1. Introduction
The speed at which data grows in IT systems [1] makes it crucial to rely on automation in order to enable enterprises and institutions to manage their processes. Automated techniques open the door for dealing with large amounts of data, a mission unthinkable for a human’s capabilities. In this paper we discuss one of these techniques: the discovery of process models. We now illustrate the main task behind process discovery by means of a (hopefully) funny example.

2. A Funny Example: The Visit of an Alien
Imagine that an alien visits you (see Figure 1) and, by some means, it wants to communicate the plan it has regarding its discovery by means of a (hopefully) funny example.

Although not knowing the meaning of each individual letter in the message above, one may detect that there are some patterns, e.g., a repetition for the sequence IA C D M E (first and last six letters in the sequence). So the question is: how can we represent the behavior of the aliens without knowing exactly the meaning of each single piece of information?

Process discovery may be a good solution for this situation: a process discovery algorithm will try to produce a (formal) model of the behavior underlying a set of sequences. For instance, the following formal model in the Business Process Modeling Notation (BPMN) [2] shown in Figure 3 represents very accurately the behavior expressed in the alien’s sequences. For those not familiar with the BPMN notation, the model above describes the following process: after I occurs, then (‘x’ gateway) either branch B followed by X occurs, or branch A followed by C and D in parallel (‘+’ gateway), and then M occurs. Both branches activate E which in turn reactivates I. Clearly, even without knowing anything about the actions taken from the alien, the global structuring of these activities becomes very apparent from a simple inspection of the BPMN model.

Now imagine that at some point the meaning of each letter is decrypted: evaluate the amount of energy in the Earth (I), high energy (B), invade the Earth (X), low energy (A), gather some human samples (C), learn the human reproduction system (D), teach humans to increase their energy resources (M), communicate the situation to the aliens in the closest UFO (E). In the presence of this new information, the value of the model obtained is significantly incremented (although maybe one may not be relaxed after realizing the global situation that the model brings into light).

I A C D M E I B X E I A D C M E I B X E I A C D M E

Figure 1. Our Imaginary Alien.

Figure 2. A Message Sent by the Alien.

3. Anatomy of a Simple Process Discovery Algorithm
The previous example illustrates one of the main tasks of a process discovery algorithm: given a set of traces (called log) corresponding to a particular behavior under study, derive a formal model which represents faithfully the process producing these traces. In its simplest form, process discovery algorithms focus on the control-flow perspective of the process, i.e., the ordering activities are performed in order to carry out the process tasks. The previous example has considered this perspective.

A log must contain enough information to extract the sequencing of the activities that are monitored. Typically, a trace identifier, an activity name and a time stamp are required to enable the corresponding sequencing (by the time stamp) for the activities belonging to a given trace (determined by the trace identifier). Other information may be required if the discovery engine must take into account additional information, like resources (what quantity was purchased?), activity originator (who performed that activity?), activity duration (how long does activity X last?), among others. An example of a discovery algorithm that takes into account other dimension is the social network miner [3], that derives the network of collaborators that carry out a given process.

Keywords: Formal Methods, Process Discovery, Software Engineering.

Josep Carmona received his MS and PhD degrees in Computer Science from the Technical University of Catalonia, in 1999 and 2004, respectively. He is an associate professor in the Software Department of the same university. His research interests include formal methods, concurrent systems, and process and data mining. He has co-authored more than 50 research papers in conferences and journals.
The core of a process discovery algorithm is the ability to extract the necessary information required to learn a model that will represent the process. Process discovery is often an unsupervised learning task, since the algorithm is usually exposed only to positive examples, i.e., successful executions of the process under study; in the example of the introduction, we were only exposed to what the alien plans to do, but we do not know what the alien does not plan to do. This complicates the learning task, since process discovery algorithms are expected to produce models that are both precise (the model produced should not deviate much from the behavior seen) and general (the model should generalize the patterns observed in the log) [4]. Obviously, the presence of negative examples would help the discovery algorithm into improving these two quality metrics, but negative information is often not available on IT logs.

How to learn a process model from a set of traces? Various algorithms exist nowadays for various models (see Section 4). However, let us use the alien’s example to reason on the discovery of the BPMN model above. If we focus on the first letter of the sequence (I), it is sometimes followed by A and sometimes by B, and always (except for the first occurrence) preceded by E. These observations can be expressed graphically as shown in Figure 4.

In BPMN notation, the or-exclusive relation between the occurrences of either A or B after I is modeled by using the ‘x’ gateway. The precedence between E and I is modeled by an edge connecting both letters in the model. Symmetrically, E is preceded either by M or by X. Also, following A both C and D occur in any order. The well-known alpha algorithm [5] can find most of these pairwise ordering relations in the log, and one may use them to craft the BPMN model as Table 1 illustrates.

Table 1 can be read as follows: if in the log A precedes B always but B is unique (there is no other letter preceded by A), then a directed arc between A and B is created. If in contrast there is always more than one letter preceded by A, then an ‘+’ gateway is inserted between A and the letters preceded by A. The sometimes relation can be read similarly.

Hence one can scan the log to extract these relations (worse-case quadratic in the size of the log) and use the table to create the BPMN model. However, this is a very restrictive way of discovery since other relations available in the BPMN notation can also be hidden in the log, like the inclusive-or relation, but the algorithm does not consider them. Process discovery algorithms are always in a trade-off between the complexity of the algorithm and the modeling capacity: the algorithm proposed in this section could be extended to consider also inclusive-or gateways, but that may significantly complicate the algorithm. Below we address informally these and other issues.

4. Algorithms and Models
There are several models that can be obtained through different process discovery algorithms: Petri nets, Event-driven Process Chains, BPMN, C-Nets, Heuristic Nets, Business Process Maps, among others. Remarkably, most of these models are supported by replay semantics that allow one to simulate the model in order to certify its adequacy in representing the log.

To describe each one of these models is out of the scope of this article, but I can briefly comment on Petri nets, which is a model often produced by discovery algorithms, due to its formal semantics and ability to represent concurrency. For the model of our running example, the corresponding Petri net that would be discovered by most of the Petri net discovery algorithms will be as shown in Figure 5.

Those readers familiar with Petri nets will find a perfect match between the underlying behavior of the Petri net and the alien’s trace. Notice that while in the BPMN model, apart from the units of information (in this case letters of the alphabet), there are other model components (gateways) whose semantics define the way the model represents the log traces.

The same happens with the Petri net above, where the circles correspond to the global behavior of the model, which is distributed among the net (only some circles are marked). While the discovery algorithm for BPMN needs to find both the connections and gateways, the analogous algorithm for Petri nets must compute the circles and connections.

Several techniques exist nowadays to accomplish the discovery of Petri nets, ranging from the log-ordering relations extracted by the alpha algorithm, down to very complex graph-based structures that are computed on top of an automaton representing the log traces.

What process discovery algorithm/modeling notation to choose? This is in fact a very good question that can only be answered partially: there is no one model that is better than the rest, but instead models that are better than others only for a particular type of behaviors. Actually, deciding the best
modeling notation for a log is a hard problem for which research must provide techniques in the next decade (a problem called representational bias selection). From a pragmatic point of view, one must select those process modeling notations one is familiar with, and expect the discovery algorithms for that notation to be good enough for the user needs.

As said before, other perspectives different from the control-flow view may be considered by process discovery algorithms: time, resources, organizational, etc.

The reference book [6] may be consulted in order to dig into these other process discovery algorithms.

6. Tools
Process discovery is a rather new discipline, if compared with related areas such as data mining or machine learning. In spite of this, one can find process mining tools both in academia (mostly) but also in industry.

The following classification is by no means exhaustive, but instead reports some of the prominent tools one can use to experience with process discovery tools:

- ACADEMIA: the ProM Framework, from Technical University of Eindhoven (TU/e) is the reference tool nowadays. It is the result of a great academic collaboration among several universities in the world to gather algorithmic support for process mining (i.e., not only process discovery). Additionally, different groups have developed several academic stand-alone tools that incorporate modern process discovery algorithms.

- INDUSTRY: some important companies have invested an effort into building process discovery tools, e.g., Fujitsu (APD), but also medium-sized or start-ups that are more focused on process mining practices, e.g.,

<table>
<thead>
<tr>
<th>A precedes B</th>
<th>Always</th>
<th>Sometimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>B Unique:</td>
<td>![A precedes B diagram]</td>
<td>![B Unique diagram]</td>
</tr>
<tr>
<td>General:</td>
<td>![General A precedes B diagram]</td>
<td>![General B Unique diagram]</td>
</tr>
</tbody>
</table>

Table 1. BPMN Model Built from Patterns in the Alien’s Messages.
Pallas Athena (ReflectOne), Fluxicon (Disco), Perspective Software (BPMMOne, Futura Reflect), Software AG (ARIS Process Performance Manager), among others.

6. Challenges
The task of process discovery may be aggravated if some of the aspects below are present:

- **Log incompleteness:** the log often contains only a fraction of the total behavior representing the process. Therefore, the process discovery algorithm is required to guess part of the behavior that is not present in the log, which may be in general a difficult task.

- **Noise:** logged behavior may sometimes represent infrequent exceptions that are not meant to be part of the process. Hence, process discovery algorithms may be hampered when noise is present, e.g., in control-flow discovery some relations between the activities may become contradictory. To separate noise from the valid information in a log is a current research direction.

- **Complexity:** due to the magnitude of current IT logs, it is often difficult to use complex algorithms that may either require loading the log into memory in order to derive the process model, or apply techniques whose complexity are not linear on the size of the log. In those cases, high level strategies (e.g., divide-and-conquer) are the only possibility to derive a process model.

- **Visualization:** even if the process discovery algorithm does its job and can derive a process model, it may be hard for a human to understand it if it has more than a hundred elements (nodes, arcs). In those cases, a hierarchical description, similar to the Google Maps application were one can zoom in or out of a model’s part, will enable the understanding of a complex process model.

**Acknowledgements**
I would like to thank David Antón for creating the alien’s drawing used in this paper.

References


