PROYECTO FIN DE CARRERA

Título:	Aplicación de Process Mining para modelos de comportamiento en entornos de Ambient Intelligence
Título (inglés):	Application of Process Mining to Behaviour Modelling in Ambient Intelligence environments
Autor:	Diego Arespacochaga Muñoz
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Mercedes Garijo Ayestarán
Vocal:	Carlos Ángel Iglesias Fernández
Secretario:	Álvaro Carrera Barroso
Suplente:	Juan Fernando Sánchez Rada

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FINAL DE GRADO

APPLICATION OF PROCESS MINING TO BEHAVIOUR MODELLING IN AMBIENT INTELLIGENCE ENVIRONMENTS

Diego Arespacochaga Muñoz

Febrero 2016

"Si quieres construir un barco, no empieces por buscar madera, cortar tablas o distribuir el trabajo. Evoca primero en los hombres y mujeres el anhelo del mar libre y ancho."
Antoine de Saint-Exupéry

Resumen

Esta memoria es el resultado de un proyecto cuyo objetivo ha sido realizar un análisis de la posible aplicación de técnicas relativas al Process Mining para entornos AmI (Ambient Intelligence).

Dicho análisis tiene la facultad de presentar de forma clara los resultados extraídos de los procesos relativos a un caso de uso planteado, así como de aplicar dichos resultados a aplicaciones relativas a entornos AmI, como automatización de tareas o simulación social basada en agentes.

Para que dicho análisis sea comprensible por el lector, se presentan detalladas explicaciones de los conceptos tratados y las técnicas empleadas. Además, se analizan exhaustivamente las dos herramientas software más utilizadas en cuanto a minería de procesos se refiere, ProM y Disco, presentando ventajas e inconvenientes de cada una, así como una comparación entre las dos.

Posteriormente se ha desarrollado una metodología para el análisis de procesos con la herramienta ProM, anteriormente mencionada, explicando cuidadosamente cada uno de los pasos así como los fundamentos de los algoritmos utilizados.

Por último, se han presentado las conclusiones extraídas del trabajo, así como las posibles líneas de continuación del proyecto.

Palabras clave: Process Mining, Ambient Intelligence, SmartHome, ProM, Disco, Heuristic Miner, Fuzzy Miner.

Agradecimientos

A mi familia, en especial a mis padres, por su cariño y su apoyo incondicional, por sus correcciones, por su respeto a mis tiempos y por su compañía, fundamental durante estos años.

A mis amigos, en especial a Jorge, compañeros inigualables en esta aventura, y a Lourdes, por ser bastón y guía, pero sobre todo amiga y compañera. A todos ellos por su paciencia, por su cariño, y por encima de todo por testimoniarme lo fundamental de la vida. Gracias.

Me gustaría dedicar unas líneas de este trabajo, que marca el final de la primera fase de una apasionante carrera, a agradecer el apoyo, la confianza y la presencia de aquellos que me han acompañado durante estos años.

A Irene, asesora, consejera y amiga, gracias por tu apoyo todos estos años.

A mis profesores y compañeros de la Escuela, con especial cariño a Juan Antonio. A todos aquellos que me he cruzado en el camino y me han permitido aprender de ellos. Mención de honor a Carlos, por su consejo, su entrega y su apoyo estos años, de forma evidente este tiempo de realización de este trabajo.

A Marcos Pou.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

Ambient intelligence (AmI) [6] deals with a new world of ubiquitous computing devices, where physical environments interact intelligently and unobtrusively with people. These environments should be aware of people's needs, customizing requirements and forecasting behaviours. AmI environments can be diverse, such as homes, offices, meeting rooms, schools, hospitals, control centers, vehicles, tourist attractions, stores, sports facilities, and music devices. Artificial intelligence research aims to include more intelligence in AmI environments, enabling better support for humans and access to the essential knowledge for making better decisions when interacting with these environments. For this purpose, process mining, applied not only to business processes but also to processes related with healthcare, security, smart cities...etc, may be the base for developing this intelligence needed by AmI environments.

Yet, there are researchers and tool vendors that assume logs to be complete and free of noise, and although there are some tools providing case-hardened process discovery techniques, many improvements are possible to construct more intuitive models that are able to explain the most likely and common behaviour.

1.2 Final project goals

The main purpose of this final project is to demonstrate the viability of using the results of process mining techniques for Ambient intelligence applications.

First we need to understand what process mining is, and which techniques does it provide. Then, we need to analyse the available process mining software to choose the most fit our needs. And we will also establish a general methodology for analysing processes.

Finally, we will mine a dataset extracted from a SmartHome in order to obtain helpful results for AmI applications such task automation or based-agents social simulation.

1.3 Structure of this final project

In this section we will provide a brief overview of all the chapters of this final project. It has been structured as follows:

Chapter 1 provides an introduction to the problem which will be approached in this

project. It provides an overview of the benefits of applying process mining potential for AmI environments. Furthermore, a deeper description of the project is also given.

Chapter 2 contains an introduction to what process mining is, an explanation about the main formats with which process mining tools work, and an overview of the existing technologies which the development of the project will rely on.

Chapter 3 describes a general methodology implemented to mine processes using ProM, and also exposes the fundamentals of the main algorithms ProM used to mine real logs.

Chapter 4 describes a selected use case, the architecture of the system, and the steps followed to exploit the dataset extracted from the SmartHome. It also briefly describes how can we apply the obtained results to AmI applications such as task automation and agent-based social simulation.

Chapter 6 sums up the findings and conclusions found throughout the document and gives a hint about future development to continue the work done for this final project.

CHAPTER 2

Analysis of Process Mining Tools

This chapter introduces what process mining is, providing a brief explanation about its main features. Then, it also describes the most common data formats used for the process mining techniques: MXML and XES, taking a picture of how are them structured. Finally, it is given a detailed analysis between the most common both technologies for Process Mining: ProM and Disco, explaining their main filters and plugins for cleaning and mining event logs, extracting conclusions and comparing the weaknesses and strengths of each tool.

2.1 Process Mining

Process mining, as it is said on [2], is a relatively young research discipline that sits between computational intelligence and data mining on the one hand, and process modelling and analysis on the other hand. The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's (information) systems.

The growth of a digital universe that is well-aligned with processes in organizations makes it possible to record and analyse events. Events may range from the withdrawal of cash from an ATM, a doctor adjusting an X-ray machine, a citizen applying for a driver licence, the submission of a tax declaration, and the receipt of an e-ticket number by a traveller. The challenge is to exploit event data meaningful, for example, to provide insights, identify bottlenecks, anticipate problems, record policy violations, recommend countermeasures, and streamline processes. Process mining aims to do exactly that.

Starting point for process mining is an event log. All process mining techniques assume that it is possible to sequentially record events such that each event refers to an activity and is related to a particular case. Event logs may store additional information about events. In fact, whenever possible, process mining techniques use extra information such as the resource executing or initiating the activity, the timestamps of the event, or data elements recorded with the event.

As shown in Figure 2.1, we can distinguish three types of process mining:

1. **Discovery**. This technique produces a model from an event log without using any a-priori information. Process discovery is the most prominent process mining technique.

2. **Conformance**. It compares existing process model with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa.

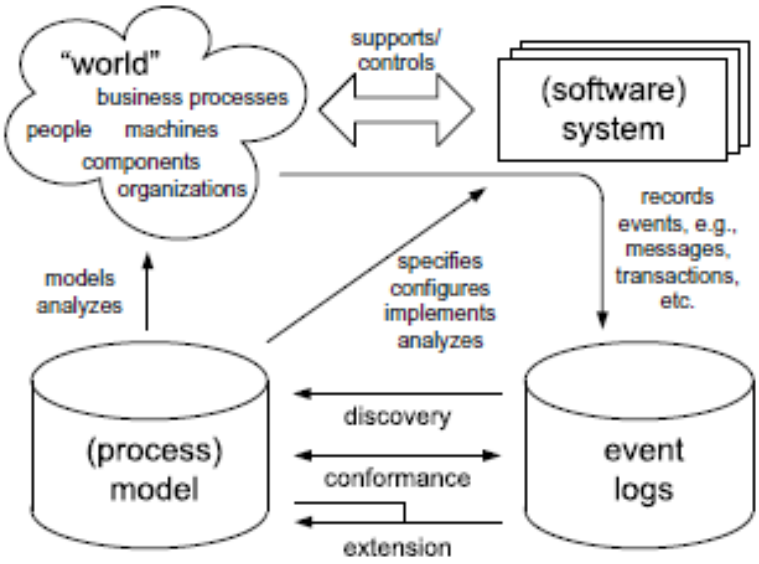


Figure 2.1: . Positioning of the three main types of process mining: (a) discovery, (b) conformance checking, and (c) enhancement [1]

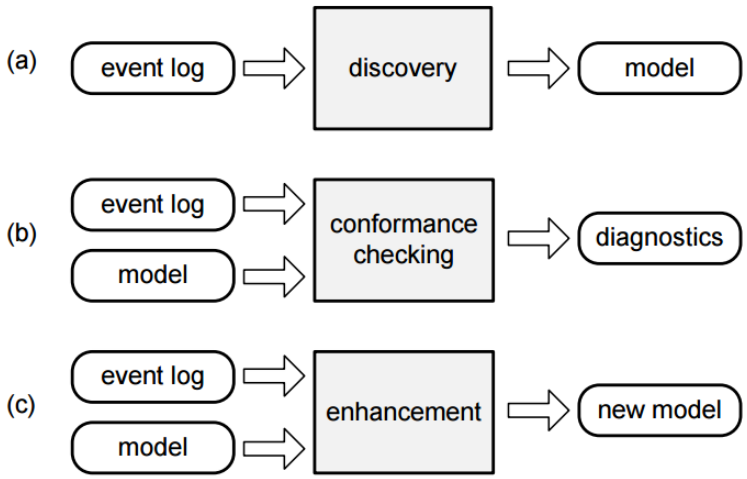


Figure 2.2: . The three basic types of process mining explained in terms of input and output: (a) discovery, (b) conformance checking, and (c) enhancement [2]

3. **Enhancement**. It extends an existing process model using information about the actual process recorded in some event log.

On Figure 2.2 we can see the input and output for each different technique. Techniques for discovery take an event log and produce a model. The discovered model is typically

a process model (e.g., a Petri net, BPMN, EPC, or UML activity diagram), however, the model may also describe other perspectives (e.g., a social network). Conformance checking techniques need an event log and a model as input. The output consists of diagnostic information showing differences and commonalities between model and log. Techniques for model enhancement (repair or extension) also need an event log and a model as input. The output is an improved or extended model.

2.2 Process Mining formats: XES and MXML

MXML emerged in 2003 and was later adopted by the process mining tool ProM. Using MXML, it is possible to store event logs using an XML-based syntax. MXML was the de facto standard format for storing event logs and was used as the native input format for ProM until XES (eXtensible Event Stream) [7] emerged as the new standard for storing event logs.

2.2.1 Mining XML Format

Mining XML format (MXML) [8] is a generic XML-based format suitable for representing and storing event log data. While focusing on the core information necessary to perform process mining, the format reserves generic fields for extra information that is potentially provided by a PAIS. There is a root node on each MXML document (*Work-flow Log*) which represents a log file. *Source* elements contained on a Workflow Log describe the system the log has been imported from. A Work-flow Log can contain an arbitrary number of *Processes* as child elements. Events occurred during the execution of a specific process are grouped in the process element. The single executions of that process definition are represented by child elements of type *ProcessInstance*. Thus, each process instance represents one specific case in the system. Finally, process instances each group an arbitrary number of *AuditTrailEntry* child nodes, each describing one specific event in the log. The *WorkflowModelElement* describes the process definition element to which the event refers, e.g. the name of the task that was executed. The second mandatory element is the *EventType*, describing the nature of the event, e.g. whether a task was scheduled, completed, etc. Two further child elements of an audit trail entry are optional, namely the *Timestamp* and the *Originator*. Timestamp holds the exact date and time of when the event has occurred, while the originator identifies the resource, e.g. person, which has triggered the event in the system.

2.2.2 XES Format

XES [7] is an XML-based standard for event logs. Its purpose is to provide a generally-acknowledged format for the interchange of event log data between tools and application domains. Its primary purpose is for process mining, i.e. the analysis of operational processes based on their event logs. However, XES has been designed to also be suitable for general data mining, text mining, and statistical analysis. The basic hierarchy of an XES document follows the universal structure of event log information. On the top level there is one log object, which contains all event information that is related to one specific process. A log contains an arbitrary number (it may be empty) of trace objects. Each trace describes the execution of one specific instance, or case, of the logged process. Every trace contains an arbitrary number (it may be empty) of event objects. Events represent atomic granules of activity that have been observed during the execution of a process. As such, an event has no duration. All information in an event log is stored in attributes. Attributes describe their parent element (log, trace, etc.). The XES standard requires for each attribute key it must contain no line feeds, no carriage returns and no tabs, as well as the key of each attribute must be unique within their enclosing container. Attributes can contain six types of elements: String, Date, Int, Float, Boolean, ID, List or Container.

2.3 Process Mining Tools: ProM

The first process mining tool we are going to talk about is ProM [9]. It is an open-source and extensible framework based on a plugin philosophy, to promote in this way the contribution of researchers and developers. This software is based on Java and it allows us to have a quick and intuitive view of the results of processing an event log file. In ProM we can find two main phases detailed next: pre-processing phase, where we will clean the event log by configuring the filters to obtain an event log only with the information that we need, and processing phase, where we will apply process mining techniques in order to mine the event log.

2.3.1 Pre-processing. Clear scenery, easy discover

As we just introduce, to avoid as much noise as we can, we will clean the log using the filtering plugins that ProM offers. In our case, we will apply a simple heuristic filter which provides to us the possibility of selecting the different classes we would like to filter by in three steps:

1. By life cycle transition of event: we can filter just the start events, only the end events or no filter (both of them will be shown).

2. By event on each life cycle transition selected: we can choose which started and/or end events do we want to keep. To select that we must indicate a percentage threshold, so the events with a frequency higher than it are retained. It means that if there is an event which has been repeated the 80% of the time and our threshold is set up at 70%, it will be the one selected, because over 70% of the traces start (or end, depending on the filter we are) with that event.

3. By event: we can also set a threshold to select the events we want to conserve depending on their frequency.

After that, we will be able to navigate through the log inspector screens that will display different information for helping us to understand the processes behind the event logs we passed as parameters.

As Figure 2.3 shows, we can know the number of completed cases as well as the number or type of the events. We can also select a specific instance to discover better all the steps it takes (Figure 2.4). Also, we can see a complete and detailed summary about all the events we are processing (Figure 2.5).

Figure 2.3: PROM Dashboard

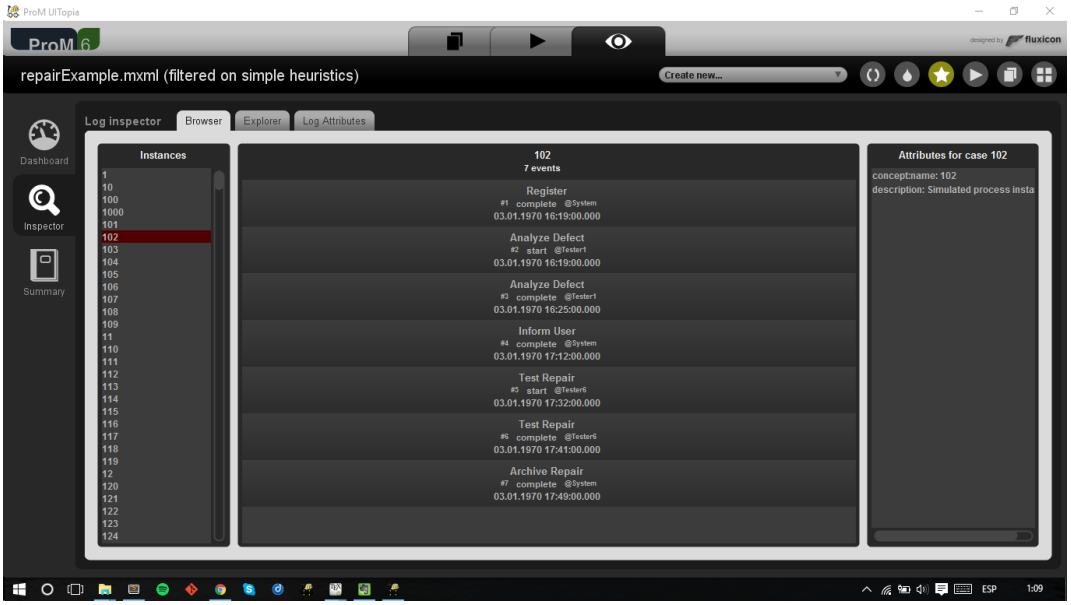


Figure 2.4: PROM Inspector

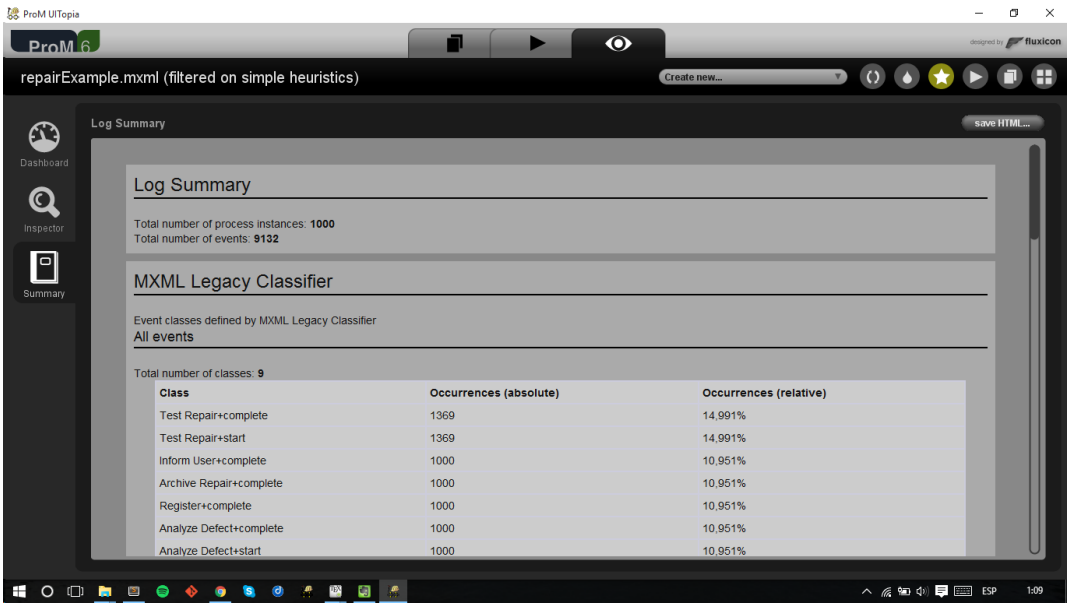


Figure 2.5: PROM Summary

2.3.2 Processing. The ability to answer questions we are wondering

As we said in the process mining section, there are three kinds of plugins: those which are based on data in the event log, which are called *"Discovery plugins"*; those which check how much the data in the event log matches the prescribed behaviour in the deployed models, which are called *"Conformance plugins"*; and those that need both a model and its logs to

discover information that will enhance this model, *"Extension plugins"*.

So once the event log has been pre-processed, we can do discovery actions, as mining the control flow perspective of process models, mining information regarding certain aspects of cases, or even mining information related to the roles/employees in the event log.

For example, the Mine for a Petri Net using Alpha-algorithm plugin allows us to know how are the cases actually being executed beholding the *Petri net* which results its output. Its input should be the previously cleaned event log. In this way we can observe lineal processes, loops, iterations or more which will help us to answer the question raised before (see Figure 2.6). Note that depending on the plugin we use to mine that data the result diagram can be diverse, or the analysis done more effective for real-life logs. For example, for real logs (determined by noise and size), alpha-algorithm has not a good result. However, ProM provides us several plugins to succeed with the maximum accuracy, like Heuristic Miner or Fuzzy Miner, which we will see on detail on chapter 3.

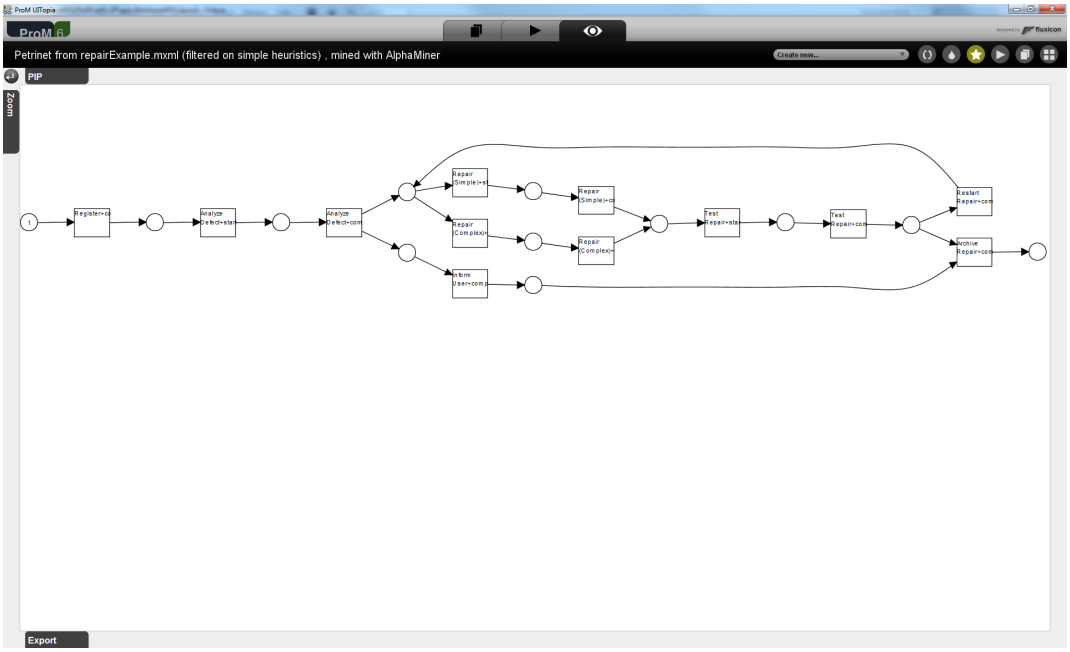


Figure 2.6: PROM Petri Net

We can also find the most frequent paths there are in the process, the distribution of all cases over the different paths through the process, or even we have the possibility of selecting a subset of traces where particular paths were executed. To answer those questions ProM offers the *pattern abstractions visualization* (Figure 2.7). That is a visualization option with different customizable parameters we can adjust to look for patterns on the event log, like the kind of pattern we want find, metrics, frequency...etc.

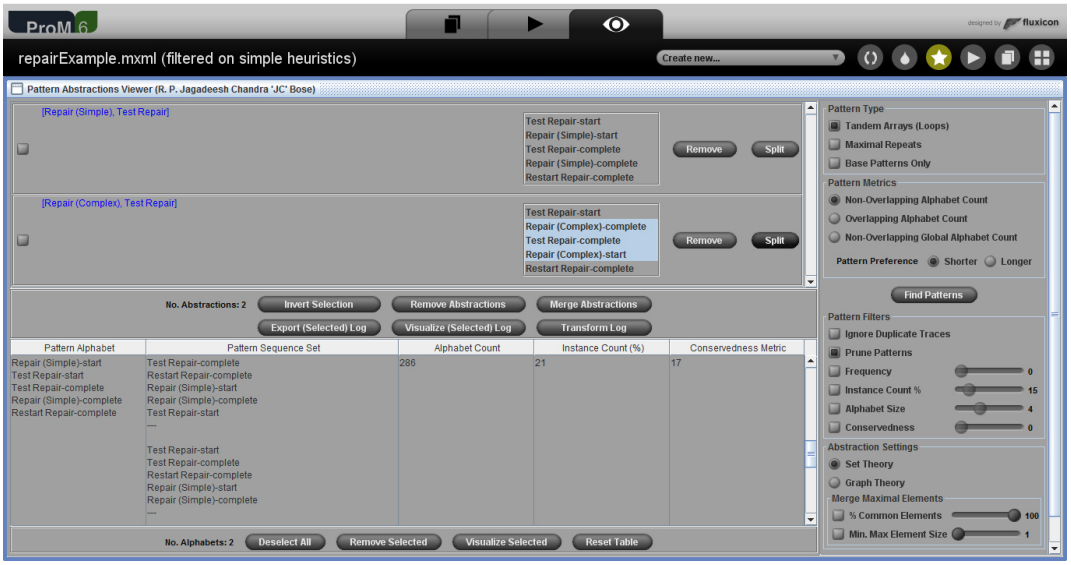


Figure 2.7: PROM Pattern Abstracts

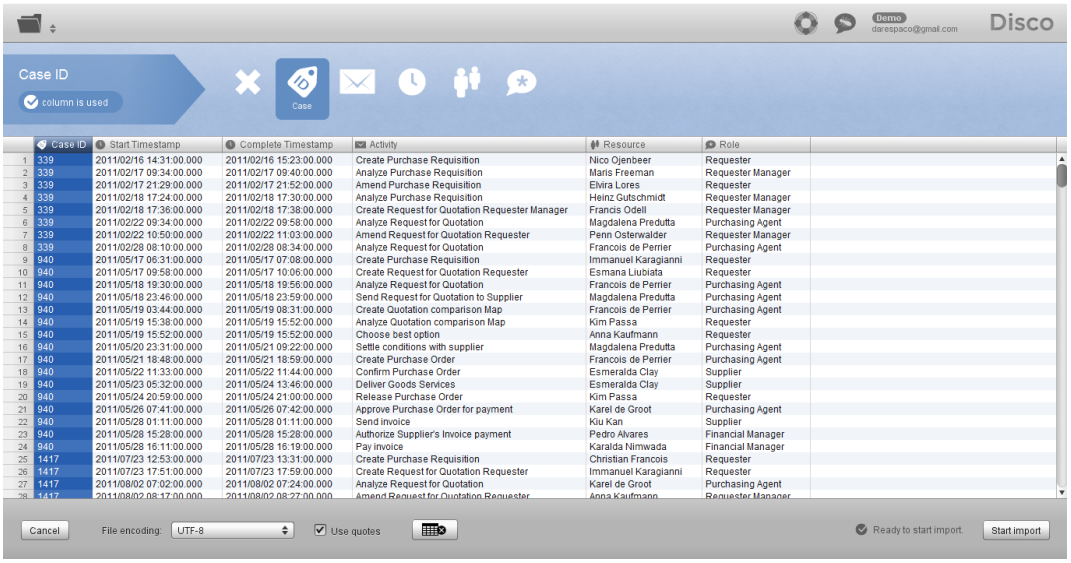
If we are wondering about more social or organizational aspects of the company like how many people are involved in a specific case, or dependencies among people, who subcontracts work to whom...etc, we have also plugins which extracts that information from the event logs. They are called *Social Network Miner* plugins. More information about this plugin can be found in [10]

2.4 Process Mining Tools: Disco

The other common process mining tool we can use is *Disco* [11]. It is a professional tool provided by Fluxicon. Is not open-source, so if we want to enjoy of all its features we must buy the licence. However, it provides also a free version that allows us to play with it so we can discover whether it fits to our needs (always with an event log that does not exceed a hundred traces).

2.4.1 Pre-processing. Clear scenery, easy discover

This tool admits CSV and XLS files to be loaded, so we can choose the format which we want to save in our event logs. Once we open the file, we must prepare the logs to be processed. Disco gives us the chance to specify the value and the format of each column, as we can see on Figure 2.8. We can choose between some predefined roles: case ID, activity, timestamp and resource, or even add those we want.



	Case ID	Start Timestamp	Complete Timestamp	Activity	Resource	Role
1	339	2011/02/16 14:31:00.000	2011/02/16 15:23:00.000	Create Purchase Requisition	Nico Ojenbeer	Requester
2	339	2011/02/17 09:34:00.000	2011/02/17 09:40:00.000	Analyze Purchase Requisition	Maris Freeman	Requester Manager
3	339	2011/02/17 21:29:00.000	2011/02/17 21:52:00.000	Amend Purchase Requisition	Elvira Lores	Requester
4	339	2011/02/18 17:24:00.000	2011/02/18 17:30:00.000	Analyze Purchase Requisition	Heinz Gutschmidt	Requester Manager
5	339	2011/02/18 17:36:00.000	2011/02/18 17:38:00.000	Create Request for Quotation Requester Manager	Francis Odell	Requester Manager
6	339	2011/02/22 09:34:00.000	2011/02/22 09:58:00.000	Analyze Request for Quotation	Magdalena Predutta	Purchasing Agent
7	339	2011/02/22 10:50:00.000	2011/02/22 11:03:00.000	Amend Request for Quotation Requester	Penn Osterwalder	Requester Manager
8	339	2011/02/28 08:10:00.000	2011/02/28 08:34:00.000	Analyze Request for Quotation	Francois de Perrier	Purchasing Agent
9	940	2011/05/17 06:31:00.000	2011/05/17 07:08:00.000	Create Purchase Requisition	Immanuel Karagianni	Requester
10	940	2011/05/17 09:58:00.000	2011/05/17 10:06:00.000	Create Request for Quotation Requester	Esmana Liubiata	Requester
11	940	2011/05/18 19:30:00.000	2011/05/18 19:56:00.000	Analyze Request for Quotation	Francois de Perrier	Purchasing Agent
12	940	2011/05/18 23:46:00.000	2011/05/18 23:59:00.000	Send Request for Quotation to Supplier	Magdalena Predutta	Purchasing Agent
13	940	2011/05/19 03:44:00.000	2011/05/19 08:31:00.000	Create Quotation comparison Map	Francois de Perrier	Purchasing Agent
14	940	2011/05/19 15:38:00.000	2011/05/19 15:52:00.000	Analyze Quotation comparison Map	Kim Passa	Requester
15	940	2011/05/19 15:52:00.000	2011/05/19 15:52:00.000	Choose best option	Anna Kaufmann	Requester
16	940	2011/05/20 23:31:00.000	2011/05/21 09:22:00.000	Settle conditions with supplier	Magdalena Predutta	Purchasing Agent
17	940	2011/05/21 18:48:00.000	2011/05/21 18:59:00.000	Create Purchase Order	Francois de Perrier	Purchasing Agent
18	940	2011/05/22 11:33:00.000	2011/05/22 11:44:00.000	Confirm Purchase Order	Esmeralda Clay	Supplier
19	940	2011/05/23 05:32:00.000	2011/05/24 13:46:00.000	Deliver Goods Services	Esmeralda Clay	Supplier
20	940	2011/05/24 20:59:00.000	2011/05/24 21:00:00.000	Release Purchase Order	Kim Passa	Requester
21	940	2011/05/26 07:41:00.000	2011/05/26 07:42:00.000	Approve Purchase Order for payment	Karel de Groot	Purchasing Agent
22	940	2011/05/28 01:11:00.000	2011/05/28 01:11:00.000	Send invoice	Kiu Kan	Supplier
23	940	2011/05/28 15:28:00.000	2011/05/28 15:28:00.000	Authorize Supplier's Invoice payment	Pedro Alvares	Financial Manager
24	940	2011/05/28 16:11:00.000	2011/05/28 16:19:00.000	Pay invoice	Karalda Nimwada	Financial Manager
25	1417	2011/07/23 12:53:00.000	2011/07/23 13:31:00.000	Create Purchase Requisition	Christian Francois	Requester
26	1417	2011/07/23 17:51:00.000	2011/07/23 17:59:00.000	Create Request for Quotation Requester	Immanuel Karagianni	Requester
27	1417	2011/08/02 07:02:00.000	2011/08/02 07:24:00.000	Analyze Request for Quotation	Karel de Groot	Purchasing Agent
28	1417	2011/08/02 08:17:00.000	2011/08/02 08:27:00.000	Amend Request for Quotation Requester	Anna Kaufmann	Requester Manager

Figure 2.8: DISCO PreProcessing

We can also select which columns we want to be part of the processing, deleting those that we consider not relevant. Note that there are obligatory roles we have to assign (like case ID, activity or timestamp) before the processing.

2.4.2 Processing. The ability to answer questions we are wondering about

Once the logs are prepared, it is time to discover what is it telling us. We have three different views through which we can get a lot of information about our processes: map view, statistics view and cases view.

1. **Map view**. That is the default view we find after start the processing action (see Figure 2.9). Here we can see a detailed map where each line width, number or colour is thought so we can get a lot of information about the process flow, activity frequencies, frequency of connections...etc. We can also adjust the level of detail through two sliders which set the percentage of the frequency of the activities and paths we want to see. So we can choose if we want to have a far view or a closer one.

   This view provide us more tools to filter, search, or select the kind of frequency we want to know, but maybe the most remarkable one is the animation tool, where we can see the process on execution and get an idea also the time each activity is taking. Disco is based on the Fuzzy Miner, that was the first mining algorithm to introduce the *map metaphor* to process mining. Combining that algorithm with their own knowledge and experience, they optimize the way they are showing this process map.

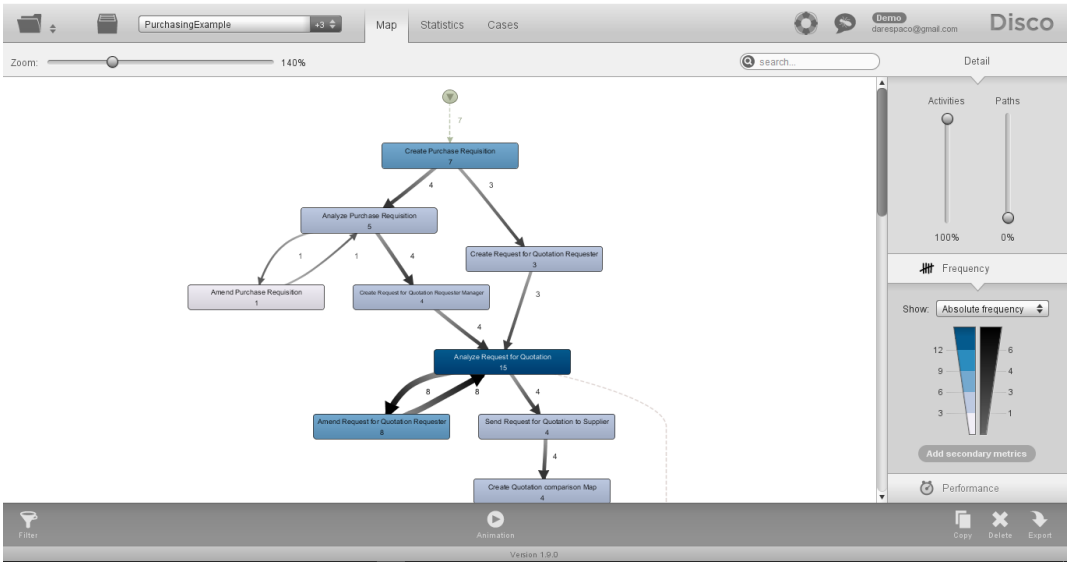


Figure 2.9: DISCO Map view

2. **Statistics view**. This view shows (Figure 2.10) overview information about the logs: number of events, number of cases, timestamp, mean case duration, median case duration...etc, accompanied by graphics where we can see clearly how this info is related with each other.

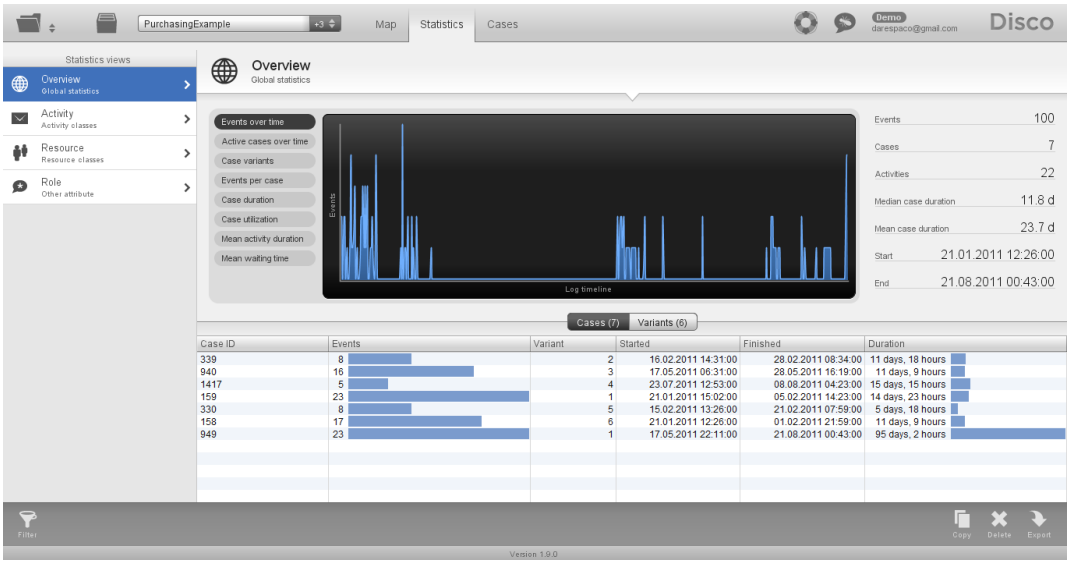


Figure 2.10: DISCO Statistics view

3. **Cases view**. Here we can have a look at the activities flow of each case (Figure 2.11). Disco groups themes into variants with the cases with the same flow. We can see specific information about each activity, as well as a graphic clarifying the results.

We can discover, for instance, if it is frequent to finish the flow with a specific activity and wonder why.

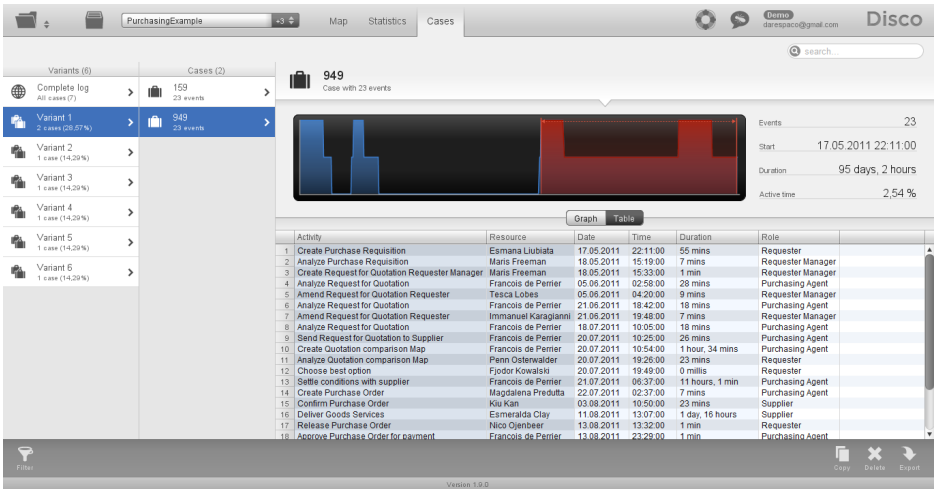


Figure 2.11: DISCO Cases view

Using Disco we can apply filters to discover, for instance, how many cases take longer than a period so which are the slowest cases (applying for that a performance filter). For that case, after applying the performance filter we can play with the map view again to see which activity is the cause of that retard, the common known *bottleneck*. We can see an example of that in Figure 2.12.

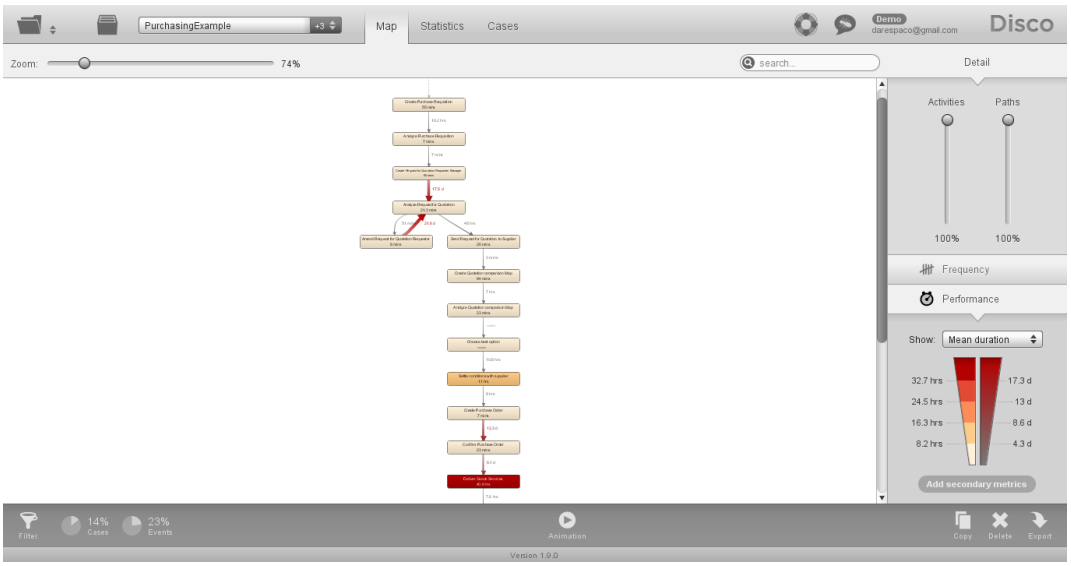


Figure 2.12: DISCO Performance filter

We can even discover if there are some derivations from the original process and change the operational system preventing the violation or providing targeted training.

Last but not least, there's also the option of knowing more about the organizational flow (see Figure 2.13), so we can find inefficiencies on the organizational units.

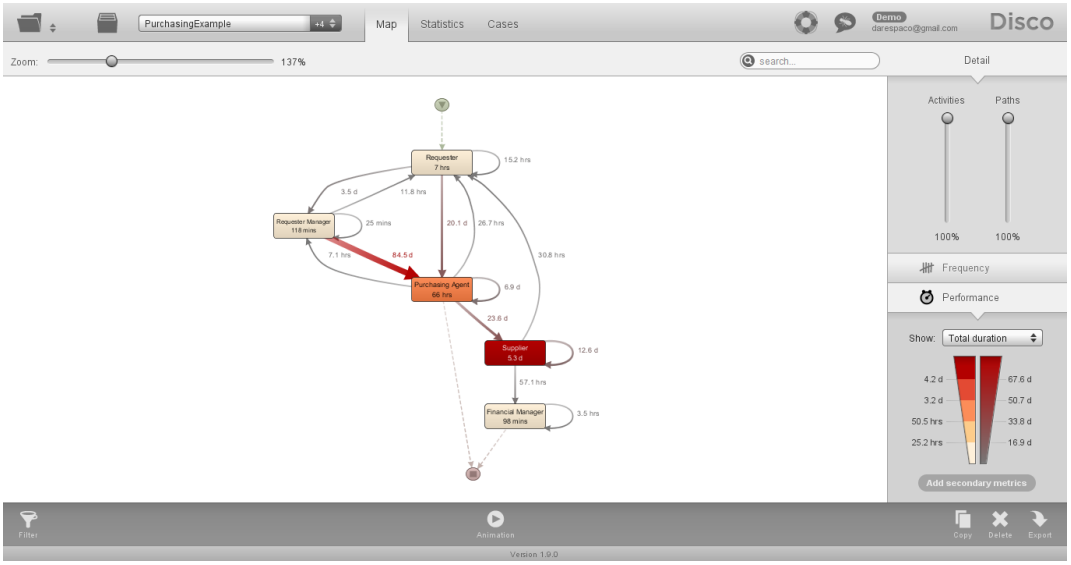


Figure 2.13: DISCO Organizational flow

To summarize, using Disco tool we will be able to answer these three general questions:

- How does the process actually look like?
- Are there deviations from the prescribed process?
- Do we meet the performance targets?

2.5 Comparative between both technologies

After this general picture of these two tools, we will propose our conclusions, supporting us on the table 2.1

As we have seen, variety of plugins are considerable higher on ProM. For example, for the most common operation in process mining: process discovery, ProM gives us the chance to apply different algorithms such as Heuristic, Alpha-miner, Fuzzy, and others which can be find on [9], whilst Disco allows us only to use the Fuzzy Miner algorithm.

Although ProM can support much more operations and can be used with a variety of mining algorithms, its weakest point is its interface. It is not intuitive at all, presents a lot of inconsistencies and results cannot be observed with clarity. On this aspect, Disco is

Features	**ProM**(v 6.5.1)	**Disco**(v 1.9.0)
Import Type	MXML, XES	CSV, XLS, MXML, XES and FXL
Import Log	unlimited	up to 5 million events
License	Open Source	Evaluation / commercial
Output model notation	BPMN, WF, Petri nets, EPCs, transition systems, heuristics	Fuzzy model
Supported	Standalone desktop version	Standalone desktop version
Filtering data	Yes	Yes
Process discovery	Yes	Yes
Conformance checking	Yes	No
Social network mining	Yes	No
Decision rule mining	Yes	No
Process visualization	Yes	Yes
Performance reporting	Yes	Yes
Discriminative rule mining	Yes	No
Trace clustering	Yes	No
Delta analysis	Yes	Yes

Table 2.1: Comparison between ProM and Disco [5].

much more suitable due to its simplicity to use and fast processing of event logs, presenting results in a very friendly interface.

Something to note is that both tools can be used together while making the process discovery, bottleneck analysis and later can be exported in XES format into ProM for further analysis.

Finally, for this project we have chosen ProM tool for two reasons: the huge number of plugins which we can mine with the event log joined to the limitation on the number of events that free version of Disco has.

CHAPTER 3

Methodology and Algorithms for Process Mining

This chapter describes how to follow a methodology in order to apply process mining techniques to an event log with ProM. Step by step, it explains the main points and the best plugins we can use. The chapter also does an analysis of the two main mining algorithms for real logs: Heuristics and Fuzzy, exposing its fundamentals and its parameters.

3.1 Methodology for doing a Process Mining analysis

Once we have understood what process mining is and how are the main Process Mining tools working, we will proceed to explain how can we apply this technology to analyse a particular dataset. Based on [12]], we can define some steps to accomplish a good process mining analysis.

1. **Previous work.** The first step is preparing the field. To do this, it is needed to pick the process we want to analyse and define the questions we want to answer: do we want to discover the process? Do we want to obtain some metrics like the most frequent paths or if there is any loop in the process? Those questions should be clearly established, because it will determine the tools and algorithms we will apply.

2. **Extracting and preprocessing the data.** The data extracted from the recording systems could be in a different format from the one we need. So we need to convert them into MXML [8]. The correct format we should get at the end consists of three required rows: case unique identifier, activity name and time stamp. We can add then more rows in order to provide more useful information, but at least those 3 columns must be defined.

3. **Create the Event Log MXML input file.** It is not a hard task to find tools which allows us to convert a file from a CSV to MXML (or even source formats like Microsoft Access Database Tables). XESame, Disco, ProM import are some applications we can use to do this conversion. As we saw in the analysis of Disco and ProM made in chapter 2, they permit a high customization of the event logs, to define the format of each row, or even the XES label we want to assign to each row.

   After doing it, our file should have a clear structure based on labels which will define each trace, each event within a trace, its duration...etc. It only remains to import it in ProM and to start to play.

4. **Inspect the Event Log.** Just we import the event log, we can see some useful general information about the process. On the screen that is depicted on Figure 3.1

and Figure 3.2 we can discover the number of cases, the number of events, the number of different events, the average number of events per case and its distribution...etc. You may even see the events in its specific case and its timestamp.

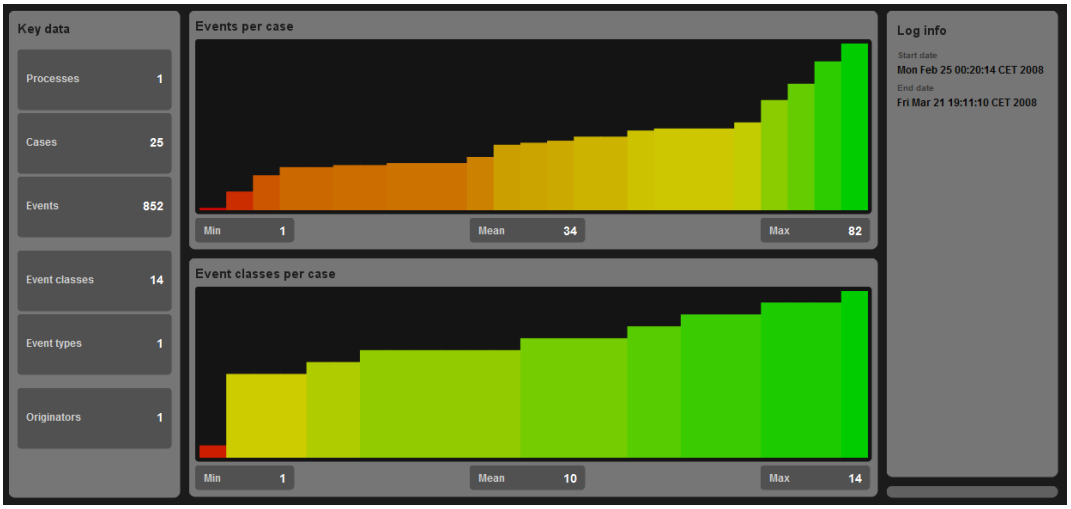


Figure 3.1: Main Log inspection screen in ProM

Class	Occurrences (absolute)	Occurrences (relative)
Hall-Bathroom door	193	22,653%
ToiletFlush	121	14,202%
Hall-Toilet door	97	11,385%
Fridge	87	10,211%
Hall-Bedroom door	61	7,16%
Plates cupboard	58	6,808%
Groceries Cupboard	50	5,869%
Frontdoor	45	5,282%
Cups cupboard	33	3,873%
Freezer	32	3,756%
Pans Cupboard	28	3,286%
Microwave	23	2,7%
Dishwasher	14	1,643%
Washingmachine	10	1,174%

Figure 3.2: Main Log inspection screen in ProM

Also in the inspection log we can use more tools available which will provide different points of view of the log. So, we can visualize an inductive mine which show us an interactive mined *Petri Net*; we can discover and define pattern abstractions in order to simplify the model avoiding spaghetti-like processes; or even we can see an X-dotted chart to discover in a view which events are taking place in each hour, weekday or month.

Definitely, that first inspection can provide us a huge amount of information about the Event Log which allows us to understand better the mining process we will do in

the next step.

5. **Mine a Process Model.** The most useful plugins in ProM to mine the control-flow perspective of a process models are the Heuristic Miner and the Fuzzy Miner. Those algorithms will be explained on the next section, in order to understand the different capabilities they provide.

   To use *Heuristic Miner* plugin (more detailed on the next Sect. 3.2), helpful when we have real-life data with not too many events, or when we need a *Petri net* model for further analysis in ProM, it is required to set some thresholds which will help to optimize the fitness value:

   - *Dependency*: activities with a higher dependency value than this can be accepted.
   - *Relative-to-best*: those activities which can be accepted, and in which the difference between its dependency and the highest dependency does not exceed the value of this threshold, are accepted.

Figure 3.3 shows partially the output of the *Heuristic Miner* plugin, where we can found boxes that represent each activity in the process. All boxes contain a number which means the relative frequency of the task, whilst each path between boxes has associated another number representing the dependency between tasks, that is, how possible is to arrive to one activity from another. This plugin can also change the thick of the paths and boxes depending on how strong is the dependency between them. As we can see in the image, fitness value is indicated on the bottom. A notable weakness of this plugin is the fact that to change the thresholds the plugin must be executed again.

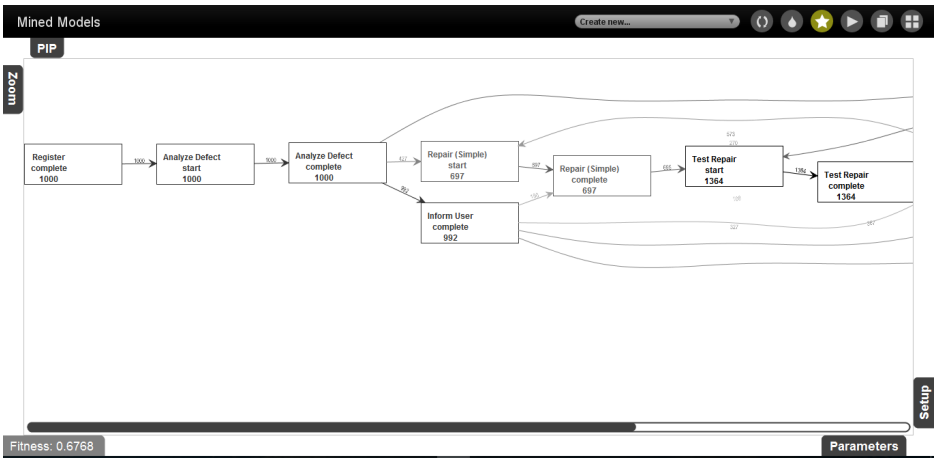


Figure 3.3: *Heuristic Miner* plugin output

The Fuzzy Model, useful to work with complex and unstructured log data, or when we want to simplify the model in an interactive manner, can be partially found in Figure 3.4. It uses significance/correlation metrics to interactively simplify the process model at desired level of abstraction. Compared to the Heuristic miner it can also leave out less important activities if you have hundreds of them. Furthermore, it provides the chance of adjusting the parameters in real time.

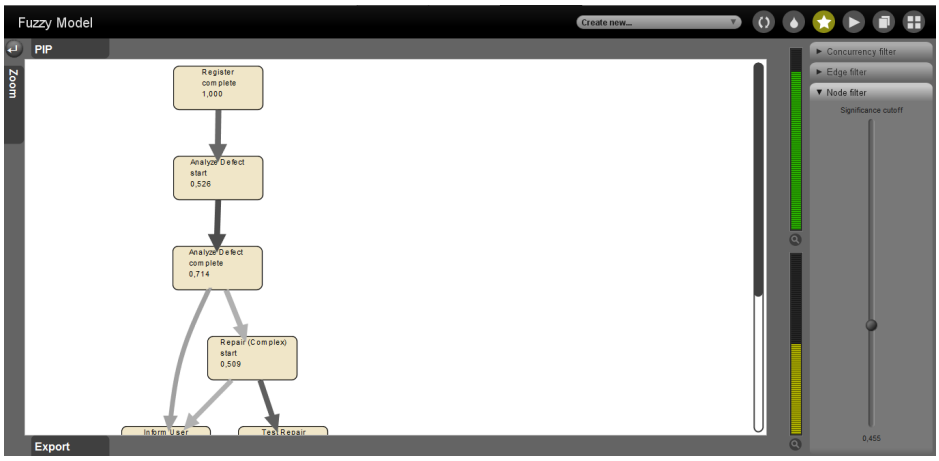


Figure 3.4: *Fuzzy Miner* plugin output

6. **Animate the Process Model.** To get a major clarity in the visualization of the process, ProM also provides an interactive tool to see in real time how the activities are being executed. To use that plugin, we need first to apply the plugin *Select Best Fuzzy Instance* to the output of the *Fuzzy Miner* plugin applied before. Then, we will use the generated file to apply the *Animate Event Log in Fuzzy Instance* plugin.

3.2 Algorithms

On this section we will present the most common algorithms used to mine process models: Heuristic Miner and Fuzzy Miner. Thus, we will expose its fundamentals, when is better to use one or another, and we will also explain the parameters which determine their fitness with the original log as well as their reliability.

3.2.1 Heuristic Miner

As we said before, *Heuristic Miner* plugin mines the control-flow perspective of a process model. To do so, it only considers the order of the events within a case. In other words,

the order of events among cases is not important.

To understand how Heuristic Miner is working, let's consider, following the notation on [3], T as a set of activities, k as an event trace, and W as an event log. So W could be, for instance, a multiset W = [ABCD, ABCD, ACBD, ACBD, AED]. We will also define the following notations:

- **A >w B** means that A appears always followed by B.
- **A →w B** means that A appears always followed by B, but B never appears followed by A, so the relation is direct. For instance, A →w B would be correct, whilst B →w C would be incorrect, because B >w C and C >w B.
- **B ||w C** suggest a potential parallelism between activities B and C, as we can note in W log.
- **A #w B** gives pairs of transitions that never follow each other directly, so there is no direct dependency relations and parallelism is unlikely.

Dependency between activities can be defined as the relation between two activities in which one of them uses to follow the other one. It is a key factor due to find a process model the log should be analysed for causal dependencies.

It should be noticed that the formal approach presupposes perfect information: a complete log without noise. However, real logs do not usually present those features. That is why formal approaches (algorithms like α-algorithm uses this approach) are not good enough for real logs. For example, in a log with a huge number of events, if a relation A →w B holds between two activities, just one B >w A relation based on an incorrect registration will prevent a correct conclusion.

It is for that reason why heuristic miner considers the frequency of the relations, providing a much better analysis of huge and noise logs (how should be real logs).

Thus, Heuristic Miner algorithm can be explained in three steps [3]:

Step 1: mining of the dependency graph

The starting point of the Heuristics Miner is the construction of a so called dependency graph. A frequency based metric is used to indicate how certain we are that there is truly a dependency relation between two events A and B (notation A ⇒w B). The calculated ⇒w values between the events of an event log are used in a heuristic search for the correct dependency relations.

Let's see how is $\Rightarrow$w defined. Being $|a >\text{w } b|$ the number of times a >w b occurs:

$$a \Rightarrow\text{w } b = (|a >\text{w } b| - |b >\text{w } a|)/(|a >\text{w } b| + |b >\text{w } a| + 1)$$

Note this value should always be between -1 and 1.

To understand this formula let's use an example extracted from [3]. Supposing we are working with a process log in such 5 traces holds that activity A is directly followed by activity B, but the other way around never occurs. A $\Rightarrow$w B = 5/6 = 0.833, which means we are not completely sure of the dependency relation (5 is a too low number, them can be caused by noise). However, if instead of 5 traces we see 50 cases in which this relation holds, even if we find one which holds the opposite relation, we would have A $\Rightarrow$w B = 49/52 = 0.94, which means we can be pretty sure of that dependency relation.

We know that each non-initial activity must have at least one other activity that is its cause, and each non-final activity must have at least one dependent activity. Using this information in the so called all-activities-connected heuristic, we can take the best candidate (with the highest A $\Rightarrow$w B score).

For instance, suppose we have a causal activity matrix (Figure 3.5) with the $\Rightarrow$w values resulting from the log W = [ABCD^9, ACBD, AED^9, ABCED, AECBD, AD].

$\Rightarrow_W$	A	B	C	D	E
A	0.0	0.909	0.900	0.500	0.909
B	0.0	0.0	0.0	0.909	0.0
C	0.0	0.0	0.0	0.900	0.0
D	-0.500	-0.909	-0.909	0.0	-0.909
E	0.0	0.0	0.0	0.909	0.0

Figure 3.5: Causal Activity matrix [3]

After applying the all-activities-connected heuristic on this matrix, we will discover A as the initial activity, inasmuch as it is the only column without positive values (so it does not depend on any other). To look for the activities which depends on A, we can look for the highest values on A row. So, we found B and E. We first choose B, and we find A as the cause of B. Then we note that D is the depending activity of B (the one with the highest value on B row). Same process must be done for E and C activities, resulting the graphic depicted on Figure 3.6, which is called *dependency graph*, where the numbers inside the boxes represents the relative frequency of each activity, whilst the number in the arcs represents the reliability of each causal relation and also the frequency.

Step 2: AND/XOR-split/join and non-observable tasks

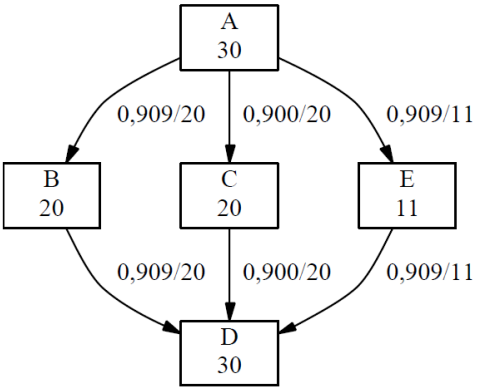
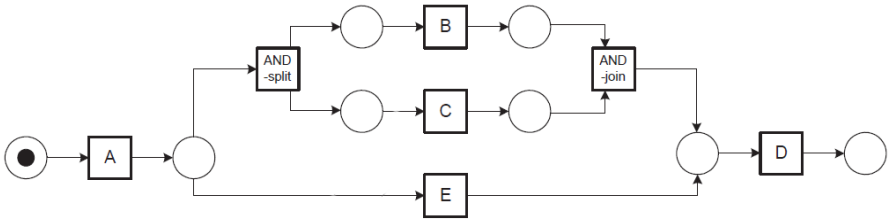


Figure 3.6: Dependency graph example [3]

Might be in our process two activities executed in parallel. For that, two non-observable activities should be considered (AND-split and AND-join). Those activities are not in the event log, so it is difficult to mine them. To treat them in a Heuristic Miner, it is used the so called *Causal Matrix* to depict the corresponding *Petri net* (see Figure 3.7).



ACTIVITY	*INPUT*	*OUTPUT*
A	$\emptyset$	$(B \vee E) \wedge (C \vee E)$
B	A	D
C	A	D
D	$(B \vee E) \wedge (C \vee E)$	$\emptyset$
E	A	D

Figure 3.7: Causal Matrix and *Petri Net* which is representing [3]

The underlying idea is relative simple. Let we start with a simple example. The dependency graph of Figure 3.6 already gives the information that activities B, C and E are in the output expression of activity A (as sown in Figure 3.7). If two activities (e.g. B and E) are in the AND-relation, the pattern BE can appear in the event log. If two activities (e.g. B and C) are in the XOR relation the pattern BC is not possible.

Step 3: Mining long distance dependencies

In some process models the choice between two activities is determined on previous choices made in other parts of the process model. For example, in the process showed on Figure 3.8, after activity D, the election between E or F activities depends on the previous choice among B or C. This fact implies a difficult mining. Mining an event log generated by the process model of Figure 3.8 with the Heuristics Miner as presented so far, will result in a dependency graph without the B to E and C to F connection.

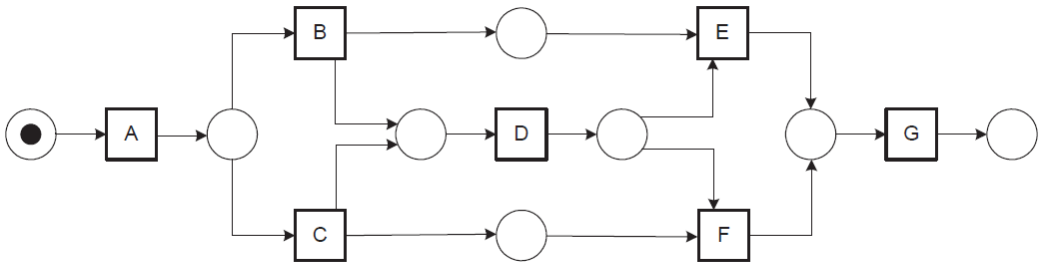


Figure 3.8: A process model with a non-free-choice construct [3]

3.2.2 Fuzzy Miner

The Fuzzy Miner is suitable for mining less-structured processes which exhibit a large amount of unstructured and conflicting behaviour, those called *spaghetti processes*. For this reason Fuzzy Miner should be able to provide a high-level view on the process, abstracting from undesired details.

On [4] can be found an awesome example to explain Fuzzy Miner's philosophy: the field of cartography. There we have locations, which could be identified with the activities in our process, and we also have traffic connections between them (also paths between activities).

To represent those elements into a map, as we see in Figure 3.9, it is common to come up with to simplify, the use of four concepts we define below.

Figure 3.9: Example of a road map [4]

- **Aggregation:** used to limit the number of information items displayed, maps often shows coherent clusters of low-level detail information in an aggregated manner.
- **Abstraction:** insignificant information not relevant must be omitted.
- **Emphasis:** the most important information must be highlighted by visual means.
- **Customization:** the level of detail must depend on the context we want to analyse.

These concepts are universal, well-understood, and established. But to see their application to the process model, [4] defines two fundamental metrics: significance and correlation.

Significance measures the relative importance of behaviour. That means the level of interest we have in events or in the dependency among them. Significance can be determined both for event classes and precedence relations over them. *Correlation*, on the other hand, measures how closely related two events following one another are. More closely correlated events are assumed to share a large amount of their data, or have their similarity expressed in their recorded names.

Now we can classify processes as *highly significant*, *less significant but highly correlated* or *less significant and lowly correlated*. The first type will be preserved, the second type will be aggregated, and the third one will be abstracted from.

So, it results a simplified process model as in Figure **??**, where bright square nodes represent significant activities and the darker octagonal node is an aggregated cluster of three less-significant activities. All nodes are labelled with their respective significance, with clusters displaying the mean significance of their elements. The brightness of edges between nodes emphasizes their significance, i.e. more significant relations are darker. Edges are also labelled with their respective significance and correlation values. By either removing or hiding less significant information, this visualization enables the user to focus on the most interesting behaviour in the process.

CHAPTER 4

Applications of PM for Task Automation and Agent-based Social Simulation

In this chapter we would like to demonstrate how, applying the just seen process mining tools and techniques to a dataset extracted from a SmartHome, we can discover patterns, unusual activities and other relevant results which could be used for intelligent applications such task automation or social simulation, among others.

4.1 Use case

Our starting point will be a use case proposed by Tim van Kasteren in [13], where he applied two probabilistic models to a dataset (hidden Markov and Conditional Random Fields) in order to recognize activities of a 26-years-old man living in a SmartHome for a period of 25 days. However, literature about how to apply process mining techniques to get these results is almost non-existing.

The SmartHome was a usual flat composed by six rooms. Within four of them were installed a set of 14 change-state sensors placed as is depicted on Table 4.1 and in Figure 4.1.

In the kitchen	**In the hall**	**In the toilet**
Microwave	Hall-Bedroom Door	Toilet Flush
Freeze	Hall-Bathroom Door	-
Fridge	Hall-Toilet Door	-
Dishwasher	Frontdoor	-
Cups Cupboard	-	-
Groceries Cupboard	-	-
Pans Cupboard	-	-
Plates Cupboard	-	-
Washing Machine	-	-

Table 4.1: Table of sensors installed on Kasteren SmartHome

After 25 days of normal life, sensors were collected 1319 events about inhabitant's activities. Our goal is to find, using process mining techniques, regular patterns on the behaviour of the man, unusual activities, the relation of each activity in the time. Definitely, any relevant information about the subject and his life during the period of the experiment.

Figure 4.2 represents the architecture of the process we followed to achieve our goal.

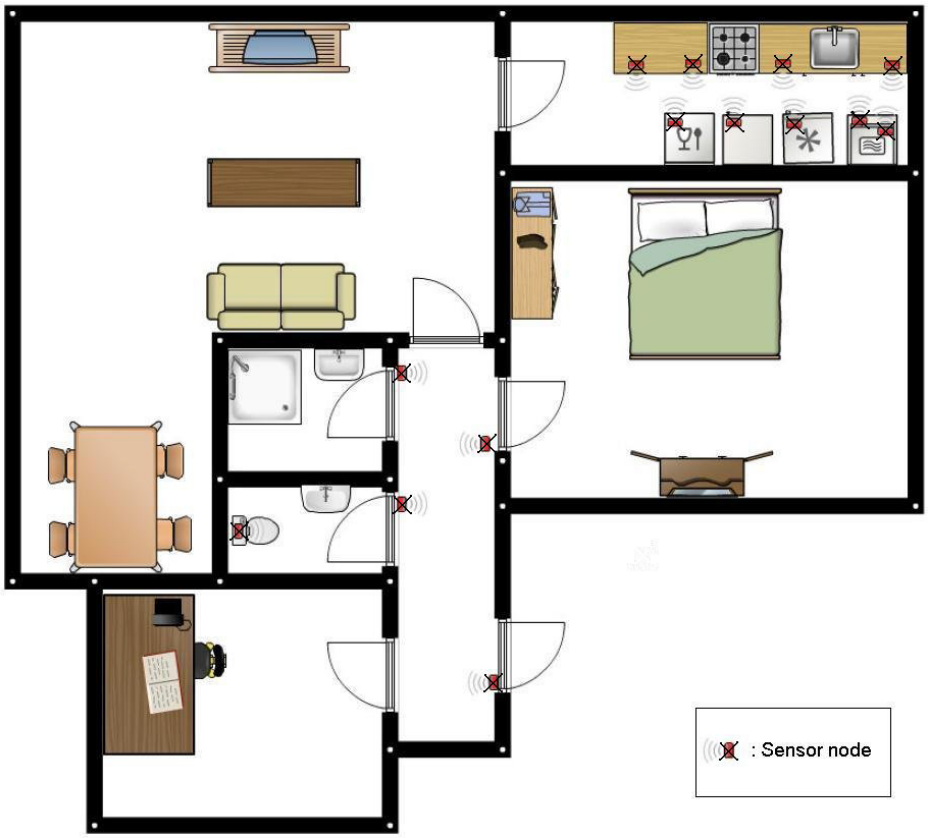


Figure 4.1: Floorplan of the house, red rectangle boxes indicate sensor nodes

Starting from the dataset mentioned above, it has been followed a data adaptation preprocess, based in a python script in order to fit the logs into the ProM fields requirements. So, we added a new column which represents the case (it is explained with more details on the next Section 4.2). After that, it has been used ProM to convert this logs, which were in CSV format into a proper format as XES, in order to be able to analyse them using three different plugins focused on particular aspects: *Simple Filter Heuristic*, to discover general information and time patterns; *Heuristic Miner*, to mine the log and get a heuristic model; and *Discovery Matrix*, to see the dependencies among activities.

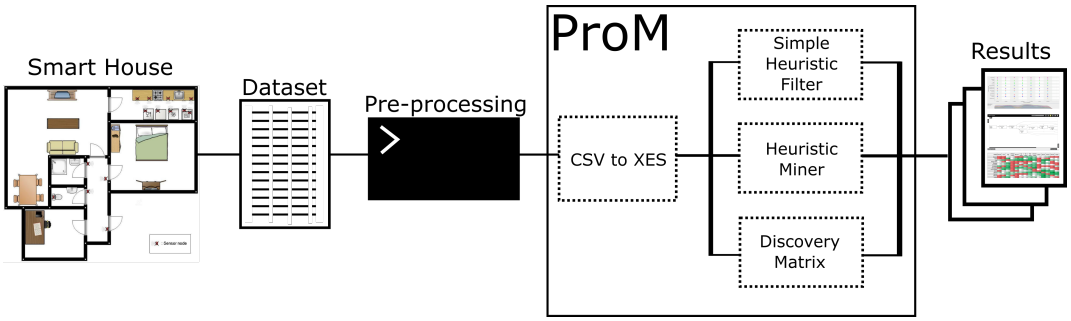


Figure 4.2: Architecture to analyse a dataset using PM

4.2 Dataset

Kasteren work provides two different datasets: the first one, obtained by the activity registration directly from the user by mean of a bluetooth headset (he registers all the events with his voice saying some keywords), and the second one, obtained by the sensors installed on the SmartHome. For this analysis we will use the second dataset, from now *senseData* because of the high level of noise registered on the first one.

The senseData is a *.txt* file which starts listing all the sensors available and assigning them an ID. Then, in the file are registered 4 fields: Start time, End time, ID and Val.

Start and End time represents the time in which sensor's value comes to 1 and comes back to 0, respectively, and has a format defined by *dd-MMM-yyyy HH:mm:ss*; ID field represents the sensor in which the activity is registered; and Val has always a value of 1. For instance, the first line, which is

Start	End	ID	Val
25-Feb-2008 00:20:14	25-Feb-2008 00:22:57	24	1

would mean that the sensor with the ID 24 (Hall-Bedroom Door) has been activated from 20:14 to 22:57 on February 25th, so the door was opened at 20:14 and was closed again at 22:57.

To be able of applying the ProM filters and algorithms properly to this senseData, as we said before, it is needed to pre-process the file. For this, we will run a python script which creates a CSV file with 4 fields: Start and end time don't change; sensor name, that replaces the previous sensor ID; and case ID, which is defined by ourselves following this criterion: activities belonging to a specific case ID are those which were carried out between

the 04:00 AM of one day and 04:00 AM of the next day (this criterion is based on the needed to define a coherent start and end points according to the scope of the data).

The script has an additional function which removes repeated activities. That task has been needed due to the noise introduced by the sensors, which used to register more than once each activity, giving incorrect information about the processes.

After that, it is obtained a CSV reduced to 853 entries (instead of the original number 1319), but containing well structured information ready to be discovered. So let's start with the experiments.

4.3 Experimental results

Now, starting from the use case just described on Section 4.1, we will apply the ProM methodology explained on Chapter 3 in order to achieve the goals of this project, that's it: to get relevant results about processes and patterns on the inhabitant's behaviour to apply them into AmI applications such task automation or agent-based simulation.

To import the log into ProM, we should first specify its format as well as few more parameters, such as the mapping of log columns to timestamps, cases and events (see Figure 4.3). Then ProM can automatically build the XES file, so once the XES file is ready, we can start to delve into it.

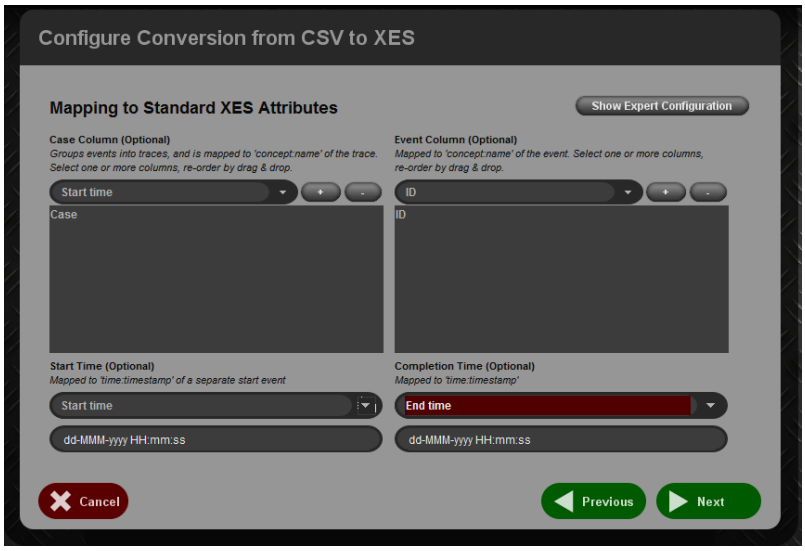


Figure 4.3: CSV to XES converter

Experiments will be conducted on the basis of answering three main issues which will be applied for later applications on this final project:

- Recognition of behaviour patterns.
- Discovery of process models.
- Analysis of activities duration and schedules.

4.3.1 Recognition of behaviour patterns

Firstly, to analyse the general output given by the log inspector will bring us some interesting information. For instance, we can note that the most frequent activity is such related to the Hall-Bathroom Door, with an occurrence of the 22,65%, whilst the most unusual activity is recorded by the sensor placed in the washing machine, which is fired 1,17% of the time. List on Figure 4.4 shows all the activities sorted out by occurrence.

Class	Occurrences (absolute)	Occurrences (relative)
Hall-Bathroom door	193	22,653%
ToiletFlush	121	14,202%
Hall-Toilet door	97	11,385%
Fridge	87	10,211%
Hall-Bedroom door	61	7,16%
Plates cupboard	58	6,808%
Groceries Cupboard	50	5,869%
Frontdoor	45	5,282%
Cups cupboard	33	3,873%
Freezer	32	3,756%
Pans Cupboard	28	3,286%
Microwave	23	2,7%
Dishwasher	14	1,643%
Washingmachine	10	1,174%

Figure 4.4: *Simple Heuristic Filter* parameters

For the raised issue, that information could seem not very useful, but it helps us to consider a general view of the inhabitant's behaviour.

Now, we can dive deeper by analysing the file with a plugin named *Discover Matrix*, which represents a causal activities' matrix. After configuring its parameters to set the mining algorithm we want to apply (we will apply heuristic algorithm for this case), we could see the output of the plugin. In order to provide a major clarity, it has been rewritten into the Excel table of Figure 4.5. As it is depicted there, some strong dependencies are held between activities. For instance, the relation among Toilet Flush and Hall-Bathroom Door has a value of 0,96, so that means we are quite sure that the next step after flush will be probably to open the Hall-Bathroom Door. Values on the table represent the certainty we have about a relation between two activities. From -1, which means to be sure that relation does not exist, to 1, which means to be sure that relation exists. A zero value means we

do not know. It is important note that we are not talking about frequency patterns, but dependence reliability.

FROM/TO	Dishwasher	Freezer	Fridge	Washing machine	Microwave	Cups cupboard	Pans cupboard	Plates cupboard	Groceries cupboard	Bedroom	Bathroom	Toilet door	Toilet flush	Frontdoor
Dishwasher	-1,000	-1,000	0,500	0,500	-1,000	-1,000	0,750	0,750	0,660	0,500	0,500	0,660	-1,000	-1,000
Freezer	-1,000	-1,000	0,880	0,500	0,800	0,833	0,500	0,875	0,750	-1,000	0,000	-1,000	0,500	0,500
Fridge	0,600	0,800	-1,000	0,750	0,857	0,917	0,833	0,917	0,875	-1,000	0,930	0,928	0,666	0,600
Washing machine	0,500	-1,000	0,500	-1,000	-1,000	-1,000	-1,000	-1,000	0,500	-1,000	0,800	0,500	-1,000	0,500
Microwave	0,500	0,600	0,850	-1,000	-1,000	0,500	-1,000	0,500	0,750	0,500	0,830	0,666	-1,000	-1,000
Cups cupboard	0,500	0,600	0,780	-1,000	0,667	-0,900	0,667	0,800	0,800	-1,000	0,500	0,500	0,500	-1,000
Pans cupboard	0,800	-1,000	0,880	-1,000	-1,000	0,666	-1,000	0,800	0,800	-1,000	-1,000	-1,000	-1,000	0,600
Plates cupboard	-1,000	0,800	0,910	-1,000	0,857	0,833	0,750	-0,800	0,930	-1,000	0,500	0,500	-1,000	-1,000
Groceries cupboard	0,600	0,875	0,750	-1,000	0,666	0,500	0,750	0,800	-1,000	0,500	0,750	0,666	0,500	-1,000
Bedroom	-1,000	0,500	0,500	-1,000	-1,000	0,500	0,500	0,666	-1,000	-0,770	0,960	0,875	0,800	0,500
Bathroom	0,666	0,500	0,750	0,666	0,666	0,666	0,800	0,875	0,750	0,944	-0,944	0,725	0,989	0,875
Toilet door	0,500	0,600	0,750	0,666	-1,000	-1,000	-1,000	0,750	0,500	0,727	0,880	-0,850	0,889	0,900
Toilet flush	-1,000	-1,000	0,660	-1,000	-1,000	0,500	0,500	0,667	0,500	0,660	0,960	0,800	-0,975	0,660
Frontdoor	-1,000	0,750	0,875	0,500	-1,000	0,500	0,500	0,600	0,660	0,500	0,930	0,750	0,500	-0,900

Figure 4.5: Table based on causal activities' matrix

4.3.2 Discovery of process models

For discovering the process models, we will use **Heuristic Miner algorithm** (described on Section 3.2).

To achieve good results applying this algorithm, we first need to do an intermediate operation: Log inspector in ProM has an option where we can find the patterns we want (setting some parameters), and then creates the pattern abstractions. We will use this tool to convert all the relations *activity start - activity end* into an abstraction, so we will improve the fitness value when we will apply the *Heuristic Miner* plugin.

On Figure 4.6 it can be seen how parameters like conservedness and alphabet size have been set on 50 and 1 respectively, which implies to select just the patterns composed by more than an event, and with a conservedness value higher than 50 (this means a good pattern). As we can see, only 14 patterns have been found, inasmuch as only the start-complete patterns have those features. After finding them, we will select all the abstractions found and we will export the new log, which we are going to mine applying the Heuristic algorithm.

Once we have the Selected Abstraction log, and before applying the *Mine for a Heuristics Net using Heuristic Miner* plugin, we will add a starting and end point using another plugin so called *Add Artificial Events*. This step is needed to achieve a higher fitness value, due to the life cycle of the events we are working with, always falls inside its corresponding case. Thereby, each process starts at 4:00 AM and ends at 4:00AM.

Then we will apply the heuristic mining basing our threshold choice on the results

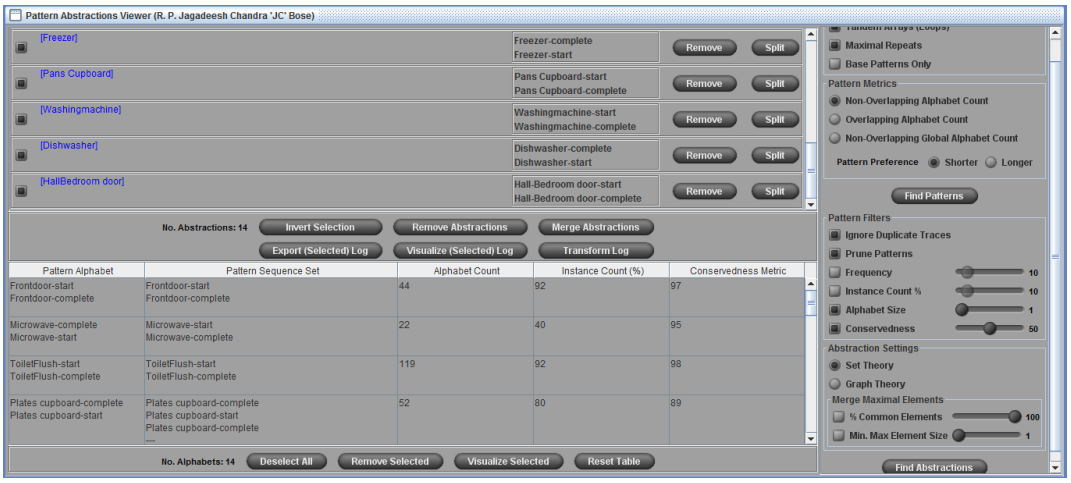


Figure 4.6: Pattern abstractions tool

explained on [3] for a log with low frequent traces and some noise. Thus, we achieved a fitness value of 0.5929 with a dependency threshold of 0.5, and relative-to-best threshold of 0.04.

Seeing Figure 4.7 we can consider a picture of the general inhabitant's behaviour and distinguish in a clear visual manner which their most usual patterns are. To understand the diagram we should remember the meaning of the numbers within the boxes, which represent the frequency of this activity, and also the numbers attached to the paths, which represent the frequency of arriving to one activity from another. Should be noticed also that values between start and complete boxes of a specific activity do not change, neither the path's one. This is because activities duration used to be short (activities are open a door, open the fridge, flush...etc), so in the most of the cases when an activity starts, the next event in the log is the activity end.

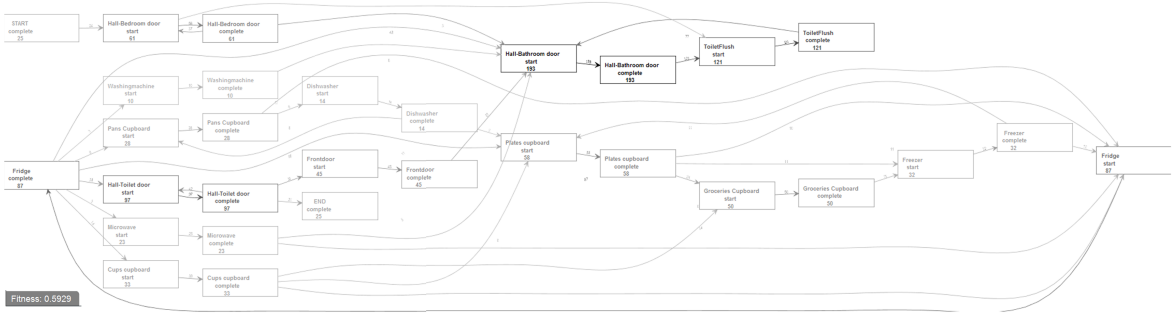


Figure 4.7: Heuristic Mine Output (find a bigger version of the figure on appendix **??**)

4.3.3 Analysis of activities duration and schedules

The log inspector in ProM provides a very powerful tool called Dotted chart which will allow us to discover everything about the relation between activities, traces and timestamps.

Starting from the original XES file, we will apply a *Simple Heuristic Filter* to delete the *complete* value from the life cycle, so we can see just one event for each activity instead of two (start and complete).

Dotted Chart tool, as we just said, has an output that is the graphic between two parameters that can be customized. For this analysis, we will choose three different relations.

1. **Relation between events and days of the week**

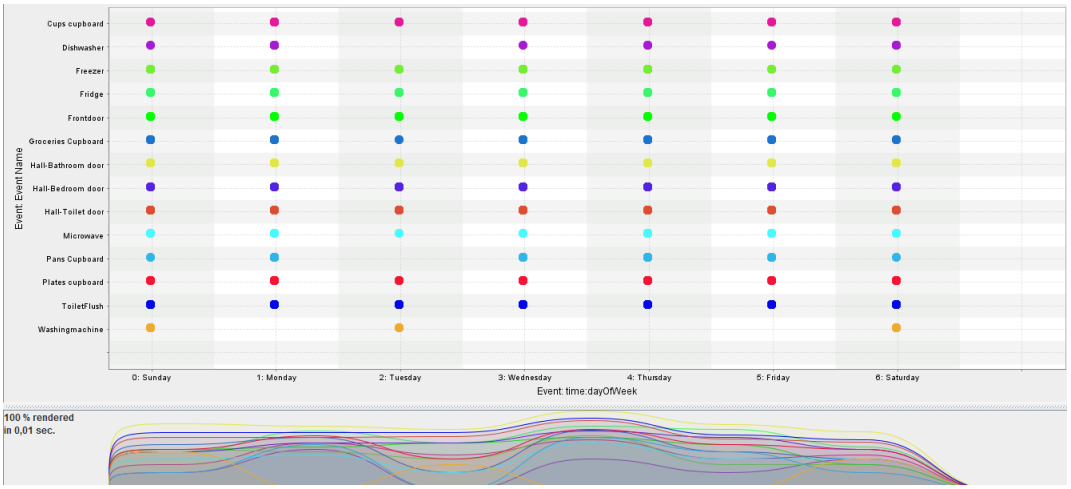


Figure 4.8: Graphic showing the relation between events and day of the week

Figure 4.8 shows basically how every activity is carried out by the subject every day except for the washing machine, which is usually used during the weekend and in the middle of the week. Note that something particular happens on Tuesday, when we can observe an irregularity on two events: pans cupboard and dishwasher.

On Figure 4.9 we can see a more detailed picture of the inhabitant routine each day of the week. For example, we can note during the week he does not usually have launch at home, whilst during the weekend, Saturday evening should be outside, and Sunday does not usually get up early.

We can observe a normal routine for a 26-years-old man. However, on Tuesday there is an unusual behaviour again, because if we follow the colours code of Figure 4.8, we see he uses to go bed at 3:00 AM.

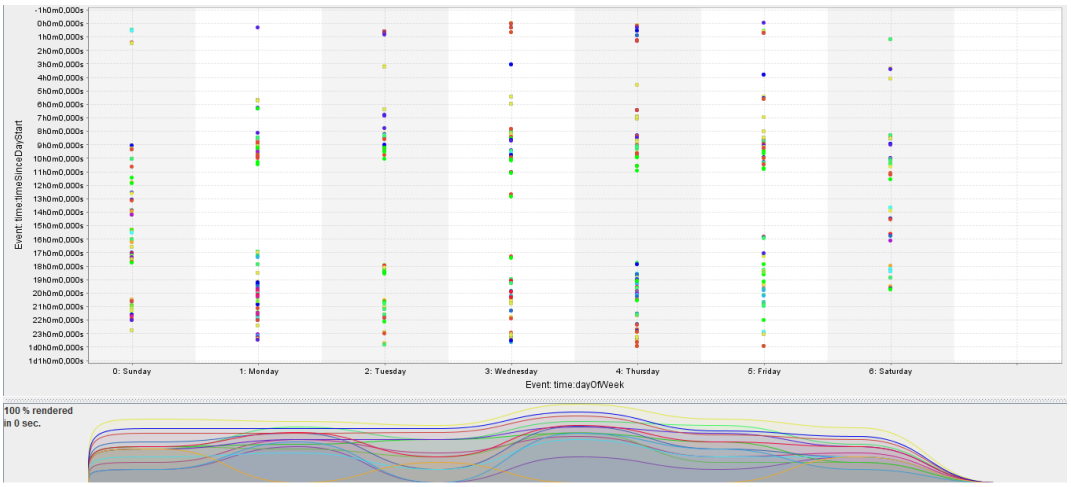


Figure 4.9: Graphic showing the relation between hours in the day and days of the week

It is even more detailed picture that we find on Figure 4.10. There we can see a mix of the previous two graphs (note that week starts on Sunday). So we can observe how frequently are the different activities at each hour on a particular week day. Thus, we confirm the unusual activity on Tuesday night (compared with the other week days).

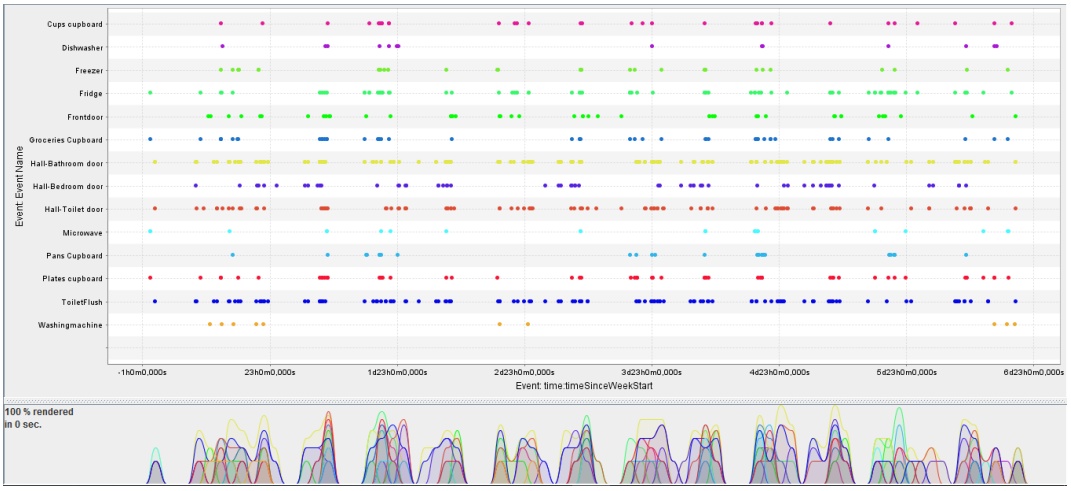


Figure 4.10: Graphic showing the relation between events and days of the week (and hours in the day)

2. **Relation between events and hours in the day**

   Figure 4.11 relates each event with the time in the day, so we can determine the moments of most activity for each event. For instance, events from bathroom and toilet does not sleep.

   An obvious fact, that maybe is more clearly seeing the graph below the image, that

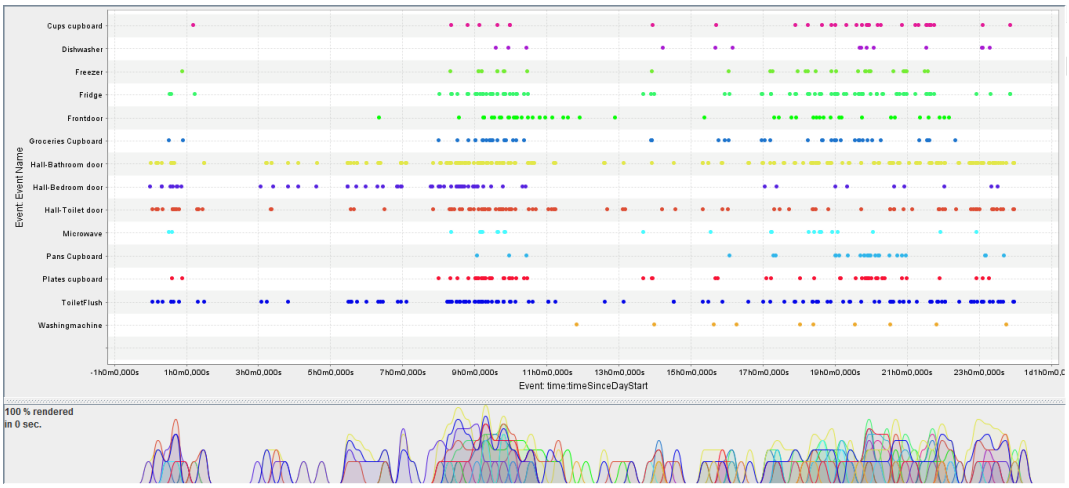


Figure 4.11: Graphic showing the relation between events and hours in the day

the period between 8:30 and 11:00 in the morning is the most active period of the house. Then, the normal activity level uses to come back from 17:00h. These facts become more consistently seeing the activity per hour of all the traces on Figure 4.12.

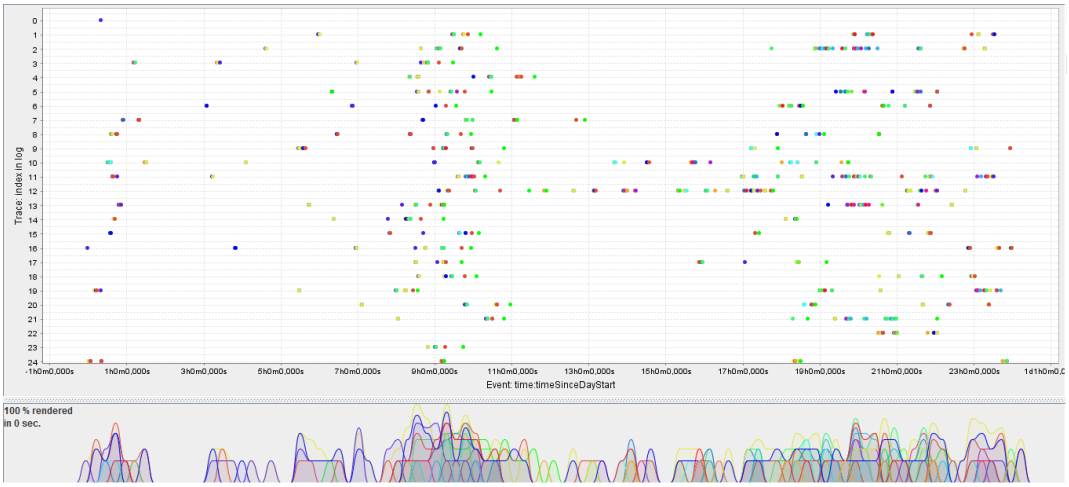


Figure 4.12: Graphic showing the relation between traces and hours in the day

3. **Relation between events and days of the month**

A much more eagle view is given by Figure 4.13 where can be observed two days with no activity: March 1st and 9th. Considering that those days were Friday and Saturday, respectively, we can assume he went out for those two weekends, which means the possible patterns found on Saturday or Sunday have less reliability than the ones found during the week, due to these days we have a lower amount of events to mine.

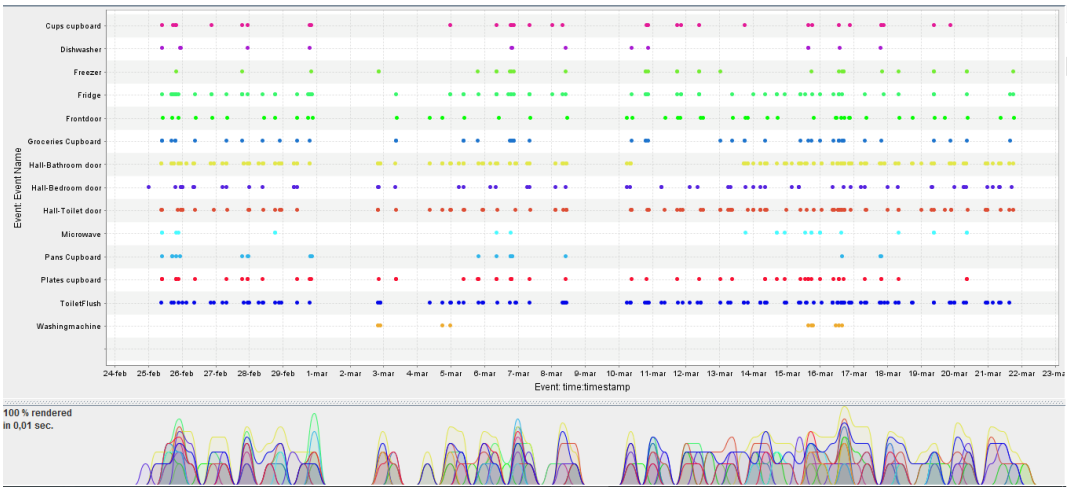


Figure 4.13: Graphic showing the relation between events and days of the month

Another unusual behaviour occurs between the afternoon of March 10th and the evening of March 13th. Those days the activity in the bathroom falls to zero, maybe due to a fault on the shower.

4.4 Applications of the results for Ambient Intelligence

After using process mining to get results from the senseData 4.2, we would like to apply this analysis into two different applications.

4.4.1 Recipes Recommendation for Task Automation

Home Automation is a growing field which is based on defining rules that are connected to different actions in order to optimize resources, time or even expenses in your house. IFTTT is a website where we can define in an extremely easy way those rules. The schema is very simple: **IF event THEN action**. There we can choose between a lot of events and actions (www.ifttt.com). Note that in the SmartHome beheld by the use case we are working on there are not installed smart devices more than some change-state sensors, so this point will remain as a possible application assuming we are in a scenery with everything connected.

To define recipes (see Figure 4.14), we will base them on results extracted from Figure 4.7. There, being A and B two different activities, we have to look at activity patterns from start to complete life cycle transition. That's it: A complete + B start. We will focus on those activities which have a unique path to another. Then, we have to check two values:

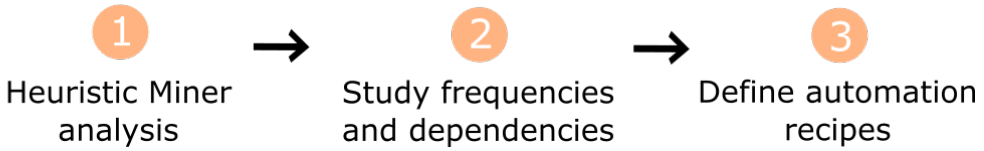


Figure 4.14: Steps followed to define recipes

the difference between frequency of A and the frequency of the path, from where we can calculate the probability for this pattern occurrence; and the dependency value between those activities, which we can find either in the table on Figure 4.5. Finally, we will establish a threshold for those parameters to define recipes.

For instance, looking at the path from *Frontdoor complete* to *Hall-Bathroom start*, we can calculate that this relation will occur with a probability of 60% (27/45). Looking at the discovery matrix (Figure 4.5), we can confirm that the reliability of going from one to another is 0.93, in fact the maximum value starting from the activity related to the front door. Therefore, we can define a rule as follows: **IF** *front door is opened* **THEN** *will send you an alert asking you if I should turn on bathroom's light.*

To illustrate this method, we have been defined some recipes extracted for the analysis:

- **IF** *Frontdoor* **THEN** *Smartphone alert recommending actions in the bathroom**
- **IF** *Hall-Bedroom door* **THEN** *Smartphone alert recommending actions in the bathroom*
- **IF** *Hall-Bathroom door* **THEN** *Smartphone alert recommending actions in the toilet*

*Actions in the bathroom can be, i.e, to turn on the shower, to heat up the water or to turn on the light.

As we can see, process mining potential for task automation could be enormous installing sensors in strategic parts of a house.

4.4.2 Agent-based Social Simulation

Agent-Based Social Simulation (ABSS) [14] is scientific discipline concerned with simulation of social phenomena, using computer-based multiagent models. In these simulations, persons or group of persons are represented by agents. A lot of social processes were observed in their model including death, disease, trade, wealth, sex and reproduction, culture,

conflict and war, and externalities such as pollution so, to apply process mining results to this field can achieve great results.

For instance, to analyse results on Section 4.3.3 provides us a huge amount of information about activities schedules: when are activities being carried out, at what time during the day, which days...etc. All this information, joined with probabilities and dependencies extracted before using Heuristic Miner, becomes very useful in terms of behaviour prediction for virtual subjects simulation to design, for example, an evacuation plan in a big building.

An application also related with ABSS, left for future works due to it exceeds the scope of this project, would be extract from ProM the average duration of each activity and use some tools as EasyFitXL [1] to discover the distribution it follows, in order to be able of predicting the duration of each activity for different virtual agents in a social simulation.

[1] http://www.mathwave.com/articles/fit-distributions-excel.html

CHAPTER 5

Conclusions and future lines

In this chapter we will describe the conclusions extracted from this final project, the achievements and thinkings about future work.

5.1 Conclusions

In this final project, we have been summarized what *Process Mining* is, explaining its main functionalities, and giving a brief introduction to common formats and algorithms. We have also analysed the different tools used for process mining, such as ProM and Disco, evaluating their weaknesses and their strengths and comparing the advantages each one can provide. Then, a general methodology has been developed in order to analyse Event Logs using ProM. And finally, a particular use case has been selected to be analysed using the elements previously described and we have demonstrated how results extracted from this analysis can be applied for Ambient Intelligence applications such as Task Automation or Agent-based social simulation.

This project has been developed due to applications for Ambient Intelligence related processes has not received enough attention in the current literature about process mining, leaving a little from the field which it has emerged from: Business Intelligence. At the same time, this projects aims to be a basis on which this new field, full of possibilities as we have seen throughout the project, can be explored.

During this project, the initially chosen use case has proved insufficient to exploit the whole possibilities that subject offers. The analysis of a more developed dataset, perhaps with a longer log registration period, or even several dataset which could be put together, would be allowed a much more complete demonstration. However, we have extracted the juice enough to cook an aperitif that aims to whet the appetite and to precede a much bigger feast.

5.2 Achieved goals

Analysis of Process Mining tools This goal has been achieved successfully. We have been able to discover the benefits each technology can provide, the main techniques each tool is using, advantages and disadvantages of each one, and then do a brief comparison between both. This helped us to choose which technology fits better with our goals. It can be seen on Section 2.3.

Development of a methodology to mine processes using ProM It was essential to

structure and understand the steps taken for the later analysis. This has been possible thanks to the literature found about ProM and their plugins. This is detailed in Chapter 3.

PM analysis of a particular dataset and its use for AmI applications This has been the main goal achieved. It has given us the chance to understand better the fundamentals of *Process Mining*, to test the methodology proposed in Chapter 3 with great results, and to demonstrate how useful can be process mining applied to Ambient Intelligence field. More information can be consulted in Chapter 4.

5.3 Future work

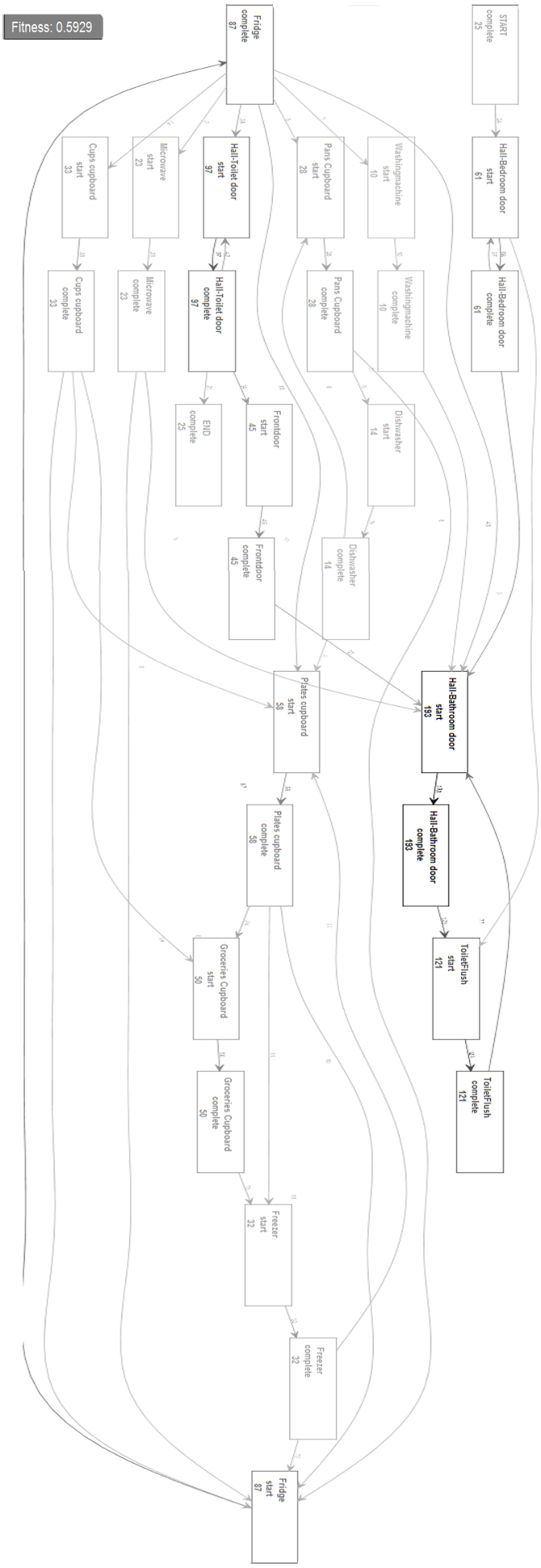
In the following points some fields of study or improvement are presented to continue the development.

- Monitoring daily life activities based on the logs registered by Estimotes [1].
- Discovering times distribution of each activity in order to use it into agent-based social simulation.
- In the Healthcare field, detecting unusual behaviours of patients in order to prevent possible diseases or bad habits.
- Helping athletes to monitor their activities in order to achieve optimal results or to help trainers to recommend personalised workouts.
- Health Smart Home, to follow dependent people at home in order to avoid its hospitalisation [15].

[1] www.estimote.com

APPENDIX A

Appendix


Fitness: 0.5929
Fridge complete 87
START complete 25
Cups cupboard start 33
Microwave start 23
Hall-Toilet door start 97
Pans Cupboard start 28
Washingmachine start 10
Hall-Bedroom door start 61
Cups cupboard complete 33
Microwave complete 23
Hall-Toilet door complete 97
Pans Cupboard complete 28
Washingmachine complete 10
Hall-Bedroom door complete 61
END complete 25
Frontdoor start 45
Dishwasher start 14
Frontdoor complete 45
Dishwasher complete 14
Plates cupboard start 58
Hall-Bathroom door start 193
Plates cupboard complete 58
Hall-Bathroom door complete 193
Groceries Cupboard start 50
ToiletFlush start 121
Groceries Cupboard complete 50
ToiletFlush complete 121
Freezer start 32
Freezer complete 32
Fridge start 87

Bibliography

[1] W. Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.

[2] IEEE Task Force on Process Mining, "Process Mining Manifesto," in *BPM Workshops*, vol. 99 of *Lecture Notes in Business Information Processing*, Springer-Verlag, Berlin, 2011.

[3] A. Weijters, W. Aalst, and A. Medeiros, "Process Mining with the Heuristics Miner-algorithm." BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.

[4] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining - adaptive process simplification based on multi-perspective metrics.," in *BPM* (G. Alonso, P. Dadam, and M. Rosemann, eds.), vol. 4714 of *Lecture Notes in Computer Science*, pp. 328–343, Springer, 2007.

[5] W. Mettrey, "A comparative evaluation of expert system tools.," *IEEE Computer*, vol. 24, no. 2, pp. 19–31, 1991.

[6] R. López-Cózar and Z. Callejas, "Multimodal dialogue for ambient intelligence and smart environments," in *Handbook of ambient intelligence and smart environments*, pp. 559–579, Springer, 2010.

[7] C. Günther, "XES Standard Definition." www.xes-standard.org, 2009.

[8] C. Günther and W. Aalst, "A Generic Import Framework for Process Event Logs," in *Business Process Management Workshops, Workshop on Business Process Intelligence (BPI 2006)* (J. Eder and S. Dustdar, eds.), vol. 4103 of *Lecture Notes in Computer Science*, pp. 81–92, Springer-Verlag, Berlin, 2006.

[9] O. Source, "Prom - framework for process mining." `http://sourceforge.net/projects/prom`, last accessed on May 31st, 2006.

[10] W. Aalst and M. Song, "Mining Social Networks: Uncovering Interaction Patterns in Business Processes," in *International Conference on Business Process Management (BPM 2004)* (J. Desel, B. Pernici, and M. Weske, eds.), vol. 3080 of *Lecture Notes in Computer Science*, pp. 244–260, Springer-Verlag, Berlin, 2004.

[11] C. W. Günther and A. Rozinat, "Disco: Discover your processes.," in *BPM (Demos)* (N. Lohmann and S. Moser, eds.), vol. 940 of *CEUR Workshop Proceedings*, pp. 40–44, CEUR-WS.org, 2012.

[12] L. F. N. da Silva, "Process Mining: Application to a case study," Faculdade de Economia, Universidade do Porto, 2014.

[13] T. van Kasteren, A. K. Noulas, G. Englebienne, and B. J. A. Kröse, "Accurate activity recognition in a home setting," in *UbiComp* (H. Y. Youn and W.-D. Cho, eds.), vol. 344 of *ACM International Conference Proceeding Series*, pp. 1–9, ACM, 2008.

[14] M. Fasli, "Formal systems and agent-based social simulation equals null?," *J. Artificial Societies and Social Simulation*, vol. 7, no. 4, 2004.

[15] G. V. Jacques Demongeot, "Multi-sensors acquisition, data fusion, knowledge mining and alarm triggering in health smart homes for elderly people," Laboratoire TIMC–IMAG, UMR CNRS 5525, faculté de médecine, université Joseph-Fourier de Grenoble, 38700 La Tronche, France, 2002.