Finding Structure in Unstructured Processes: The Case for Process Mining

(invited paper)

W.M.P. van der Aalst and C.W. Günther
Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{w.m.p.v.d.aalst,c.w.gunther}@tue.nl

Abstract

Today there are many process mining techniques that allow for the automatic construction of process models based on event logs. Unlike synthesis techniques (e.g., based on regions), process mining aims at the discovery of models (e.g., Petri nets) from incomplete information (i.e., only example behavior is given). The more mature process mining techniques perform well on structured processes. However, most of the existing techniques fail miserably when confronted with unstructured processes. This paper attempts to "bring structure to the unstructured" by using an integrated combination of abstraction and clustering techniques. The ultimate goal is to present process models that are understandable by analysts and that lead to improved system/process redesigns.

1. Introduction

The International Conference on Application of Concurrency to System Design (ACSD) serves as a "forum for disseminating theoretical results with application potential and advanced methods and tools for the design of complex concurrent systems". Hence most contributions are focusing on methods to better design a wide range of concurrent systems. This paper takes a different perspective. The goal is *not to design a new system but to analyze a system that is being used in a particular context*. The main idea is that it is only possible to come to an improved system design if there is a good understanding of the existing design and its actual use.

Process mining has emerged as a way to analyze systems and their actual use based on the event logs they produce [3, 4, 5, 7, 10, 13, 17, 20, 22]. Note that, unlike classical data mining, the focus of process mining is on concurrent processes and not on static or mainly sequential structures. Also note that commercial "Business Intelligence" (BI) tools are not doing any process mining. They typically look at aggregate data seen from an external perspective (frequencies, averages, utilization, service levels, etc.). Unlike BI tools, process mining looks "inside the process" (What are the causal dependencies?, Where is the bottleneck?, etc.) and at a very refined level. In the context of a hospital, BI tools focus on performance indicators such as the number of knee operations, the length of waiting lists, and the success rate of surgery. Process mining is more concerned with the paths followed by individual patients and whether certain procedures are followed or not.

Process mining always starts with *event logs*. Events logs may originate from all kinds of systems ranging from enterprise information systems to embedded systems. Process mining is a very broad area both in terms of (1) applications (from hospitals and banks to embedded systems in cars, copiers, and sensor networks) and (2) techniques. This paper will focus on the *discovery of processes* as a particular process mining technique. An example is the α-algorithm [4] which automatically constructs a Petri net based on a set of observed system traces. Process discovery is related to the *synthesis problem* addressed by techniques such as the "Theory of Regions" [6, 8, 16]. However, it is important to realize that *process mining is also very different since synthesis techniques typically assume a complete description of the behavior*. This is not the case for process mining. In reality one only observes a fraction of all possible traces. Therefore, a model that is only able to reproduce the log is useless!

Process mining techniques work well on structured processes with little exceptional behavior and strong causal dependencies between the steps in the process. However, the most interesting logs are much more difficult to mine. Note that unstructured processes are typically difficult to manage while having a big potential for improvement. Therefore our aim is to *"find structure in unstructured processes"*. For this purpose we use the *"map metaphor"*. Just like processes are visualized using graphs (e.g., Petri nets), (road) maps visualize some spatial reality.

Maps are highly evolved human artifacts for planning and way-finding which are heavily using *abstraction* (leaving out details, e.g., minor towns and roads) and *clustering* (aggregating entities, e.g. a city represented as "blob" hiding details inside). Maps abstract and cluster away unneeded details to allow humans to recognize and correctly interpret complex information. Our goal is to apply such ideas to the complex "spaghetti-like" process models discovered using process mining. This paper presents some of our initial ideas. These ideas have been implemented in *ProM* [14]. ProM is an open-source plug-able framework that provides a wide range of process mining techniques.

The remainder of this invited paper is structured as follows. First we provide an overview of process mining (Section 2) and discuss some typical process discovery approaches (Section 3). In Section 3 we also stress the differences between process mining and synthesis. Section 4 shows the problems encountered when mining unstructured processes. Motivated by these problems, we propose a solution approach in Section 5. Section 6 concludes the paper.

2. Process Mining: A Short Overview

Process mining is applicable to a wide range of systems. These systems may be pure information systems (e.g., ERP systems) or systems where the hardware plays a more prominent role (e.g., embedded systems). The only requirement is that the system produces *event logs*, thus recording (parts of) the actual behavior.

An interesting class of information systems that produce event logs are the so-called *Process-Aware Information Systems* (PAISs) [15]. Examples are classical workflow management systems (e.g. Staffware), ERP systems (e.g. SAP), case handling systems (e.g. FLOWer), PDM systems (e.g. Windchill), CRM systems (e.g. Microsoft Dynamics CRM), middleware (e.g., IBM's WebSphere), hospital information systems (e.g., Chipsoft), etc. These systems provide very detailed information about the activities that have been executed.

However, not only PAISs are recording events. Also *embedded systems* increasingly log events. An embedded system is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Examples are medical systems (e.g., X-ray machines), mobile phones, car entertainment systems, production systems (e.g., wafer steppers), copiers, sensor networks, etc. Software plays an increasingly important role in such systems and, already today, many of these systems record events. An example is the "CUSTOMer-CARE Remote Services Network" of Philips Medical Systems (PMS). This is a worldwide internet-based private network that links PMS equipment to remote service centers. An event that occurs within an X-ray machine (e.g., moving the table, setting the deflector, etc.) is recorded and analyzed. Another example is the event logging infrastructure developed by ASML. ASML manufactures chip-making equipment such as wafer scanners. Its products are also connected to the internet. This connection is used to distribute the events logged. Already during the testing phase of a wafer scanner in the factory, thousands of events are recorded via an internet connection. This can be used for improving the service, improving the products, and for improving the test processes. The logging capabilities of the machines of PMS and ASML illustrate the way in which embedded systems produce event logs.

The goal of process mining is to extract information (e.g., process models) from these logs, i.e., process mining describes a family of *a-posteriori* analysis techniques exploiting the information recorded in the event logs. Typically, these approaches assume that it is possible to sequentially record events such that each event refers to an activity (i.e., a well-defined step in the process) and is related to a particular case (i.e., a process instance). Furthermore, some mining techniques use additional information such as the performer or originator of the event (i.e., the person / resource executing or initiating the activity), the timestamp of the event, or data elements recorded with the event (e.g., the size of an order).

Process mining addresses the problem that most "process/system owners" have limited information about what is actually happening. In practice, there is often a significant gap between what is prescribed or supposed to happen, and what *actually* happens. Only a concise assessment of reality, which process mining strives to deliver, can help in verifying process models, and ultimately be used in system or process redesign efforts.

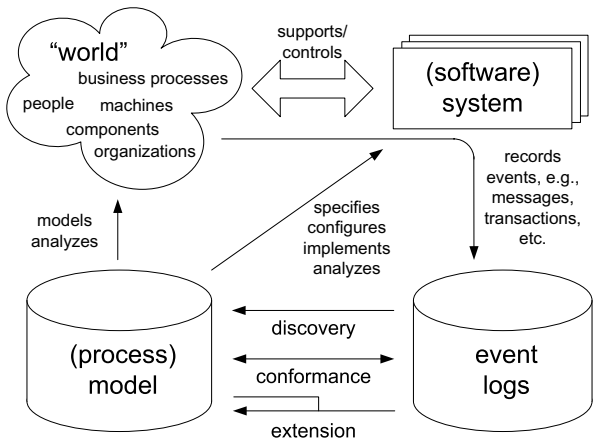

Figure 1. Three types of process mining: (1) Discovery, (2) Conformance, and (3) Extension.

The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs. We consider three basic types of process mining (Figure 1):

- **Discovery**: There is no a-priori model, i.e., based on an event log some model is constructed. For example, using the α-algorithm [4] a process model can be discovered based on low-level events.
- **Conformance**: There is an a-priori model. This model is used to check if reality conforms to the model. For example, there may be a process model indicating that purchase orders of more than one million Euro require two checks. Another example is the checking of the four-eyes principle. Conformance checking may be used to detect deviations, to locate and explain these deviations, and to measure the severity of these deviations.
- **Extension**: There is an a-priori model. This model is extended with a new aspect or perspective, i.e., the goal is not to check conformance but to enrich the model with the data in the event log. An example is the extension of a process model with performance data, i.e., some a-priori process model is used on which bottlenecks are projected.

Traditionally, process mining has been focusing on *discovery*, i.e., deriving information about the original process model, the organizational context, and execution properties from enactment logs. An example of a technique addressing the control flow perspective is the α-algorithm, which constructs a Petri net model [11, 19] describing the behavior observed in the event log. However, process mining is not limited to process models (i.e., control flow) and recent process mining techniques are more and more focusing on other perspectives, e.g., the organizational perspective or the data perspective. For example, there are approaches to extract social networks from event logs and analyze them using social network analysis [1]. This allows organizations to monitor how people, groups, or software/system components are working together.

Conformance checking compares an a-priori model with the observed behavior as recorded in the log. In [20] it is shown how a process model (e.g., a Petri net) can be evaluated in the context of a log using metrics such as "fitness" (Is the observed behavior possible according to the model?) and "appropriateness" (Is the model "typical" for the observed behavior?). However, it is also possible to check conformance based on organizational models, predefined business rules, temporal formulas, Quality of Service (QoS) definitions, etc.

There are different ways to *extend* a given process model with additional perspectives based on event logs, e.g., decision mining, performance analysis, and user profiling. Decision mining, also referred to as decision point analysis, aims at the detection of data dependencies that affect the routing of a case [21]. Starting from a process model, one can analyze how data attributes influence the choices made in the process based on past process executions. Classical data mining techniques such as decision trees can be leveraged for this purpose. Similarly, the process model can be extended with timing information (e.g., bottleneck analysis).

At this point in time there are mature tools such as the ProM framework [14], featuring an extensive set of analysis techniques which can be applied to real-life logs while supporting the whole spectrum depicted in Figure 1.

3. Process Discovery with ProM

In this paper, we focus on *control-flow discovery*, i.e., based on some event log we try to discover a process model (e.g., a Petri net) characterizing the behavior recorded in the log. MXML, the format used by ProM [14], allows for the logging of a wealth of information. However, for this paper we limit ourselves to the basic information present in any event log.

Let A be a set of *activity* names. For every *process instance* (often referred to as *case*), a sequence of activities is recorded. Such a sequence of activities is called a *trace*. Note that there may be different process instances that have the same trace. A^* is the set of all possible traces given a set of activities A. An *event log* is a set of process instances. Since only the corresponding traces are of interest here, an event log L is described by a bag (i.e., a multi-set) of traces. In other words: $L \in A^* \rightarrow \mathbb{N}$, i.e., for any possible $\sigma \in A^*$, $L(\sigma)$ denotes the number of process instances having a sequence of activities σ.

As an example consider the process of handling customer orders. An example of a trace would be *(register, ship, send_bill, payment, accounting, approved, close)*. This trace represents a sequence of 7 activities. To represent the log we use more convenient (i.e., shorter) names: *r=register*, *s=ship*, *sb=send_bill*, *p=payment*, *ac=accounting*, *ap=approved*, *c=close*. Moreover, there are additional activities such as *em=express_mail*, *rj=rejected*, and *rs=resolve*. Using this shorter notation we now consider a log L where each of the following traces occurs once (r, s, sb, p, ac, ap, c), $(r, sb, em, p, ac, ap, c)$, $(r, sb, p, em, ac, rj, rs, c)$, $(r, em, sb, p, ac, ap, c)$, $(r, sb, s, p, ac, rj, rs, c)$, (r, sb, p, s, ac, ap, c), and $(r, sb, p, em, ac, ap, c)$. It is easy to imagine that such traces could be extracted from the logs of some information system (e.g., a SAP R/3 system). It should be noted that all traces start with r (register order) and end with c (close order). Moreover, some of the other activities also appear

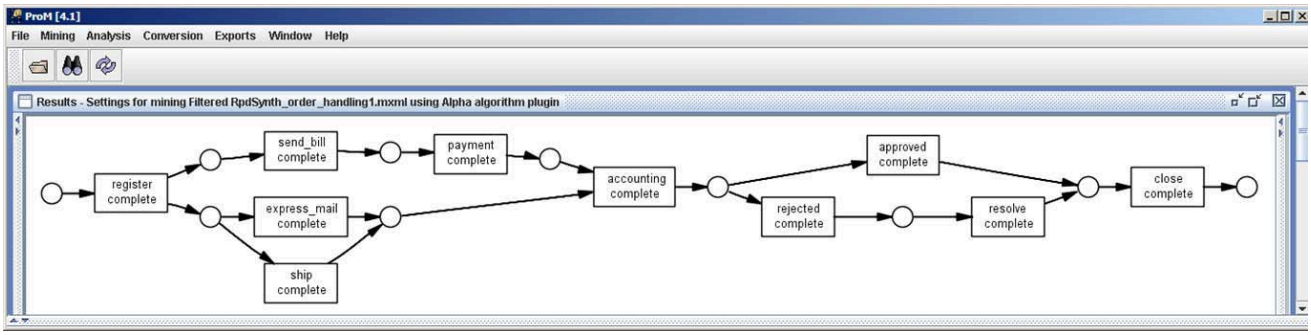

Figure 2. A Petri net discovered by ProM using the α-algorithm.

in all traces, e.g., sb (send bill), p (payment), and ac (accounting). In all traces either s (ship) or em (express mail) occurs. If ap (approved) occurs in this small set of traces, then rj (rejected) and rs (resolve) do not occur (and vice versa). Note that for the human eye it is difficult to make these conclusions and construct a process model that corresponds to the behavior recorded in event log L. Therefore, many process discovery algorithms have been proposed to automatically construct process models (e.g., a Petri net) [3, 4, 5, 7, 10, 13, 17, 22].

ProM implements about 20 different process discovery algorithms. Since a complete review of the different algorithms is outside the scope of this paper, we limit ourselves to showing some examples. Figure 2 shows a Petri net discovered using the α-algorithm [4], i.e., based on the event log L a model is constructed automatically. It is easy to see that the traces in the log can indeed be reproduced by this Petri net. Note that the α-algorithm "discovers" choices and concurrency. Although the example does not contain any loops, the α-algorithm can also discover iterations. The α-algorithm is rather sensitive to noise and exceptional behavior and has problems handling more advanced control-flow patterns. Figure 3 shows two alternative techniques that are more robust. The multi-phase miner always produces a model that can replay the log [13]. It uses Event-driven Process Chains (EPCs) as a default representation as shown on the left-hand-side of Figure 3. However, the EPCs can be converted in other formats such as various types of Petri nets, YAWL models, BPEL specifications, etc. The drawback of the technique used by the multi-phase miner is that it has a tendency to over-generalize, i.e., sometimes the model allows for too much behavior. The model shown on the right-hand-side of Figure 3 is produced by the heuristics miner [22]. The heuristics miner represents processes in a notation dedicated to process mining. However, its results can be converted to other notations. The heuristics miner specializes in dealing with noise and exceptional situations.

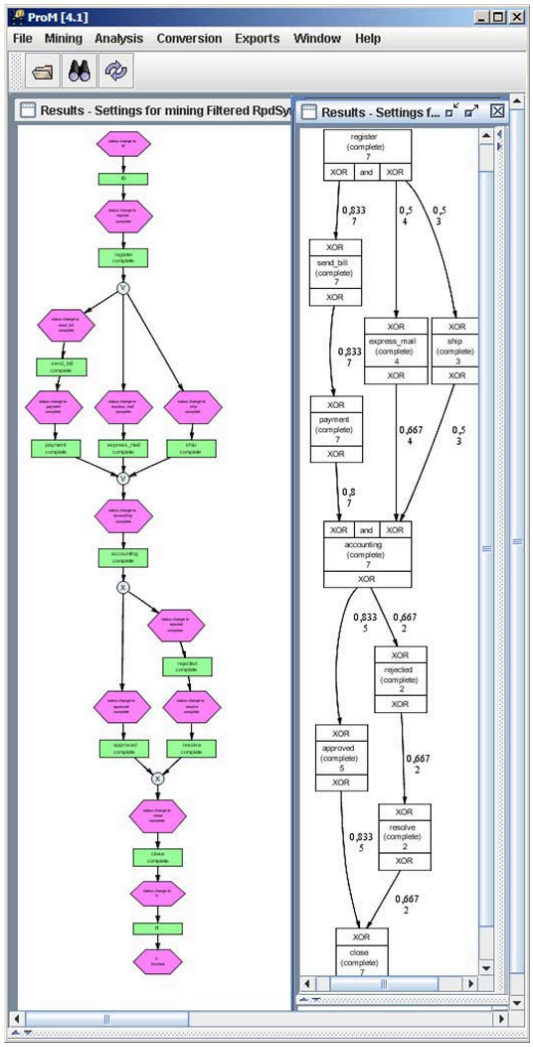

Figure 3. Two process models discovered using the multi-phase miner (left) and the heuristics miner (right).

It is not necessary to understand the three discovered

models shown in figures 2 and 3. However, it is important to note that there are various process mining algorithms that perform well on structured processes. ProM offers a wide variety of process discovery techniques. Using ProM the discovered models can be converted to the desired format (Petri nets, EPCs, etc.).

In the next section, we will show that process discovery is not easy for large unstructured processes. However, before doing so we would like to discuss the *link between process mining and synthesis*. Synthesis methods try to extract a *precise* model from a transition system or a set of behaviors (traces or partial orders). A well-known example is the "Theory of Regions" [16, 12, 8] which attempts to convert a transition system into a Petri net model. The first papers on the Theory of Regions only dealt with a special class of transition systems called elementary transition systems. The class of elementary transition systems is very restricted. In practice, most of the time, people deal with arbitrary transition systems that only by coincidence fall into the class of elementary transition systems. However, several researchers (e.g., Cortadella et al. [8]) have worked on generalizations. The approach presented in [8] uses labeled Petri nets, i.e., different transitions can refer to the same event. For this approach it has been shown that the reachability graph of the synthesized Petri net is *bisimilar* to the initial transition system. Other synthesis methods have been described in literature, e.g., in [18] some first steps are made towards solving the Petri net synthesis problem for non-sequential specifications (also known as scenarios, runs, labeled partial orders, or pomsets) and in [9] the limits of the "Theory of Regions" are explored. These synthesis approaches seem related to process mining. However, because they attempt to produce a model that exactly corresponds to the given behavior (e.g., in terms of a transition system, set of traces, or regular expression), they tend to "overfit".

To illustrate the "overfitting" of region-based approaches, consider the log L described before. Let us make an alternative log L' where the trace $(r, sb, s, p, ac, rj, rs, c)$ is replaced by (r, sb, s, p, ac, ap, c). ProM offers two mining plug-ins based on regions. The first plug-in incrementally builds regions while parsing the log. The second plug-in first builds a transition system and then uses Petrify to construct a Petri net [2]. When using the default settings, both region-based plug-ins produce a result similar to the three discovered models shown in Figures 2 and 3. The approach described in [2] offers dozens of strategies to build transition systems to balance between "overfitting" (too specific) and "underfitting" (too general). Using the default settings, the corresponding plug-in represents each state as the set of activities that already have taken place. Based on this state representation, it is easy to construct a transition system and then use the approach presented in [8] to synthesize a labeled Petri net. If we apply this to the initial log (L), we obtain the net shown in Figure 2 (i.e., the result is identical to the net generated by the α-algorithm [4]). However, if we apply the region-based plug-in to L', the result is quite different as shown in Figure 4. The discovered model assumes that all shipped orders will be approved and never rejected, i.e., the occurrence of s (ship) implies the occurrence of ap (approved) and excludes rj (rejected) and rs (resolve). This happens to be the case in this particular set of traces (L'). However, it shows how sensitive the approach is to the occurrence of combinations of events. The three algorithms described earlier, i.e., the α miner [4] (Figure 2), the multi-phase miner [13] (left-hand-side of Figure 3), and the heuristics miner [22] (right-hand-side of Figure 3), all produce the original model for L' showing that they are less sensitive to the occurrence of combinations of events.

The examples show that there is delicate balance between "overfitting" and "underfitting". Region-based approaches tend to consider everything that did not occur in the log as "negative examples", i.e., if the exact behavior did not happen in a particular log, it will never happen. Therefore, these approaches need to be adapted as described in [2]. However, even for these adapted region-based approaches [2] and dedicated process mining techniques [3, 4, 5, 7, 10, 13, 17, 22], the discovery of real-life processes remains a challenge, as shown in the next section.

4. The Process Spaghetti Problem

The fundamental idea of process mining is both simple and persuasive: There is a process which is unknown to us, but we can follow the traces of its behavior, i.e. we have access to enactment logs. Feeding those into a process mining technique will yield an aggregate description of that observed behavior, e.g. in form of a process model.

In the beginning of process mining research, mostly artificially generated logs were used to develop and verify mining algorithms. Then, also logs from real-life workflow management systems, e.g. Staffware, could be successfully mined with these techniques. Early mining algorithms had high requirements towards the qualities of log files, e.g. they were supposed to be complete and limited to events of interest. Yet, most of the resulting problems could be easily remedied with more data, filtering the log and tuning the algorithm to better cope with problematic data.

While these successes were certainly convincing, most real-life processes are not executed within rigid, inflexible workflow management systems and the like, which enforce correct, predictive behavior. It is the inherent inflexibility of these systems which drove the majority of process owners (i.e., organizations having the need to support processes) to choose more flexible or ad-hoc solutions. Concepts like

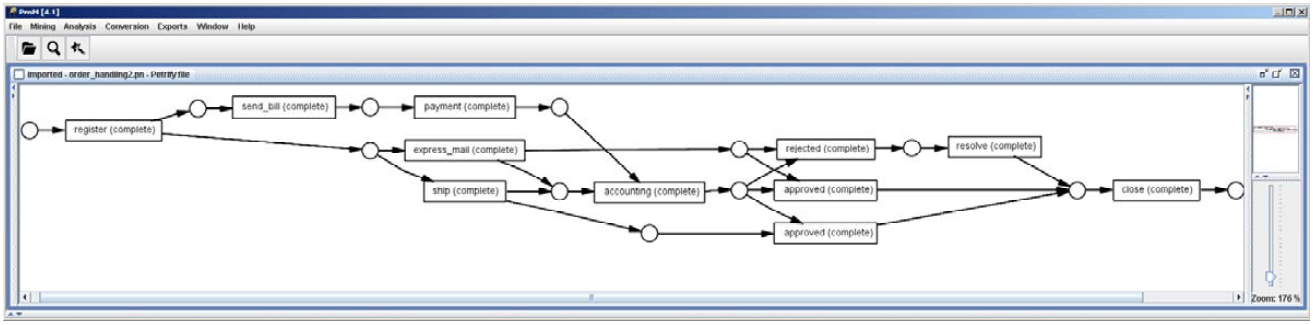

Figure 4. A Petri net discovered by ProM using the region-based approach described in [2].

Adaptive Workflow or Case Handling either allow users to change the process at runtime, or define processes in a somewhat more "loose" manner which does not strictly define a specific path of execution. Yet the most popular solutions for supporting processes do not enforce any defined behavior at all, but merely offer functionality like sharing data and passing messages between users and resources. Examples for these systems are ERP (Enterprise Resource Planning) and CSCW (Computer-Supported Cooperative Work) systems, custom-built solutions, or plain E-Mail.

It is obvious that *executing a process within such less restrictive environments will lead to more diverse and less-structured behavior*. This abundance of observed behavior, however, unveiled a fundamental weakness in most of the early process mining algorithms. When these are used to mine logs from less-structured processes, the result is usually just as unstructured and hard to understand. These "spaghetti" process models do not provide any meaningful abstraction from the event logs themselves, and are therefore useless to process analysts. It is important to note that these "spaghetti" models are not incorrect. The problem is that *the processes themselves* are really "spaghetti-like", i.e., the model is an accurate reflection of reality.

An example of such a "spaghetti" model is given in Figure 5. This figure shows only a small excerpt (ca. 25%) of a highly unstructured process model. It has been mined from machine test logs using the Heuristics Miner, one of the traditional process mining techniques which is most resilient towards noise in logs [22]. Although this result is rather useful, certainly in comparison with other early process mining techniques, it is plain to see that deriving helpful information from it is not easy.

Event classes found in the log are interpreted as activity nodes in the process model. Their sheer amount makes it difficult to focus on the interesting parts of the process. The abundance of arcs in the model, which constitute the actual "spaghetti", introduce an even greater challenge for interpretation. Separating cause from effect, or the general

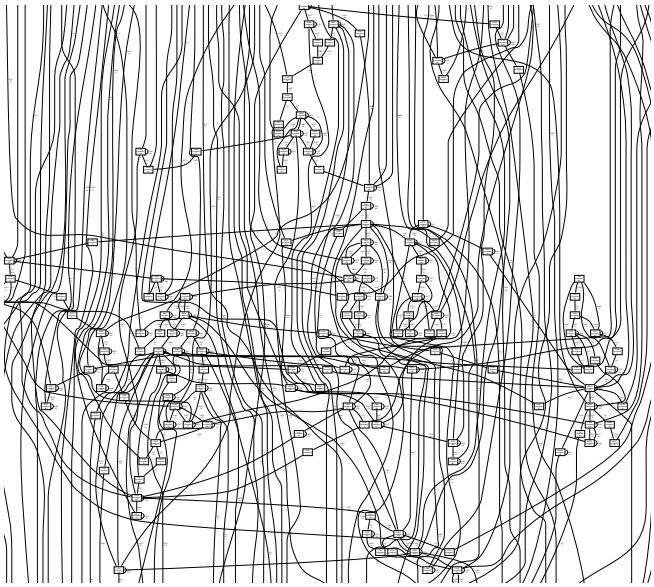

Figure 5. Excerpt of a typical "Spaghetti" process model (ca. 25% of complete model).

direction in which the process is executed, is not possible because virtually every node is transitively connected to any other node in both directions. This mirrors the crux of flexibility in process execution – when people are free to execute anything in any given order they will usually make use of such feature, which renders monitoring business activities an essentially infeasible task.

We argue that the fault for these problems lies neither with less-structured processes, nor with process mining itself. Rather, it is the result of a number of, mostly implicit, assumptions which process mining has historically made, both with respect to the event logs under consideration and regarding the processes which have generated them. While being perfectly sound in structured, controlled environments, *these assumptions do not hold in less-structured, real-life environments* and thus ultimately make traditional

process mining fail there.

Assumption 1: **All logs are reliable and trustworthy**. Any event type found in the log is assumed to have a corresponding logical activity in the process. However, activities in real-life processes may raise a random number of seemingly unrelated events. Activities may also go unrecorded, while other events do not correspond to any activity at all.
The assumption that logs are well-formed and homogeneous is also often not true. For example, a process found in the log is assumed to correspond to one logical entity. In less-structured environments, however, there are often a number of "tacit" process types which are executed, and thus logged, under the same name.
Also, the idea that all events are raised on the same level of abstraction, and are thus equally important, is not true in real-life settings. Events on different levels are "flattened" into the same event log, while there is also a high amount of informational events (e.g., debug messages from the system) which need to be disregarded.

Assumption 2: **There exists an exact process which is reflected in the logs**. This assumption implies that there is the one perfect solution out there, which needs to be found. Consequently, the mining result should model the process *completely*, *accurately*, and *precisely*. However, as stated before, spaghetti models are not necessarily incorrect – the models look like spaghetti, because they precisely describe every detail of the less-structured behavior found in the log. A more high-level solution, which is able to abstract from details, would thus be preferable.
Traditional mining algorithms have also been confined to a single *perspective* (e.g., control flow, data), as such isolated view is supposed to yield higher precision. However, perspectives are interacting in less-structured processes, e.g. the data flow may complement the control flow, and thus also needs to be taken into account.
In general, the assumption of a perfect solution is not well-suited for real-life application. Reality often differs significantly from theory, in ways that had not been anticipated. Consequently, useful tools for practical application must be *explorative*, i.e. support the analyst to tweak results and thus capitalize on their knowledge.

We have conducted process mining case studies in organizations like Philips Medical Systems, UWV, Rijkswaterstaat, the Catharina Hospital Eindhoven and the AMC hospital Amsterdam, and various Dutch municipalities. Our experiences in these case studies have shown the above assumptions to be violated in all ways imaginable. Therefore, to make process mining a useful tool in practical, less-structured settings, these assumptions need to be discarded. The next section outlines a novel mining approach which takes these lessons into account.

5. Bringing Structure to the Unstructured

Process mining techniques which are suitable for less-structured environments need to be able to provide a high-level view on the process, abstracting from undesired details. The field of cartography has always been faced with a quite similar challenge, namely to simplify highly complex and unstructured topologies. Activities in a process can be related to locations in a topology (e.g., towns or road crossings) and precedence relations to traffic connections between them (e.g., railways or motorways).

Figure 6. Example of a road map.

When one takes a closer look at maps (such as the example in Figure 6), the solution cartography has come up with to simplify and present complex topologies, one can derive a number of valuable concepts from them.

Aggregation: To limit the number of information items displayed, maps often show *coherent clusters of low-level detail information* in an aggregated manner. One example are cities in road maps, where particular houses and streets are combined within the city's transitive closure (e.g., the city of Eindhoven in Figure 6).

Abstraction: Lower-level information which is *insignificant in the chosen context* is simply omitted from the visualization. Examples are bicycle paths, which are of no interest in a motorist's map.

Emphasis: More significant information is *highlighted* by visual means such as *color*, *contrast*, *saturation*, and *size*. For example, maps emphasize more important roads by displaying them as thicker, more colorful and contrasting lines (e.g., motorway "E25" in Figure 6).

Customization: There is no one single map for the world. Maps are specialized on a defined *local context*, have a specific *level of detail* (city maps vs highway maps),

and a dedicated *purpose* (interregional travel vs alpine hiking).

These concepts are universal, well-understood, and established. In this paper we explore how they can be used to simplify and properly visualize complex, less-structured processes. To do that, we need to develop appropriate decision criteria on which to base the simplification and visualization of process models. We have identified two fundamental *metrics* which can support such decisions: (1) *significance* and (2) *correlation*.

Significance, which can be determined both for event classes (i.e., activities) and binary precedence relations over them (i.e., edges), measures the *relative importance* of behavior. As such, it specifies the level of interest we have in events, or their occurring after one another. One example for measuring significance is by frequency, i.e. events or precedence relations which are observed more frequently are deemed more significant.

Correlation on the other hand is only relevant for precedence relations over events. It measures *how closely related* two events following one another are. Examples for measuring correlation include determining the overlap of data attributes associated to two events following one another, or comparing the similarity of their event names. More closely correlated events are assumed to share a large amount of their data, or have their similarity expressed in their recorded names (e.g. "check_customer_application" and "approve_customer_application").

Based on these two metrics, we can sketch our approach for process simplification as follows.

- *Highly significant* behavior is *preserved*, i.e. contained in the simplified model.
- *Less significant* but *highly correlated* behavior is *aggregated*, i.e. hidden in clusters within the simplified model.
- *Less significant* and *lowly correlated* behavior is *abstracted from*, i.e. removed from the simplified model.

This approach can greatly reduce and focus the displayed behavior, by employing the concepts of aggregation and abstraction. Based on such simplified model, we can employ the concept of *emphasis*, by highlighting more significant behavior.

Figure 7 shows an excerpt from a simplified process model, which has been created using our approach. Bright square nodes represent significant activities, the darker octagonal node is an aggregated cluster of three less-significant activities. All nodes are labeled 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 labeled with

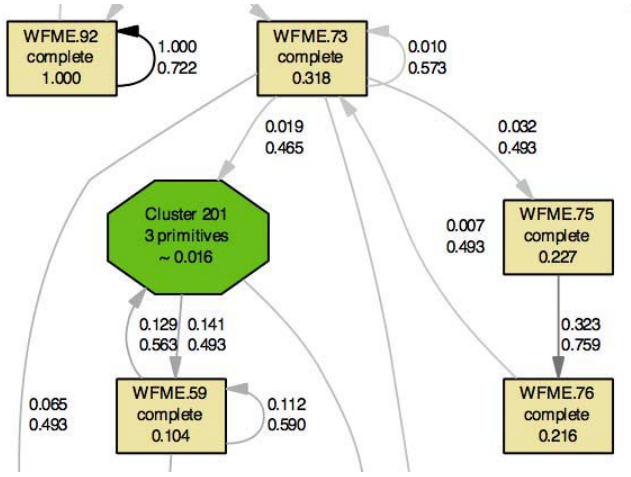

Figure 7. Excerpt of a simplified and decorated process model.

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 behavior in the process.

Yet, the question of what constitutes "interesting" behavior can have a number of answers, based on the process, the purpose of analysis, or the desired level of abstraction. In order to yield the most appropriate result, significance and correlation measures need to be configurable. We have thus developed a set of metrics, which can each measure significance or correlation based on different perspectives (e.g., control flow or data) of the process. By influencing the "mix" of these metrics and the simplification procedure itself, the user can *customize* the produced results to a large degree.

We have implemented our approach as the Fuzzy Miner plugin for the ProM framework [14]. Figure 8 shows the result view of the Fuzzy Miner, with the simplified graph view on the left, and a configuration pane for simplification parameters on the right. By setting a threshold value, the user can determine how much behavior will be displayed explicitly, i.e. how aggressively the process model will be simplified.

Note that this approach is the result of valuable lessons learnt from a great number of case studies with real-life logs. As such, both the applied metrics and the simplification algorithm have been optimized using large amounts actual, less-structured data. While it is difficult to validate the approach formally, the Fuzzy Miner has already become one of the most useful tools in case study applications.

For example, Figure 8 shows the result of applying the Fuzzy Miner to a large test log of manufacturing machines (155.296 events in 16 cases, 826 event classes). Unlike

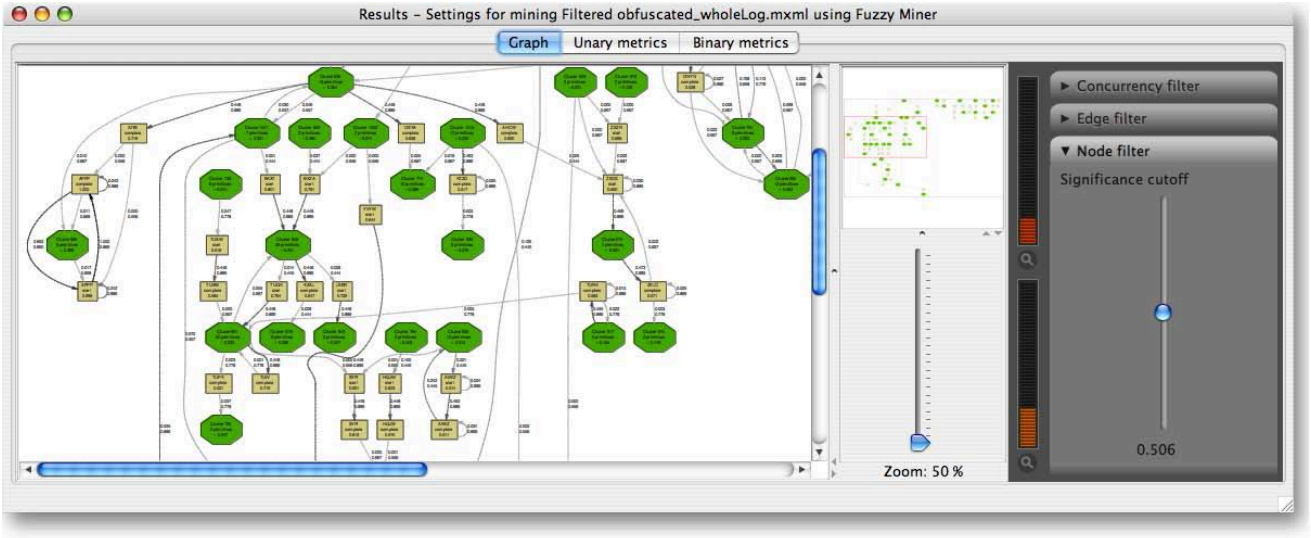

Figure 8. Screenshot of the Fuzzy Miner, applied to the very large and unstructured log also used for mining the model in Figure 5

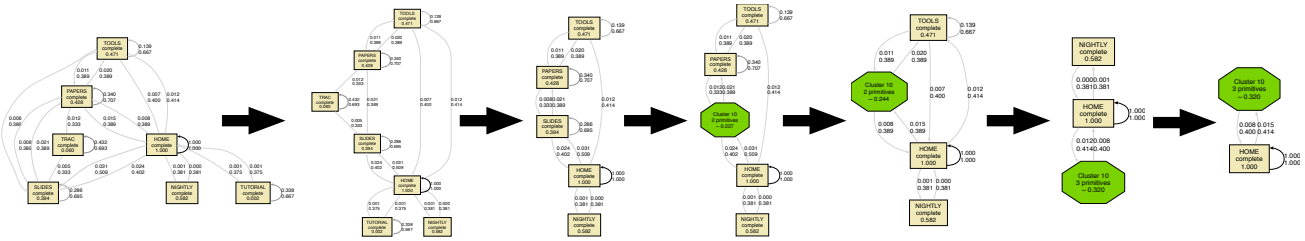

Figure 9. Progressive simplification of a small, yet highly unstructured process model.

the excerpt shown in Figure 5, Figure 8 shows the whole process (and not just an excerpt) and was obtained without filtering the log. It is obvious that our approach is able to clean up a large amount of confusing behavior, and to infer and extract structure from what is chaotic. We have successfully used the Fuzzy Miner on various machinery test and usage logs, development process logs, hospital patient treatment logs, logs from case handling systems and web servers, among others.

Figure 9 shows a sequence of progressively aggressive simplifications created by our approach. The example used here is a web server access log, i.e. each process instance is a user browsing through sections of our web site. The leftmost process model is non-simplified, i.e. describes precisely the behavior found in the log. By removing the least significant edges between nodes, our approach already infers a coarse structure in the first simplified model. The remaining steps visualize the gradual simplification of the model, by removing and clustering activity nodes, down to only one non-simplified activity node.

Our approach abandons the fundamental assumptions of traditional process mining introduced in Section 4: *We do not assume the information recorded in event logs to be flawless and trustworthy, but rather employ a set of configurable metrics*. When tuned to a specific process and analysis question, these metrics can correctly capture the *significance* and *correlation* of model elements. Based on this information, we have developed a method which can simplify a process model. *Rather than assuming one perfect solution, this approach allows the user to interactively explore the process and find an appropriate visualization*.

Explorative and interactive tools like the Fuzzy Miner perfectly complement the set of traditional mining algorithms. Once a suitable approximate process model has been found, it is much easier to filter the information in a log so that it reflects this model. Using such filtered logs, also more formal and exact methods may be applied, which allow to e.g. analyze the correctness or conformance of the process.

6. Conclusion

The message of this paper is twofold. First of all, it was shown that *classical synthesis approaches cannot be used for process mining* since they have a tendency to "overfit" (i.e., if a combination of events did not happen in the log, it is considered impossible). Hence, dedicated process mining techniques are needed that balance between "overfitting" and "underfitting". Second, even the best process discovery techniques tend to *produce "spaghetti-like models" for real-life processes*. The reason is that these processes *are* complicated and unstructured. Therefore, we proposed a technique which is able to simplify the models presented to the user. Comparable to the drawing of road maps we use *abstraction* and *clustering* to produce models that are comprehensible.

The ideas presented in this paper have been realized in the context of the open-source tool ProM. ProM provides a wide range of process mining techniques and can be downloaded from `www.processmining.org`.

Acknowledgements
This research is supported by EIT, NWO-EW, the Technology Foundation STW, and the SUPER project (FP6). Moreover, we would like to thank the many people involved in the development of ProM.

References

[1] W. van der Aalst, H. Reijers, and M. Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative work*, 14(6):549–593, 2005.

[2] W. van der Aalst, V. Rubin, B. Dongen, E. Kindler, and C. Günther. Process Mining: A Two-Step Approach using Transition Systems and Regions. BPM Center Report BPM-06-30, BPMcenter.org, 2006.

[3] W. van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.

[4] W. van der Aalst, A. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[5] R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.

[6] E. Badouel, L. Bernardinello, and P. Darondeau. The Synthesis Problem for Elementary Net Systems is NP-complete. *Theoretical Computer Science*, 186(1-2):107–134, 1997.

[7] J. Cook and A. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

[8] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers*, 47(8):859–882, Aug. 1998.

[9] P. Darondeau. Unbounded Petri Net Synthesis. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 413–438. Springer-Verlag, Berlin, 2004.

[10] A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.

[11] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

[12] J. Desel and W. Reisig. The Synthesis Problem of Petri Nets. *Acta Informatica*, 33(4):297–315, 1996.

[13] B. van Dongen and W. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T. Ling, editors, *International Conference on Conceptual Modeling (ER 2004)*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.

[14] B. van Dongen, A. Medeiros, H. Verbeek, A. Weijters, and W. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.

[15] M. Dumas, W. van der Aalst, and A. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.

[16] A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.

[17] J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.

[18] R. Lorenz and G. Juhas. Towards Synthesis of Petri Nets from Scenariose. In S. Donatelli and P. Thiagarajan, editors, *Application and Theory of Petri Nets 2006*, volume 4024 of *Lecture Notes in Computer Science*, pages 302–321. Springer-Verlag, Berlin, 2006.

[19] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

[20] A. Rozinat and W. van der Aalst. Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In C. Bussler et al., editor, *BPM 2005 Workshops (Workshop on Business Process Intelligence)*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2006.

[21] A. Rozinat and W. van der Aalst. Decision Mining in ProM. In S. Dustdar, J. Faideiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer-Verlag, Berlin, 2006.

[22] A. Weijters and W. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.