Cited on page 15)

[178] Tax, N., Alasgarov, E., Sidorova, N., Haakma, R.: On generation of time-based label refinements. arXiv preprint arXiv:1609.03333 (2016) (Cited on page 15)

[179] Tax, N., Alasgarov, E., Sidorova, N., Haakma, R., van der Aalst, W.M.: Generating time-based label refinements to discover more precise process models. JAISE 11(2), 165–182 (2019) (Cited on page 18)

[180] Tax, N., Dalmas, B., Sidorova, N., van der Aalst, W.M., Norre, S.: Interest-driven discovery of local process models. Information Systems 77, 105–117 (2018), `https://www.sciencedirect.com/science/article/pii/S0306437917304477` (Cited on page 5)

[181] Tax, N., Sidorova, N., van der Aalst, W.M., Haakma, R.: Heuristic approaches for generating local process models through log projections. In: 2016 IEEE SSCI. pp. 1–8 (2016) (Cited on page 18)

[182] Tax, N., Sidorova, N., Haakma, R., Aalst, W.: Mining process model descriptions of daily life through event abstraction. In: IntelliSys 2016. pp. 83–104 (2016) (Cited on page 15)

[183] Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.: Event abstraction for process mining using supervised learning techniques. In: Proceedings of SAI ISC. pp. 251–269 (2016) (Cited on page 18)

[184] Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.: Mining local process models. Journal of Innovation in Digital Ecosystems 3(2), 183–196 (2016), https://www.sciencedirect.com/science/article/pii/S2352664516300232 (Cited on page 5)

[185] Tazari, M.R., Furfari, F., Fides Valero, Á., Hanke, S., Höftberger, O., Kehagias, D., Mosmondor, M., Wichert, R., Wolf, P.: The universaal reference model for aal. IOS press (2012) (Cited on page v)

[186] Theodoropoulou, G., Bousdekis, A., Miaoulis, G., Voulodimos, A.: Process mining for activities of daily living in smart homecare. In: PCI 2020. pp. 197–201 (2020) (Cited on page 18)

[187] Torres, V., Serral, E., Valderas, P., Pelechano, V., Grefen, P.: Modeling of iot devices in business processes: A systematic mapping study. In: 2020 IEEE 22nd Conference on Business Informatics (CBI). vol. 1, pp. 221–230. IEEE (2020) (Cited on pages 15 and 90)

[188] Trzcionkowska, A., Brzychczy, E.: Practical aspects of event logs creation for industrial process modelling. MAPE 1(1), 77–83 (Sep 2018) (Cited on page 48)

[189] Valencia-Parra, A., Ramos-Gutierrez, B., Varela-Vaca, A.J., Gomez-Lopez, M.T., Bernal, A.G.: Enabling process mining in aircraft manufactures: Extracting event logs and discovering processes from complex data pp. 166–177 (2019) (Cited on page 48)

[190] Van Der Aalst, W.: Process mining: discovery, conformance and enhancement of business processes, vol. 2. Springer (2011) (Cited on pages 10 and 28)

[191] Van Der Aalst, W.: Process mining: Overview and opportunities. ACM Transactions on Management Information Systems (TMIS) 3(2), 1–17 (2012) (Cited on page 80)

[192] Van Houdt, G., Depaire, B., Martin, N.: Unsupervised event abstraction in a process mining context: A benchmark study. In: Process Mining Workshops: ICPM 2020 International Workshops, Padua, Italy, October 5–8, 2020, Revised Selected Papers 2. pp. 82–93. Springer (2021) (Cited on pages viii, 12, 27, and 91)

[193] Van Houdt, G., de Leoni, M., Martin, N., Depaire, B.: An empirical evaluation of unsupervised event log abstraction techniques in process mining. Information Systems 121, 102320 (2024) (Cited on pages 5 and 12)

[194] Van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: Proceedings of the 10th Intl. Conf. on Ubiquitous computing. pp. 1–9 (2008) (Cited on pages 18, 19, 22, and 73)

[195] Veneruso, S., Bertrand, Y., Leotta, F., Serral, E., Mecella, M.: A model-based simulator for smart homes: Enabling reproducibility and standardization. JAISE 15(2), 143–166 (2023) (Cited on pages vii, viii, xi, 17, 18, 19, 52, 56, 57, 64, and 73)

[196] Veneruso, S., Ferro, L.S., Marrella, A., Mecella, M., Catarci, T., et al.: A game-based learning experience for improving cybersecurity awareness. In: CEUR WORKSHOP PROCEEDINGS. vol. 2597, pp. 235–242. CEUR-WS (2020) (Cited on page xiii)

[197] Veneruso, S., Leotta, F., Mecella, M.: On the usefulness of human behaviour process models: a user study. In: 2024 20th International Conference on Intelligent Environments (IE) (2024) (Cited on pages xii and 79)

[198] Veneruso, S.V., Catarci, T., Ferro, L.S., Marrella, A., Mecella, M.: V-door: A real-time virtual dressing room application using oculus rift. In: Proceedings of the International Conference on Advanced Visual Interfaces. pp. 1–3 (2020) (Cited on pages xiii and xiv)

[199] Veneruso, S.V., Ferro, L.S., Marrella, A., Mecella, M., Catarci, T.: Cybervr: an interactive learning experience in virtual reality for cybersecurity related issues. In: Proceedings of the International Conference on Advanced Visual Interfaces. pp. 1–8 (2020) (Cited on pages x, xiii, and xiv)

[200] Verbeek, H., Buijs, J.C., Van Dongen, B.F., Van Der Aalst, W.M.: Xes, xesame, and prom 6. In: Intl. Conf. on Advanced Information Systems Eng. pp. 60–75. Springer (2010) (Cited on page 33)

[201] Weijters, A., Ribeiro, J.T.S.: Flexible heuristics miner (fhm). In: 2011 IEEE symposium on computational intelligence and data mining (CIDM). pp. 310–317. IEEE (2011) (Cited on page 27)

[202] Wiemuth, M., Junger, D., Leitritz, M., Neumann, J., Neumuth, T., Burgert, O.: Application fields for the new object management group (omg) standards case management model and notation (cmmn) and decision management notation (dmn) in the perioperative field. International journal of computer assisted radiology and surgery 12, 1439–1449 (2017) (Cited on page 90)

[203] Yadav, S.K., Tiwari, K., Pandey, H.M., Akbar, S.A.: A review of multimodal human activity recognition with special emphasis on classification, applications, challenges and future directions. Knowledge-Based Systems 223, 106970 (2021), https://www.sciencedirect.com/science/article/pii/S0950705121002331 (Cited on page 5)

[204] Yang, J., Nguyen, M.N., San, P.P., Li, X., Krishnaswamy, S.: Deep convolutional neural networks on multichannel time series for human activity recognition. In: Ijcai. vol. 15, pp. 3995–4001. Buenos Aires, Argentina (2015) (Cited on page 92)

[205] Yang, Y., Saleemi, I., Shah, M.: Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. IEEE transactions on pattern analysis and machine intelligence (2012) (Cited on page 77)

[206] Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive and mobile computing 8(1), 36–66 (2012) (Cited on page 1)

[207] van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A.: Event abstraction in process mining: literature review and taxonomy. Granular Computing 6, 719–736 (2021) (Cited on pages 27, 48, 90, and 91)

[208] Zerbato, F., Seiger, R., Di Federico, G., Burattin, A., Weber, B.: Granularity in process mining: Can we fix it? In: CEUR Workshop Proceedings. vol. 2938, pp. 40–44 (2021) (Cited on page 27)

[209] Zhang, D., Yao, L., Zhang, X., Wang, S., Chen, W., Boots, R., Benatallah, B.: Cascade and parallel convolutional recurrent neural networks on eeg-based intention recognition for brain computer interface. In: Proceedings of the aaai conference on artificial intelligence. vol. 32 (2018) (Cited on page 93)

Appendix

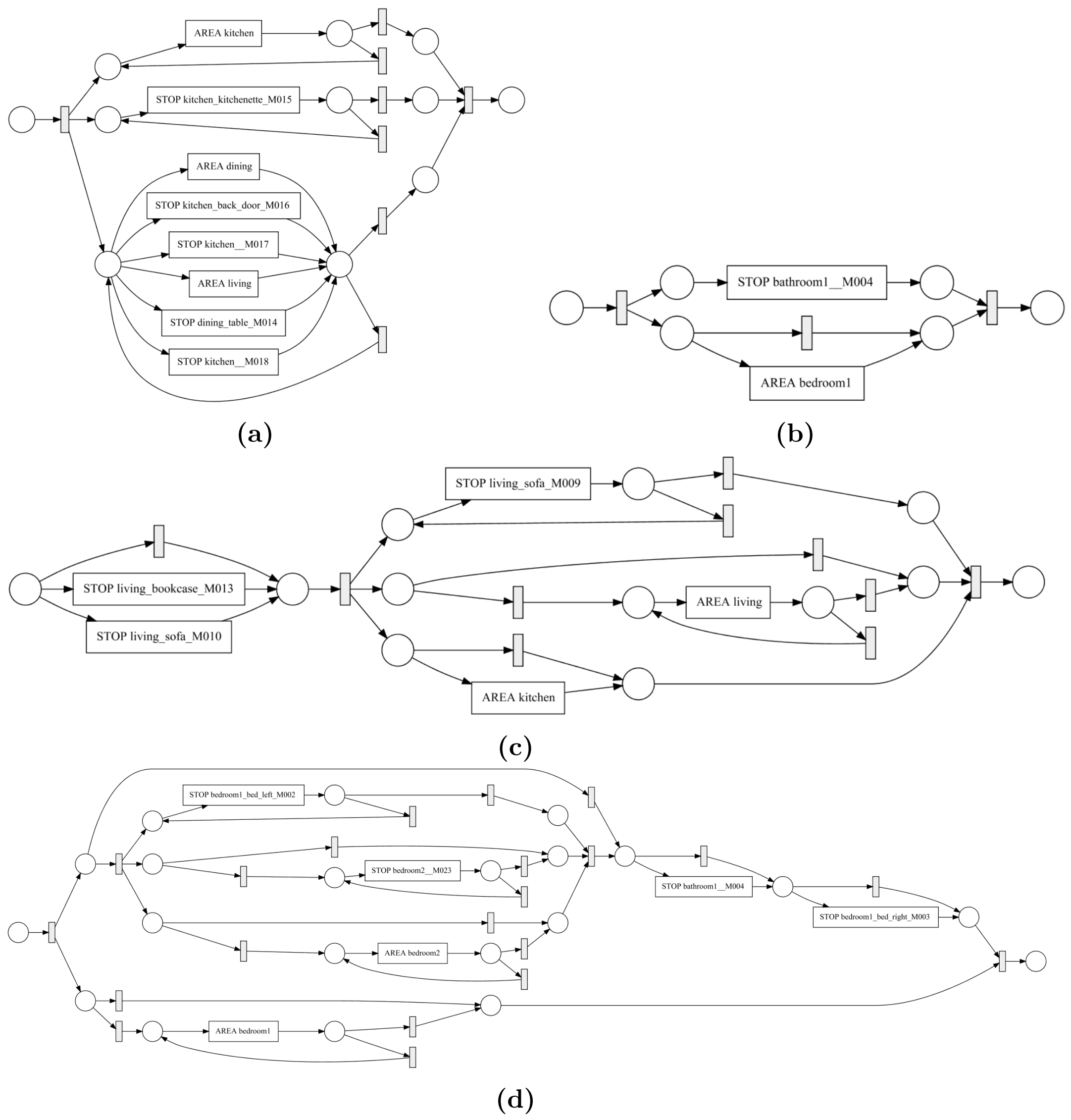


Figure A.1. Habit “05:15-07:00”: (a) Petri net of the habit filtered on *meal preparation*, (b) Petri net of the habit filtered on *bed to toilet*, (c) Petri net of the habit filtered on *relax*, (d) Petri net of the habit filtered on *sleeping*.

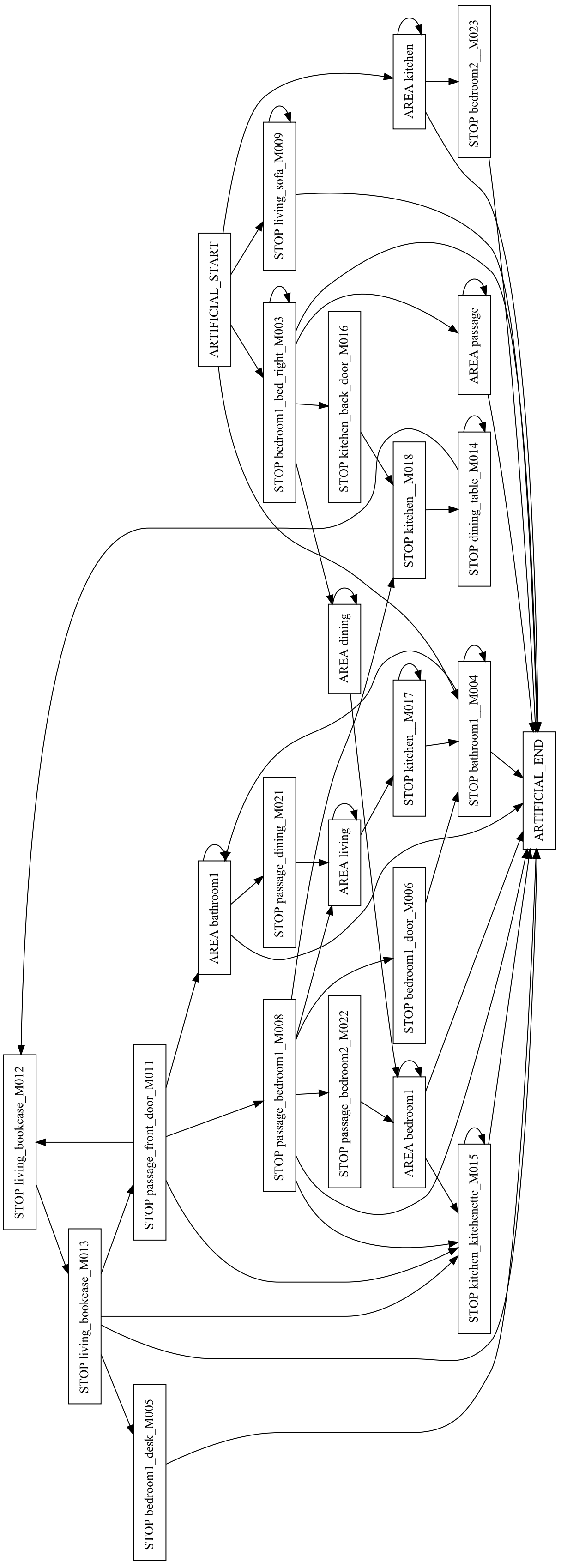


Figure A.2. Process model representing the “05:15-07:00” habit range extracted from the *heuristic miner*.

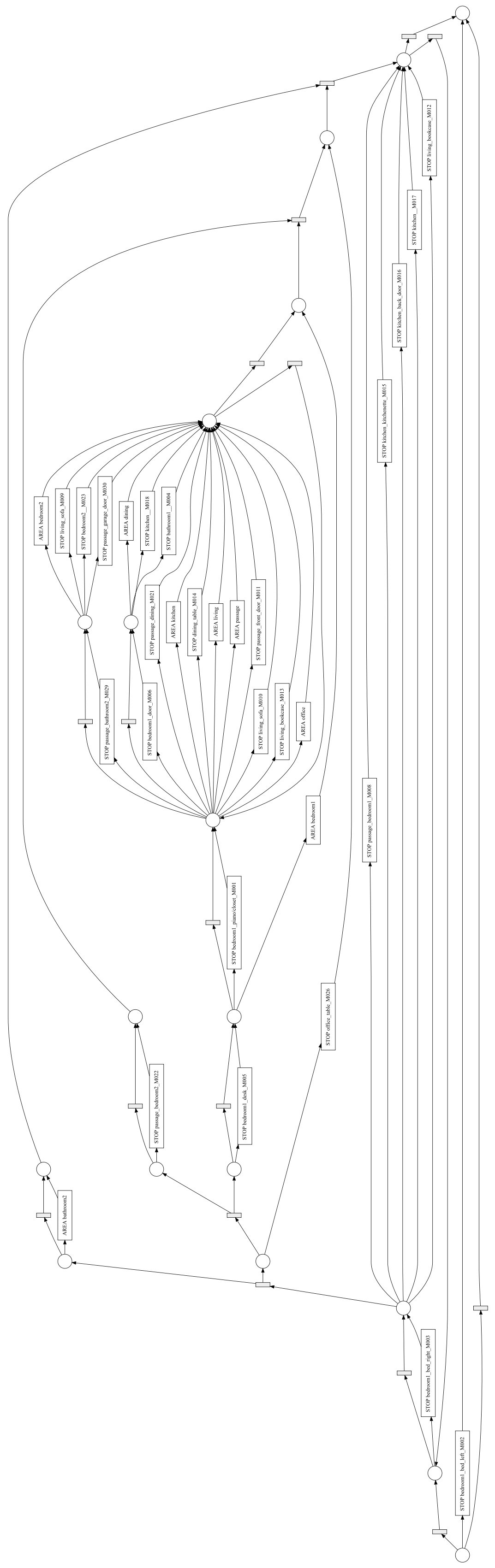


Figure A.3. Process model representing the "05:15-07:00" habit range extracted from the *inductive miner*.

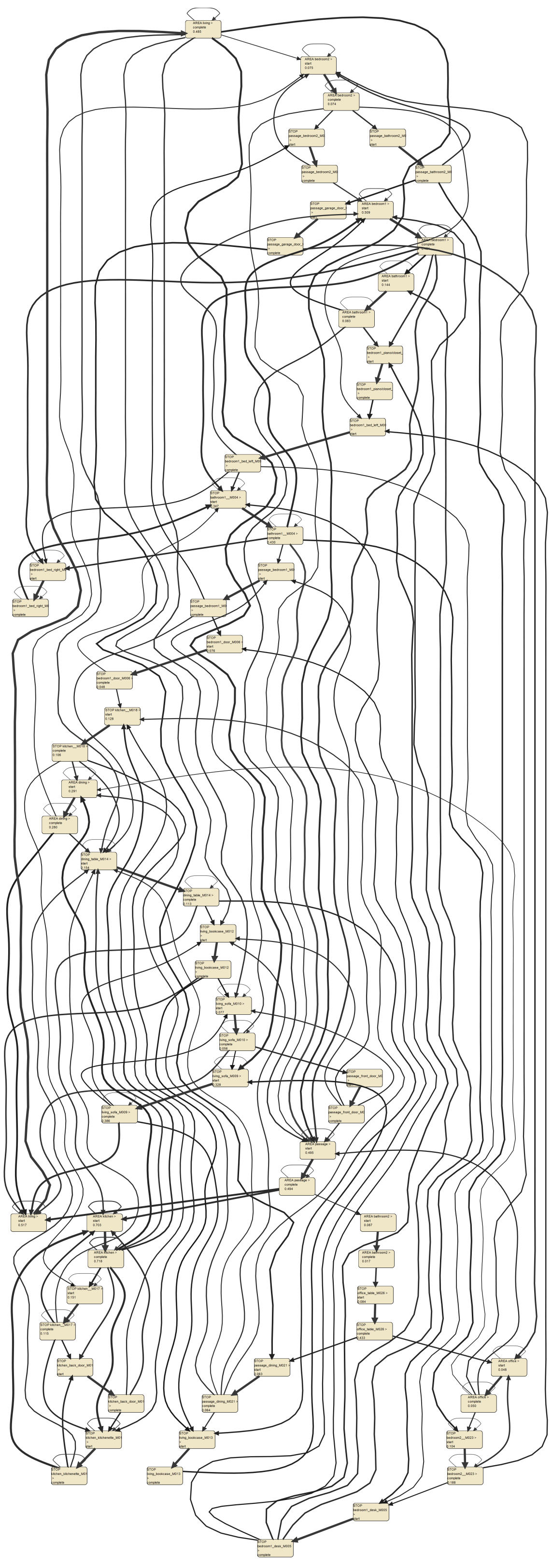
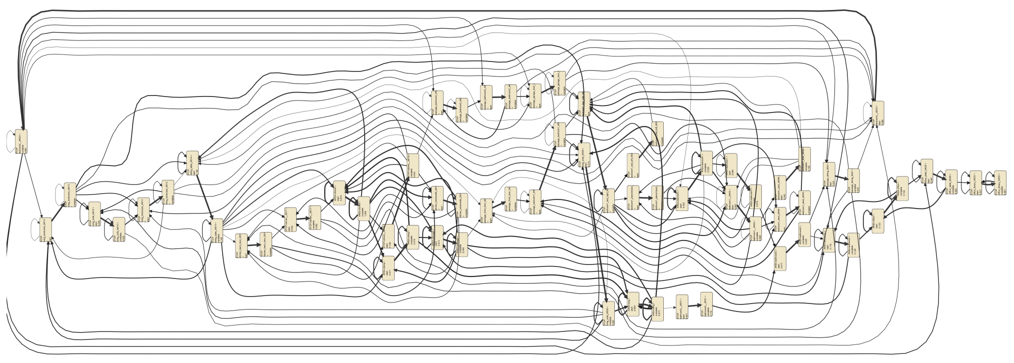
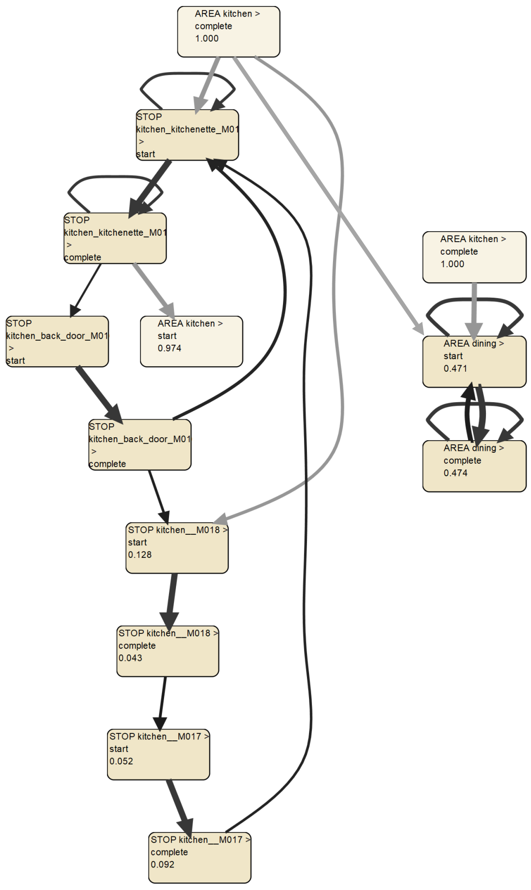


Figure A.4. Process model representing the “05:15-07:00” habit range extracted from the *fuzzy miner*.



(a)



(b)

Figure A.5. Activity "eating": (a) Process model extracted from the *fuzzy miner*, (b) a filtered component that emphasizes significant nodes in the model provided in (a).

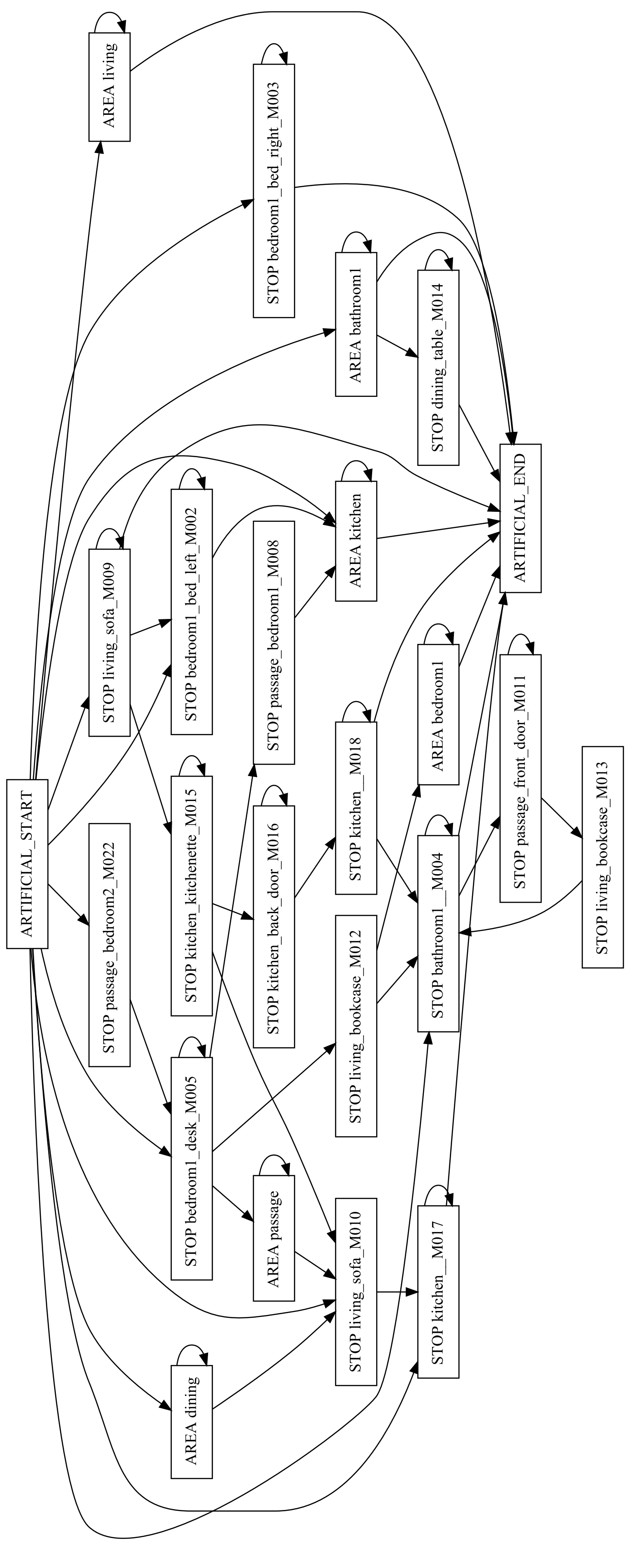


Figure A.6. Process model representing the "bed to toilet" activity extracted from the *heuristic miner*.

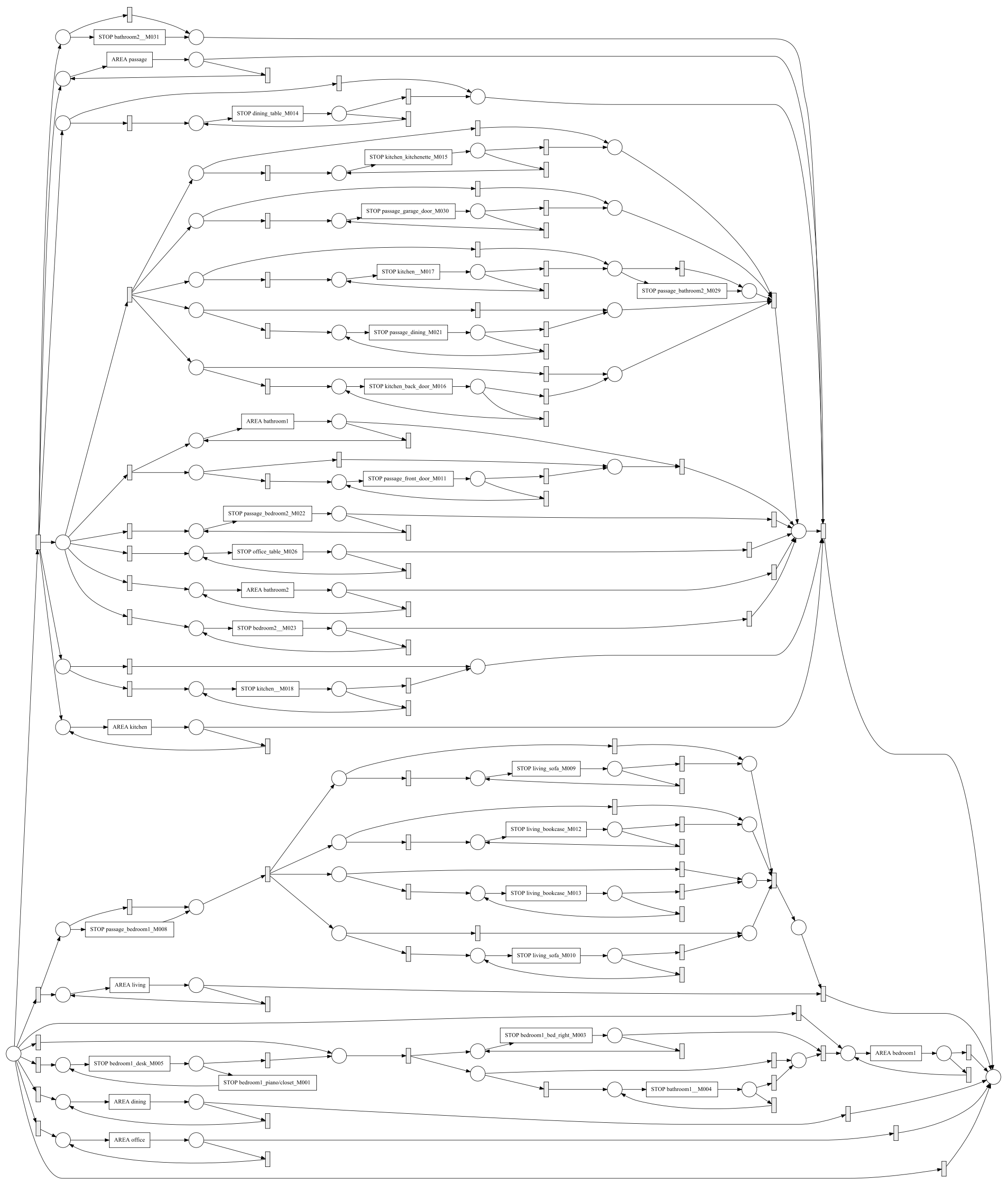


Figure A.7. Process model representing the "leave home" activity extracted from the *inductive miner.*