Service Mining: Using Process Mining to Discover, Check, and Improve Service Behavior

Wil van der Aalst, *Senior Member, IEEE*

Abstract—Web services are an emerging technology to implement and integrate business processes within and across enterprises. Service-orientation can be used to decompose complex systems into loosely coupled software components that may run remotely. However, the distributed nature of services complicates the design and analysis of service-oriented systems that support end-to-end business processes. Fortunately, services leave trails in so-called *event logs* and recent breakthroughs in *process mining* research make it possible to discover, analyze, and improve business processes based on such logs.
Recently, the *Task Force on Process Mining* released the *Process Mining Manifesto*. This manifesto is supported by 53 organizations and 77 process mining experts contributed to it. The active participation from end-users, tool vendors, consultants, analysts, and researchers illustrate the growing significance of process mining as a bridge between data mining and business process modeling.
In this paper, we focus on the opportunities and challenges for *service mining*, i.e., applying process mining techniques to services. We discuss the guiding principles and challenges listed in the Process Mining Manifesto and also highlight challenges specific for service-orientated systems.

Index Terms—Process Mining, Business Process Management, Service Discovery, Conformance Checking

1 Introduction

Web services have emerged as an established paradigm for architecting and implementing business collaborations within and across organizational boundaries [1], [2]. In this paradigm, the functionality provided by business applications is encapsulated within web services, i.e., software components described at a semantic level, which can be invoked by application programs or by other services through a stack of Internet standards including HTTP, XML, SOAP, WSDL and UDDI [1], [2], [3], [4], [5]. Once deployed, web services provided by various organizations can be inter-connected in order to implement business collaborations, leading to composite web services.

In the context of web services, typically all kinds of *events* are being recorded. It is possible to record events related to *activities inside services* or *interactions between services* (e.g., messages) [6], [7], [8], [9]. The autonomous nature of services and the fact that they are loosely coupled makes it important to monitor and analyze their behavior. In this paper, we will refer to this as *service mining*.

Process mining is an enabling technology for service mining. Process mining aims to *discover, monitor and improve real processes by extracting knowledge from event logs* readily available in today's information systems [10], [11], [12].

Starting point for process mining is an *event log*. Each event in such a log refers to an *activity* (i.e., a well-defined step in some process) and is related to a particular *case* (i.e., a process instance). The events belonging to a case are ordered and describe one "run" of the process. Event logs may store additional information about events. In fact, whenever possible, process mining techniques use supplementary information such as the *resource* (i.e., person, device, or software component) executing or initiating the activity, the *timestamp* of the event, and other *data attributes* (e.g., the size of an order).

Typically, three types of process mining can be distinguished: (a) *process discovery*, (b) *conformance checking*, and (c) *model enhancement* [10]. Discovery techniques learn a model from an event log without using any additional information. This results in a so-called initial process model. This model can also be made by hand. In both situations, conformance checking techniques can be used to compare the observed behavior (event logs) with the modeled behavior (initial process model). This results in diagnostics showing deviations between model and log. After conformance checking, model and log are aligned and information from the log may be used to enhance the model. The model may be repaired or extended with other perspectives such as the time or resource perspective. For example, timestamps in the event log may be used to add timing information (waiting times and service times)

• *Wil van der Aalst is with the Department of Mathematics and Computer Science of Eindhoven University of Technology in The Netherlands and Queensland University of Technology in Brisbane, Australia. E-mail: see vdaalst.com*

to the model. Such an extended model can be used for decision support.

Recently, the IEEE Task Force on Process Mining released the *Process Mining Manifesto* [13]. This manifesto describes *guiding principles* and *challenges* for process mining. In this paper, we will summarize these and relate them to service mining. In particular, we will highlight two challenges specific for service mining: (a) "How to Correlate Instances?" and (b) "How to Analyze Services Out of Context?".

The first challenge looks at the problem that process instances (i.e., cases) in one service need to be related to process instances in other services. For example, one customer order may relate to several deliveries in another service. It does not make much sense to look at one service in isolation. Therefore, *correlation* between services is important.

The second challenge specific for service mining is the problem that one can only observe services running in a particular environment. The behavior may change dramatically when a service is embedded in another environment. The performance or correctness of a service in one context may say very little about the performance or correctness of the very same service in another context.

The remainder of this paper is organized as follows. Section 2 briefly introduces our view on service orientation. Then, we introduce the topic of process mining using an example involving two services (Section 3). Section 4 summarizes the guiding principles and challenges described in the Process Mining Manifesto by the IEEE Task Force on Process Mining. In Section 5 we elaborate on the two challenges specific for service mining. Section 6 concludes the paper.

2 Services

Although the term "web services" is often associated to a stack of Internet standards, we take a more broader perspective in this paper. Service-orientation does not depend on a particular technology as is eloquently described in the introduction of the "Service Sciences Research Manifesto" [14]. The key idea of service-orientation is to *subcontract work to specialized services in a loosely coupled fashion*.

In a *Service Oriented Architecture* (SOA) services are interacting, e.g., by exchanging messages. By combining basic services more complex services can be created [1], [2]. *Orchestration* is concerned with the composition of services seen from the viewpoint of single service (the "spider in the web"). *Choreography* is concerned with the composition of services seen from a global viewpoint focusing on the common and complementary observable behavior. Choreography is particularly relevant in a setting where there is no single coordinator.

The terms orchestration and choreography describe two aspects of integrating services to create end-to-end business processes. The two terms overlap somewhat and the distinction has been heavily discussed in the last decade. Orchestration and choreography can be seen as different "perspectives". Choreography is concerned with the overall set of interactions between several services that do not need to be in some hierarchical relationship to one another. Orchestration is concerned with the interactions of a single service with its environment (classical composition).

A service X may have both a *sell side* and *buy side*. The sell side describes the interface that the service offers to its service consumers, i.e., other services that use X. The buy side describes the interface that is used to interact with other service providers, i.e., the services used by X. A service having both a sell side and a buy side may act as a *service producer* and a *service consumer*.

Interactions between services may be *synchronous* or *asynchronous*. In the context of web services, most interactions are asynchronous and realized through message passing. We consider three types of communication primitives: *synchronize* (synchronous communication), *send* (asynchronous communication), and *receive* (asynchronous communication) [15].

In many service-oriented systems it is possible to record communicative actions. For example, it may be possible to tap-off the messages that are being exchanged. In some systems it is also possible to observe the activities inside a service. For example, middleware products and BPM/WFM systems often provide detailed audit trails covering all activities supported [9].

It is relatively easy to collect all events related to an orchestration as one service serves as the "spider in the web". For choreographies this is often more difficult as control is distributed. Even when each service records events in a detailed manner, it may still be impossible to merge all event data into a single overall event log.

In the remainder, we first focus on process mining and the manifesto by the IEEE Task Force on Process Mining. Subsequently, we zoom in on challenges specific for service mining.

3 Using Process Mining for the Analysis of Services

Process mining techniques are able to *extract knowledge from event logs* commonly available in today's information systems [10]. These techniques provide new means to *discover, monitor, and improve processes* in a variety of application domains. In this section we introduce the three basic types of process mining: *process discovery* (Section 3.1), *conformance checking* (Section 3.2), and *model enhancement* (Section 3.3).

3.1 Process Discovery

As indicated earlier, a so-called *event log* serves as a starting point for analysis. An event log can be seen as a collection of events. An example of an event may be:

$caseid$	:	customer order 7564
$activity$	:	place order
$timestamp$	:	28-11-2011@16:54
$resource$	:	John
$amount$	:	€2500
$ordereditems$	:	[iPod, iPhone 4, iPad 2]

Depending on the event log, there may be thousands or even millions of such events. Although all event attributes can be used for process mining, we focus on the first two attributes that are mandatory for process mining. *Any event should refer to a case (i.e., a process instance) and an activity.* Moreover, events related to a particular case should be ordered.

If we limit ourselves to the minimal information needed for process mining, then *an event log can be described as a multiset of traces where each trace corresponds to a sequence of activities*. Given a set of activities $A_1 = \{po, rg, sp, rr, cg, pc, cl\}$, an example trace is $\langle po, rg, cg, sp, pc, cl\rangle$. This trace represents the sequence of activities executed for a particular instance. An example of an event log is $L_1 = [\langle po, rg, cg, sp, pc, cl\rangle^{20}, \langle po, rg, sp, cg, pc, cl\rangle^{18}, \langle po, sp, rg, cg, pc, cl\rangle^{17}, \langle po, sp, rg, pc, cg, cl\rangle^{13}, \langle po, sp, pc, rg, cg, cl\rangle^{12}, \langle po, rg, sp, pc, cg, cl\rangle^{8}, \langle po, rr, cl\rangle^{7}]$. This event log contains 7 different traces describing 95 cases. For example, there are 7 cases that follow trace $\langle po, rr, cl\rangle$.

The goal of process discovery is to discover process models from logs such as L_1. For example, the α algorithm [16] will automatically construct the Petri net depicted in the left subprocess of Fig. 1. This Petri net models that all cases start with activity po ("place order") and end with activity cl ("close"). In-between these two activities either the set of four activities rg, sp, cg and pc, or just activity rr ("receive rejection") is executed. Activity rg is always followed by cg and activity sp is always followed by pc. However, rg may be before or after sp, etc. The left subprocess shown in Fig. 1 allows for seven different traces. These are exactly the traces that can be found in event log L_1.

Now consider the right subprocess shown in Fig. 1. This process has six activities $A_2 = \{ro, sg, rp, cp, gc, sr\}$. An example of an event log over A_2 is $L_2 = [\langle ro, sg, rp, cp, gc\rangle^{88}, \langle ro, sr\rangle^{7}]$. This log also describes 95 cases and can be seen as the counterpart of L_1. The subprocess on the right-hand side of Fig. 1 can be discovered from L_2 using process mining techniques such as the α algorithm [16].

One can view Fig. 1 as a model of two interacting services. Whereas L_1 and L_2 describe the event logs of individual services, one can also look at an event log describing the combined overall behavior. An example of a trace possible according to Fig. 1 is $\sigma_1 = \langle po, ro, sg, rg, cg, sp, rp, cp, pc, gc, cl\rangle$. Another example trace is $\sigma_2 = \langle po, ro, sr, rr, cl\rangle$. If we project σ_1 and σ_2 onto A_1, we find traces present in L_1. Projecting these traces onto A_2 yields traces in L_2.

In our example logs, events refer to activity names. An event log could also hold events related to messages being sent or received. In some logs one may even find events related to the start and completion of activities. This way one can measure the duration of activities. Fig. 1 only shows asynchronous communication as both services are connected through message places $m1, \ldots, m6$. It is also possible to model synchronous interaction by transition fusion. For example, if po and ro interact synchronously, one can model this by merging both transitions into one. However, all of these variations do not change the basic idea of process discovery for services: automatically learning the behavior of services based on event data.

One could argue that there is no need to discover services since they have been implemented before. Instead of process discovery based on event logs, one could also refactor models based on source code or documentation. However, given the autonomous nature of services, one party may not have models of the services of another party, services may change their behavior over time, or only use a fraction of the possible modeled behavior. Therefore, it may be more reliable and efficient to discover service behavior from real event data. Moreover, the discovered model is only the starting point for other types of analysis as is explained next. The alignment of real event data and models is the key ingredient of process mining.

3.2 Conformance Checking

Combining services to realize desired end-to-end processes is a far from trivial and error prone process. It is easy to inadvertently introduce deadlocks and other behavioral anomalies. For example, the combined model in Fig. 1 contains a subtle error. This error has been added to illustrate the need for verification techniques. Consider the trace: $\langle po, ro, sp, sr\rangle$. This trace results in the state where the places $p2$, $p5$, $m3$, $m4$, and $p14$ each contain one token, i.e., the process on the right rejected the order whereas the process on the left already paid for it. The resulting state is a so-called deadlock.

The two individual processes have no such problems when analyzed in isolation. The left subprocess has 11 possible states and it is always possible to end with a token in $p8$. The right subprocess has 6 states and it is always possible to end with a token in $p14$. The combined process has 26 reachable states. There are two states where no transitions are enabled: the desired final state with tokens in $p8$ and $p14$ and the deadlock just described.

The lion's share of attention in service analysis has been devoted to design-time verification. However, correctness issues are not limited to design-time. *At run-time the actual behavior may deviate from the modeled behavior.* Conformance checking techniques aim

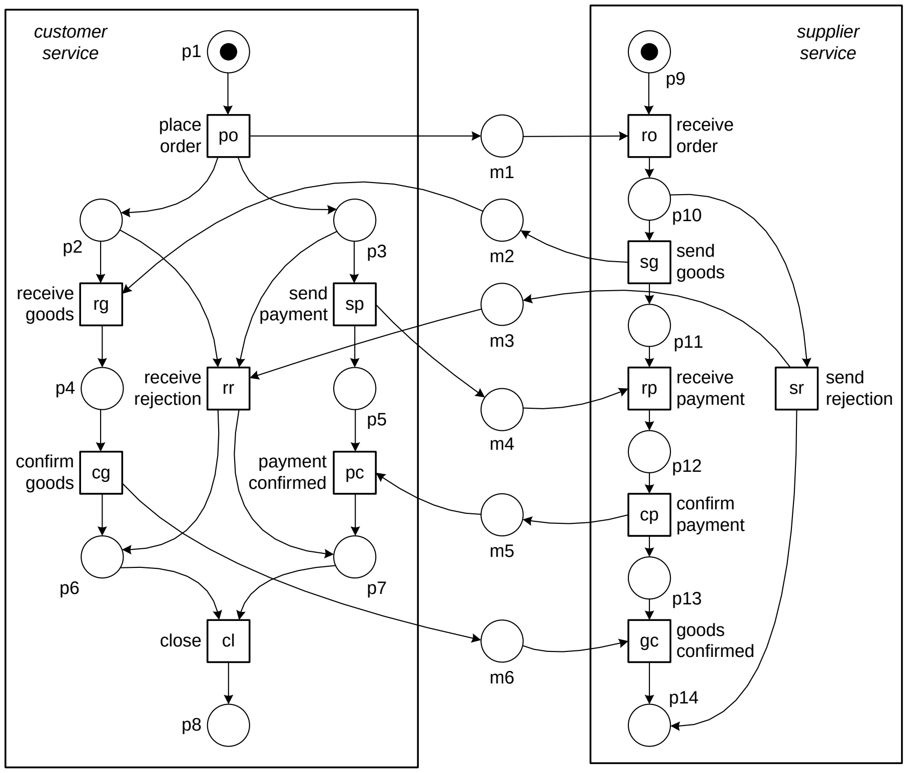

Fig. 1. A process composed of two services modeled in terms of Petri nets.

to find discrepancies between the modeled behavior and the real behavior recorded in event logs. For example, suppose that an event log contains the trace: $\langle po, rg, ro, sg, rp, cp, cg, sp, pc, gc, cl\rangle$. This trace is not possible according to the combined model in Fig. 1. First of all, the goods are received by the customer service without ever being sent by the supplier service. Second, the payment is received before being sent.

Figure 2 shows an alternative model for the customer service. If the original customer service in Fig. 1 is replaced by this service, the deadlock is removed. Hence, from a (design-time) correctness point of view the service in Fig. 2 is better than the original one. However, if event log $L1$ (the log describing 95 cases) describes the observed behavior, then conformance is poor because many traces in the log are impossible according to the model. Only the eight cases following trace $\langle po, rg, sp, pc, cg, cl\rangle$ and the seven cases following trace $\langle po, rr, cl\rangle$ fit completely. The other $95 - (8 + 7) = 80$ cases deviate from the model and, therefore, do not fit.

Conformance can be viewed from two angles: (a) the model does not capture the real behavior ("the model is wrong") and (b) reality deviates from the desired model ("the event log is wrong"). The first viewpoint is taken when the model is supposed to be *descriptive*, i.e., capture or predict reality. The second viewpoint is taken when the model is *normative*, i.e., used to influence or control reality.

Conformance checking is very important for compliance checking, auditing, certification, and run-time monitoring. Moreover, it can be used to judge the quality of discovered or hand-made models. Typically,

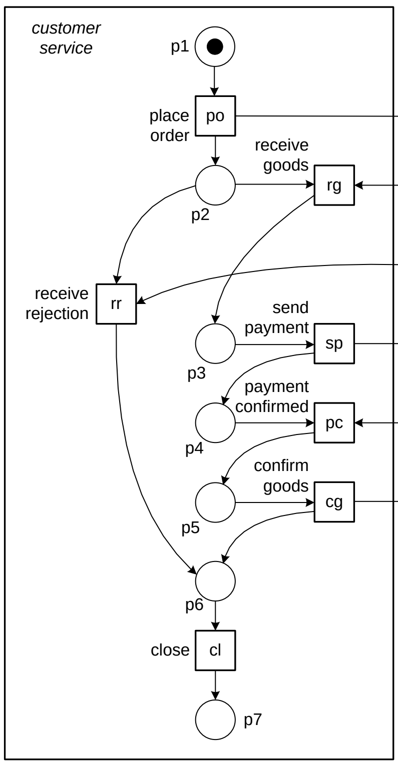

Fig. 2. A sequential customer service.

four quality dimensions for comparing model and log are considered: (a) *fitness*, (b) *simplicity*, (c) *precision*, and (d) *generalization* [10].

A model with good *fitness* allows for most of the behavior seen in the event log. A model has perfect fitness if all traces in the log can be replayed by the model from beginning to end. There are various ways of quantifying fitness [10], [17], [18]. Often fitness is described by a number between 0 (very poor fitness) and 1 (perfect fitness).

Obviously, the *simplest* model that can explain the behavior seen in the log is the best model. This principle is known as Occam's Razor.

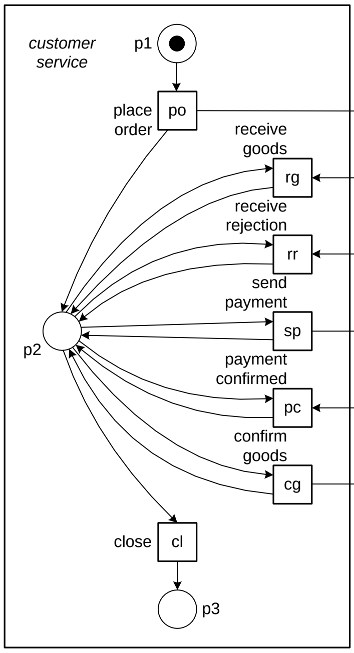

Fig. 3. A so-called "flower model" allowing for too much behavior.

Fitness and simplicity alone are not sufficient to judge the quality of a discovered process model. For example, it is very easy to construct an extremely simple Petri net ("flower model") that is able to replay all traces in an event log (but also any other event log referring to the same set of activities). See for example the customer service shown in Fig. 3. After po all other transitions are enabled and remain enabled until the end. For example, traces such as $\langle po, cl\rangle$ and $\langle po, rg, rg, rg, rg, cl\rangle$ are possible according to this "flower model" but very unlikely given the observed behavior recorded in event log L_1.

Similarly, it is often undesirable to have a model that only allows for the exact behavior seen in the event log. Remember that the log contains only example behavior and that many traces that are possible may not have been seen yet. (Note that in our simple example all traces are frequent, so the problem does not apply here.)

A model is *precise* if it does not allow for "too much" behavior. Clearly, the "flower model" lacks precision. A model that is not precise is "underfitting". Underfitting is the problem that the model over-generalizes the example behavior in the log (i.e., the model allows for behaviors very different from what was seen in the log).

A model should also *generalize* and not restrict behavior to just the examples seen in the log. A model that does not generalize sufficiently is "overfitting". Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior (i.e., the model explains the particular sample log, but a next sample log of the same process may produce a completely different process model).

All four quality dimensions for comparing model and log can be quantified in various ways. See [10], [17], [18] for more details. Also see [7], [8] for a more declarative constraint-based modeling language suitable for conformance checking in the context of services.

3.3 Model Enhancement

It is also possible to extend or improve an existing process model using the event log. A non-fitting process model can be corrected using the diagnostics provided by the alignment of model and log. Moreover, event logs may contain information about resources, timestamps, and case data. For example, an event referring to activity "place order" and case "customer order 7564" may also have attributes describing the person that entered the order (e.g., "John"), the time of the event (e.g., "28-11-2011@16:54"), the ordered amount (e.g., "€2500"), and the multiset of items ordered. After aligning model and log it is possible to *replay* the event log on the model. While replaying one can analyze these additional attributes and add other perspectives to the model.

For example, as Fig. 4 shows, it is possible to analyze *waiting times* in-between activities. Simply measure the time differences between causally related events and compute basic statistics such as averages, variances, and confidence intervals. This way it is possible to identify the main bottlenecks [10], [19].

Information about resources can be used to discover *roles*, i.e., groups of people frequently executing related activities [20]. Here, standard clustering techniques can be used. It is also possible to construct social networks based on the flow of work and analyze resource performance (e.g., the relation between workload and service times).

Standard classification techniques can be used to analyze the *decision points* in a process model. For example, after receiving the order (place $p10$) there are two possibilities: the goods are sent (sg) or a rejection is sent (sr). Using the data known about the case prior to the decision, we can construct a decision tree explaining the observed behavior [10], [21].

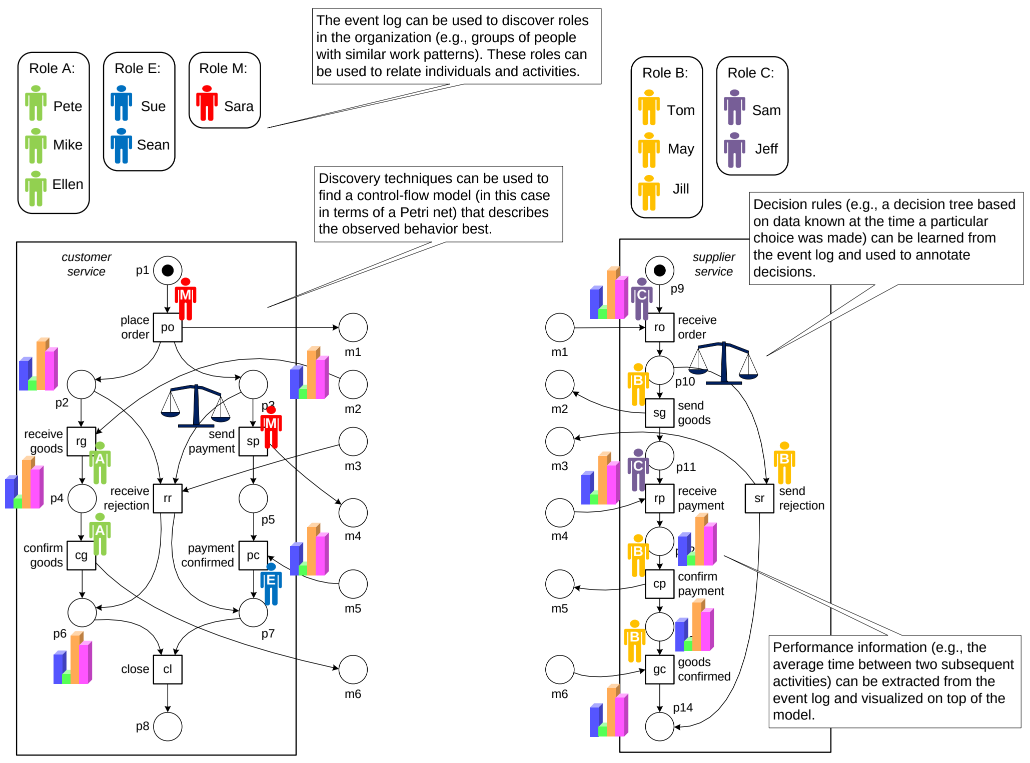
Role A:
Pete
Mike
Ellen
Role E:
Sue
Sean
Role M:
Sara
The event log can be used to discover roles in the organization (e.g., groups of people with similar work patterns). These roles can be used to relate individuals and activities.
Role B:
Tom
May
Jill
Role C:
Sam
Jeff
Discovery techniques can be used to find a control-flow model (in this case in terms of a Petri net) that describes the observed behavior best.
Decision rules (e.g., a decision tree based on data known at the time a particular choice was made) can be learned from the event log and used to annotate decisions.
customer service
supplier service
place order
receive goods
send payment
receive rejection
payment confirmed
confirm goods
close
receive order
send goods
receive payment
send rejection
confirm payment
goods confirmed
Performance information (e.g., the average time between two subsequent activities) can be extracted from the event log and visualized on top of the model.

Fig. 4. By replaying the event log on the process model for a service it is possible to analyze bottlenecks, decision points, and the distribution of work over roles and resources.

Figure 4 illustrates that process mining is not limited to control-flow discovery. In fact, process mining may cover a variety of perspectives. The *control-flow perspective* only focuses on the control-flow, i.e., the ordering of activities. The goal of mining this perspective is to find a good characterization of all possible paths. The result is typically expressed in terms of a Petri net or some other process notation (e.g., EPCs, BPMN, or UML activity diagrams). The *organizational perspective* focuses on information about resources hidden in the log, i.e., which actors (e.g., people, systems, roles, or departments) are involved and how are they related. The goal is to either structure the organization by classifying people in terms of roles and organizational units or to show the social network. The *system architecture perspective* focuses on the IT infrastructure, i.e., the decomposition into services and other software components. Techniques for mining the organizational perspective [20] can also be applied to this more technical perspective. For example, social network analysis can be used to visualize the communication among services. The *case perspective* focuses on properties of cases. Obviously, a case can be characterized by its path in the process or by the actors working on it. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order, it may be interesting to know the supplier or the number of products ordered. The *time perspective* is concerned with the timing and frequency of events. When events bear timestamps it is possible to discover bottlenecks, measure service levels, monitor the utilization of resources, and predict the remaining processing time of running cases.

Process mining is not restricted to offline analysis and can also be used for *predictions* and *recommendations* at runtime. For example, the completion time of a partially handled customer order can be predicted using a discovered process model with timing information [19].

4 Process Mining Manifesto

The IEEE Task Force on Process Mining recently released a *manifesto* describing *guiding principles* and *challenges* [13]. The manifesto aims to increase the visibility of process mining as a new tool to improve the (re)design, control, and support of operational business processes. It is intended to guide software developers, scientists, consultants, and end-users. Before summarizing the manifesto, we briefly introduce the task force.

4.1 Task Force on Process Mining

The growing interest in log-based process analysis motivated the establishment of the *IEEE Task Force on Process Mining*. The goal of this task force is to promote the research, development, education, and understanding of process mining. The task force was established in 2009 in the context of the Data Mining

Technical Committee of the Computational Intelligence Society of the IEEE. Members of the task force include representatives of more than a dozen commercial software vendors (e.g., Pallas Athena, Software AG, Futura Process Intelligence, HP, IBM, Fujitsu, Infosys, and Fluxicon), ten consultancy firms (e.g., Gartner and Deloitte) and over twenty universities.

Concrete objectives of the task force are: to make end-users, developers, consultants, managers, and researchers aware of the state-of-the-art in process mining, to promote the use of process mining techniques and tools, to stimulate new process mining applications, to play a role in standardization efforts for logging event data, to organize tutorials, special sessions, workshops, panels, and to publish articles, books, videos, and special issues of journals. For example, in 2010 the task force standardized *XES* (www.xes-standard.org), a standard logging format that is extensible and supported by the *OpenXES library* (www.openxes.org) and by tools such as ProM, XESame, Nitro, etc. See http://www.win.tue.nl/ieeetfpm/ for recent activities of the task force.

4.2 Guiding Principles

As with any new technology, there are obvious mistakes that can be made when applying process mining in real-life settings. Therefore, the six guiding principles listed in Table 1 aim to prevent users/analysts from making such mistakes. As an example, consider guiding principle *GP4*: "Events Should Be Related to Model Elements". It is a misconception that process mining is limited to control-flow discovery, other perspectives such as the organizational perspective, the time perspective, and the data perspective are equally important. However, the control-flow perspective (i.e., the ordering of activities) serves as the layer connecting the different perspectives. Therefore, it is important to relate events in the log to activities in the model. Conformance checking and model enhancement heavily rely on this relationship (cf. sections 3.2 and 3.3). After relating events to model elements, it is possible to "replay" the event log on the model [10]. Replay may be used to reveal discrepancies between an event log and a model, e.g., some events in the log are not possible according to the model. Techniques for conformance checking can be used to quantify and diagnose such discrepancies. Timestamps in the event log can be used to analyze the temporal behavior during replay. Time differences between causally related activities can be used to add average/expected waiting times to the model. These examples illustrate the importance of guiding principle GP4; the relation between events in the log and elements in the model serves as a starting point for different types of analysis.

4.3 Challenges

Process mining is an important tool for modern organizations that need to manage non-trivial operational processes. On the one hand, there is an incredible growth of event data. On the other hand, processes and information need to be aligned perfectly in order to meet requirements related to compliance, efficiency, and customer service. Despite the applicability of process mining there are still important challenges that need to be addressed; these illustrate that process mining is an emerging discipline. Table 2 lists the eleven challenges described in the manifesto [13].

As an example consider Challenge *C4*: "Dealing with Concept Drift". The term *concept drift* refers to the situation in which the process is changing while being analyzed [22]. For instance, in the beginning of the event log two activities may be concurrent whereas later in the log these activities become sequential. Processes may change due to periodic/seasonal changes (e.g., "in December there is more demand" or "on Friday afternoon there are fewer employees available") or due to changing conditions (e.g., "the market is getting more competitive"). Such changes impact processes and it is vital to detect and analyze them [22].

The manifesto is supported by 53 organizations and 77 process mining experts from all over the globe contributed to it. The manifesto is available in Chinese, Dutch, English, French, German, Greek, Italian, Japanese, Korean, Portuguese, Spanish, and Turkish (cf. www.win.tue.nl/ieeetfpm).

5 Specific Challenges for Service Mining

The challenges mentioned in the manifesto apply to wide range of application domains (including services computing). However, for service mining, i.e., applying process mining to services, we also discuss two more specific challenges.

5.1 How to Correlate Instances?

Figure 1 describes two interacting services. Although there may be many orders, the Petri net describes only one instance of the customer service and one instance of the supplier service. These two instances relate to the same order. Because there may be many orders, instances of the customer service need to be related to instances of the supplier service and vice versa. Relating instances of different services is commonly referred to as *correlation*. For example, when the supplier service receives a payment (activity *rp*), this payment needs to be correlated to the order for which the goods have been sent.

In general there is not a one-to-one correspondence between instances (i.e., cases) of different services. This is illustrated in Fig. 5 which sketches instances

TABLE 1
Six Guiding Principles Listed in the Manifesto

GP1	**Event Data Should Be Treated as First-Class Citizens** Events should be *trustworthy*, i.e., it should be safe to assume that the recorded events actually happened and that the attributes of events are correct. Event logs should be *complete*, i.e., given a particular scope, no events may be missing. Any recorded event should have well-defined *semantics*. Moreover, the event data should be *safe* in the sense that privacy and security concerns are addressed when recording the event log.
GP2	**Log Extraction Should Be Driven by Questions** Without concrete questions it is very difficult to extract meaningful event data. Consider, for example, the thousands of tables in the database of an ERP system like SAP. Without questions one does not know where to start.
GP3	**Concurrency, Choice and Other Basic Control-Flow Constructs Should be Supported** Basic workflow *patterns* supported by all mainstream languages (e.g., BPMN, EPCs, Petri nets, BPEL, and UML activity diagrams) are *sequence*, *parallel routing* (AND-splits/joins), *choice* (XOR-splits/joins), and *loops*. Obviously, these patterns should be supported by process mining techniques.
GP4	**Events Should Be Related to Model Elements** Conformance checking and enhancement heavily rely on the relationship between *elements in the model* and *events in the log*. This relationship may be used to "replay" the event log on the model. Replay can be used to reveal discrepancies between event log and model (e.g., some events in the log are not possible according to the model) and can be used to enrich the model with additional information extracted from the event log (e.g., bottlenecks are identified by using the timestamps in the event log).
GP5	**Models Should Be Treated as Purposeful Abstractions of Reality** A model derived from event data provides a *view on reality*. Such a view should serve as a purposeful abstraction of the behavior captured in the event log. Given an event log, there may be multiple views that are useful.
GP6	**Process Mining Should Be a Continuous Process** Given the dynamical nature of processes, it is not advisable to see process mining as a one-time activity. The goal should not be to create a fixed model, but to breathe life into process models such that users and analysts are encouraged to look at them on a daily basis.

TABLE 2
Some of the Most Important Process Mining Challenges Identified in the Manifesto

C1	**Finding, Merging, and Cleaning Event Data** When extracting event data suitable for process mining several challenges need to be addressed: data may be *distributed* over a variety of sources, event data may be *incomplete*, an event log may contain *outliers*, logs may contain events at *different level of granularity*, etc.
C2	**Dealing with Complex Event Logs Having Diverse Characteristics** Event logs may have very different characteristics. Some event logs may be extremely large making them difficult to handle whereas other event logs are so small that not enough data is available to make reliable conclusions.
C3	**Creating Representative Benchmarks** Good benchmarks consisting of example data sets and representative quality criteria are needed to compare and improve the various tools and algorithms.
C4	**Dealing with Concept Drift** The process may be changing while being analyzed. Understanding such concept drifts is of prime importance for the management of processes.
C5	**Improving the Representational Bias Used for Process Discovery** A more careful and refined selection of the representational bias is needed to ensure high-quality process mining results.
C6	**Balancing Between Quality Criteria such as Fitness, Simplicity, Precision, and Generalization** There are four competing quality dimensions: (a) fitness, (b) simplicity, (c) precision, and (d) generalization. The challenge is to find models that score good in all four dimensions.
C7	**Cross-Organizational Mining** There are various use cases where event logs of multiple organizations are available for analysis. Some organizations work together to handle process instances (e.g., supply chain partners) or organizations are executing essentially the same process while sharing experiences, knowledge, or a common infrastructure. However, traditional process mining techniques typically consider one event log in one organization.
C8	**Providing Operational Support** Process mining is not restricted to off-line analysis and can also be used for online operational support. Three operational support activities can be identified: *detect*, *predict*, and *recommend*.
C9	**Combining Process Mining With Other Types of Analysis** The challenge is to combine automated process mining techniques with other analysis approaches (optimization techniques, data mining, simulation, visual analytics, etc.) to extract more insights from event data.
C10	**Improving Usability for Non-Experts** The challenge is to hide the sophisticated process mining algorithms behind user-friendly interfaces that automatically set parameters and suggest suitable types of analysis.
C11	**Improving Understandability for Non-Experts** The user may have problems understanding the output or is tempted to infer incorrect conclusions. To avoid such problems, the results should be presented using a suitable representation and the trustworthiness of the results should always be clearly indicated.

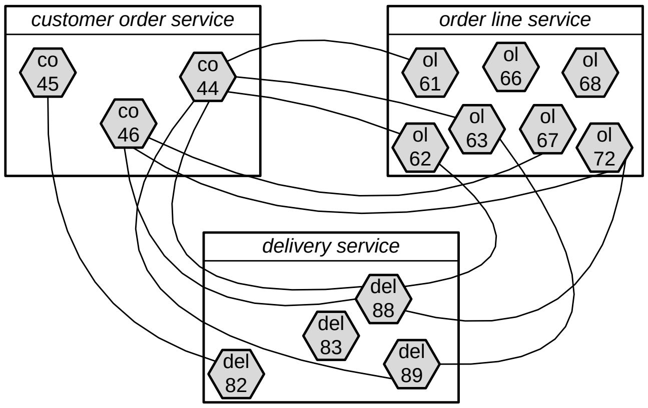

Fig. 5. Three services having several instances each.

of three related services. The customer order service has three instances: $co44$, $co45$, and $co46$. Each of these instances corresponds to a customer order. The customer order service uses an order line service which handles individual order lines. For example, payments are handled at the order level, but order picking is done at the level of individual items. Customer order $co44$ consists of three order lines ($ol61$, $ol62$, and $ol63$), i.e., three items have been ordered. The actual delivery of such items is subcontracted to a delivery service. The delivery service tries to combine multiple items into a single delivery. However, this is not always possible. For example, order $co44$ is related to two delivery instances: $del88$ and $del89$. It may also be the case that one delivery is related to order lines of different customer orders (see for example $del88$). Hence, there is a many-to-many relationship between customer order instances and delivery instances.

Situations similar to the one described in Fig. 5 occur frequently in real-life processes. However, existing process notations and process mining algorithms have difficulties dealing with many-to-many relationships between service instances. In fact, three related problems can be identified. First of all, there is a representation problem. Traditional process notations such as BPMN, EPCs, and UML activity diagrams model the life-cycle of one instance in isolation. Notable exceptions are *proclets* [23] and other *artifact-centric* process models [24]. Second, there is the problem of relating messages to instances. There are various approaches that all suggest analyzing the content of each message. Subsequently, messages are related based on reoccurring pieces of text (e.g., an identifier or a customer name) [6], [25], [26]. Finally, there is a lack of process mining techniques able to discover process models with many-to-many relationships. Some initial work has been done in the context of the ACSI project [27]. For example, it is possible to do conformance checking on proclets [28], [29]. However, better techniques are needed to truly support more complex correlation patterns.

As discussed in [10], instance correlation and process views are closely related. Process models should be seen as views on reality. The scope of such a view depends on the correlations deemed to be relevant. For example, one may want to view customer orders in isolation (abstracting away order lines and deliveries) or view these orders from the viewpoint of deliveries. The empirical nature of process mining helps us to understand the "fabric of real business processes" better whereas conventional process modeling languages are tailored towards straightjacketing instances into monolithic process models.

5.2 How to Analyze Services Out of Context?

For the second challenge we return to the problem that the two interacting services shown in Fig. 1 can deadlock. Although each of the services in isolation has no problem, the composed service deadlocks when trace $\langle po, ro, sp, sr\rangle$ is executed. The reason is that the customer may decide to pay whereas the supplier may decide to reject the order. In fact, there is no supplier service that can work properly with the customer service modeled in Fig. 1. The choice between rg and rr is controlled by the environment via places $m2$ and $m3$. However, independent of this external decision, the customer service may decide to pay (sp). Because asynchronous message passing is used, these two choices cannot be coordinated properly. This illustrates the subtle interplay between a service and it environment.

Let us now consider the two alternative services depicted in Fig. 6. The process resulting from these combined services does not have any deadlocks or other anomalies. Inspection of the state space consisting of 16 states shows that it is always possible to reach the desired final state with tokens in $p8$ and $p16$.

A possible event log generated by the two services in Fig. 6 is $L' = [\langle po, ro, sg, rg, sp, rp, cp, pc, cg, gc, fi, cl\rangle^{40}, \langle po, ro, sg, rg, sp, rp, cp, pc, cg, gc, cl, fi\rangle^{28}, \langle po, ro, sg, rg, sp, rp, cp, pc, cg, cl, gc, fi\rangle^{20}, \langle po, ro, sr, rr, cl\rangle^{7}]$. This log describes 95 cases distributed over four possible traces. Event log L' can be projected onto the customer service (left-hand side of Fig. 6) yielding log $L'_1 = [\langle po, rg, sp, pc, cg, cl\rangle^{88}, \langle po, rr, cl\rangle^{7}]$. Similarly, L' can be projected onto the supplier service resulting in event log $L'_2 = [\langle ro, sg, rp, cp, gc, fi\rangle^{88}, \langle ro, sr\rangle^{7}]$.

Although L'_1 and L'_2 do not reveal any problems, it should be noted that the two services are limiting each other's behavior. For example, a possible trace of the customer service that was not observed is $\langle po, rg, pc, sp, cg, cl\rangle$. This trace could not be observed because the supplier service only confirms the payment after it has been received (which makes sense). Possible traces of the supplier service

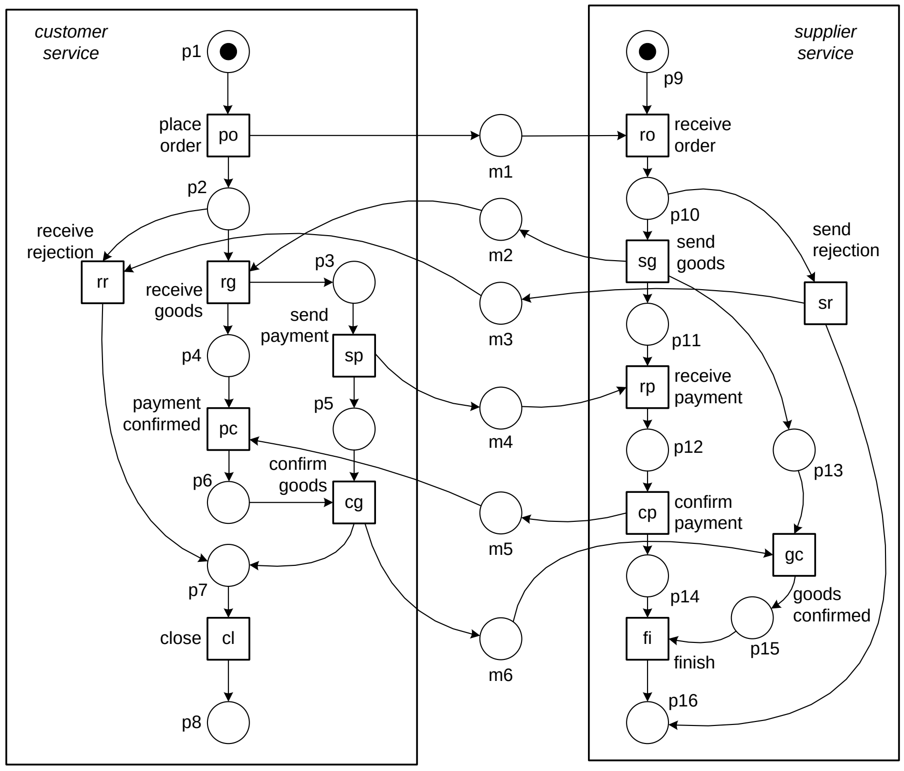

Fig. 6. Another process composed of two services that limit each other's behavior.

that were not observed are $\langle ro, sg, gc, rp, cp, fi \rangle$ and $\langle ro, sg, rp, gc, cp, fi \rangle$. The reason is that the customer service only confirms the receipt of the goods after it has gotten a payment confirmation. These examples show that the environment of a service may severely limit the behavior that can actually be observed. One can only record the behavior of a service in a particular context. This implies that certain anomalies do not surface in one context and only emerge if the environment of the service is changed.

The observation that service behavior is limited by its context has implications for both process discovery and conformance checking.

The discovered process model may only show a fraction of the potential service behavior. For example, based on event log L_2', process discovery techniques will discover a sequential model for the supplier service. The conclusion that the goods can only be confirmed after all payment related activities have completed only holds in the context depicted in Fig. 6.

The same phenomenon makes it difficult to reason about precision. Based on event log L_2', analysis may suggest that the model in Fig. 6 is "underfitting" because only two of the four possible traces are actually observed.

Today's process mining techniques do not take the direction of messages (send or receive) and the nature of choices into account. For example, consider the choice between rr and rg. This choice is controlled by inbound messages $m2$ and $m3$. The choice between sg and sr on the other hand is not controlled by the environment and is made inside the supplier service. The challenge is to adapt discovery techniques and conformance checking techniques to incorporate such distinctions.

5.3 Related Work

After describing the two challenges specific for service mining, we briefly point out related work on the analysis of services based on event data.

In [9] a concrete application of process mining to web services is described. IBM's WebSphere product is used as a reference system and its CEI (Common Event Infrastructure) logs are analyzed using ProM.

An approach to check the conformance of web services was described in [6]. The paper includes a description of various experiments using Oracle BPEL. The token-based replay techniques presented in [18] were used to measure conformance.

In [7], [8] an LTL-based approach to check conformance was proposed. This approach uses a graphical declarative language to describe the normative behavior of services. Rather than modeling a detailed process, this approach allows for checking graphically specified constraints such as "a payment should always be confirmed".

The topic of event correlation has been investigated in the context of system specification, system development, and services analysis. In [15] and [30] various interaction and correlation patterns are described. In

[25] a technique is presented for correlating messages with the goal to visualize the execution of web services.

Dustdar et al. [31], [32], [33] proposed techniques for services interaction mining, i.e., applying process mining techniques to the analysis of service interactions.

Nezhad et al. [26], [34] developed techniques for event correlation and process discovery from web service interaction logs. The authors introduce the notion of a "process view" which is the result of a particular event correlation. However, they argue that correlation is subjective and that multiple views are possible. A collection of process views is called the "process space".

In [35], Simmonds et al. propose a technique for the run-time monitoring of web service conversations. The authors monitor conversations between partners at runtime as a means of checking behavioral correctness of the entire web service system. This is related to the earlier work on conformance checking [6], [7], [8], [17], [18] mentioned before.

In [36], a web service mining framework is proposed that allows unexpected and interesting service compositions to automatically emerge in a bottom-up fashion. The authors propose some mining techniques aiming at the discovery of such service compositions.

An event-driven approach to validate the transactional behavior of service compositions was proposed in [37]. Verification is done at design-time or run-time using the event calculus formalism.

Within the ACSI project [27] the focus is on many-to-many relationships between instances. So-called "proclets" [23] are used to model artifact centric models. A conformance checking approach for such models is presented in [28], [29] and implemented in ProM.

In [38] the topic of "cross-organizational mining" was introduced. Here the goal is not to analyze interacting services but to compare services that are variants of one another. Cross-organizational mining can be used for benchmarking and reference modeling.

6 Conclusion

Process mining aims to bridge the gap between *Business Intelligence* (BI) and *Business Process Management* (BPM). BI techniques are typically not process-centric and provide rather simplistic reporting and dashboard functionalities. Sometimes BI tools offer more advanced data mining capabilities. However, classical data mining techniques [21] such as classification, clustering, regression, association rule learning, and sequence/episode mining are not process-centric at all. BPM techniques on the other hand gravitate around process models [39]. These models are typically made by hand and are used for analysis (e.g., simulation and verification) and enactment by BPM systems. However, mainstream BPM approaches are not using event data. Yet, activities executed by people, services, machines, and software components leave trails in so-called *event logs*. Process mining techniques use such logs to discover, analyze, and improve business processes.

In this paper, we discussed process mining in the context of services. We used the guiding principles and challenges described in the recently released Process Mining Manifesto [13] to describe the state-of-the-art in process mining. Given the abundance of event logs in the context of web services, we advocate the use of such techniques for what we call *service mining*. Service mining is concerned with (a) the discovery of service behavior, (b) checking conformance of services, and (c) service model extension (e.g., showing bottlenecks based on event data). We also discussed two challenges specific for service mining: (a) "How to Correlate Instances?" and (b) "How to Analyze Services Out of Context?". These challenges show that, despite the huge potential of service mining, additional research is needed to improve the applicability of process mining in distributed and loosely coupled systems.

Acknowledgments

The author would like to thank all that contributed to the Process Mining Manifesto: Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malu Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul , Frank van Geffen, Sukriti Goel, Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maggi, Donato Malerba, Ronny Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid Motahari Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Wynn.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services Concepts, Architectures and Applications*. Springer-Verlag, Berlin, 2004.

[2] L. Zhang, J. Zhang, and H. Cai, *Services Computing, Core Enabling Technology of the Modern Services Industry*. Springer-Verlag, Berlin, 2007.

[3] W. van der Aalst, "Don't go with the flow: Web services composition standards exposed," *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 72–76, 2003.

[4] F. Casati, E. Shan, U. Dayal, and M. Shan, "Business-oriented management of Web services," *Communications of the ACM*, vol. 46, no. 10, pp. 55–60, 2003.

[5] B. Benatallah, F. Casati, and F. Toumani, "Representing, Analysing and Managing Web Service Protocols," *Data and Knowledge Engineering*, vol. 58, no. 3, pp. 327–357, 2006.

[6] W. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H. Verbeek, "Conformance Checking of Service Behavior," *ACM Transactions on Internet Technology*, vol. 8, no. 3, pp. 29–59, 2008.

[7] W. van der Aalst and M. Pesic, "Chapter 2: Specifying and Monitoring Service Flows: Making Web Services Process-Aware," in *Test and Analysis of Web Services*, L. Baresi and E. Nitto, Eds. Springer-Verlag, Berlin, 2007, pp. 11–56.

[8] ——, "DecSerFlow: Towards a Truly Declarative Service Flow Language," in *International Conference on Web Services and Formal Methods (WS-FM 2006)*, ser. Lecture Notes in Computer Science, M. Bravetti, M. Nunez, and G. Zavattaro, Eds., vol. 4184. Springer-Verlag, Berlin, 2006, pp. 1–23.

[9] W. van der Aalst and H. Verbeek, "Process Mining in Web Services: The WebSphere Case," *IEEE Bulletin of the Technical Committee on Data Engineering*, vol. 31, no. 3, pp. 45–48, 2008.

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

[11] 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*, vol. 47, no. 2, pp. 237–267, 2003.

[12] W. van der Aalst, H. Reijers, A. Weijters, B. van Dongen, A. Medeiros, M. Song, and H. Verbeek, "Business Process Mining: An Industrial Application," *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

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

[14] H. Chesbrough and J. Spohrer, "A Research Manifesto for Services Science," *Communications of the ACM*, vol. 49, no. 7, pp. 35–40, 2006.

[15] W. van der Aalst, A. Mooij, C. Stahl, and K. Wolf, "Service Interaction: Patterns, Formalization, and Analysis," in *Formal Methods for Web Services*, ser. Lecture Notes in Computer Science, M. Bernardo, L. Padovani, and G. Zavattaro, Eds., vol. 5569. Springer-Verlag, Berlin, 2009, pp. 42–88.

[16] W. van der Aalst, A. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[17] A. Adriansyah, B. van Dongen, and W. van der Aalst, "Conformance Checking using Cost-Based Fitness Analysis," in *IEEE International Enterprise Computing Conference (EDOC 2011)*. IEEE Computer Society, 2011.

[18] A. Rozinat and W. van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[19] W. van der Aalst, M. Schonenberg, and M. Song, "Time Prediction Based on Process Mining," *Information Systems*, vol. 36, no. 2, pp. 450–475, 2011.

[20] M. Song and W. van der Aalst, "Towards Comprehensive Support for Organizational Mining," *Decision Support Systems*, vol. 46, no. 1, pp. 300–317, 2008.

[21] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. MIT press, Cambridge, MA, 2001.

[22] R. Bose, W. van der Aalst, I. Zliobaite, and M. Pechenizkiy, "Handling Concept Drift in Process Mining," in *International Conference on Advanced Information Systems Engineering (Caise 2011)*, ser. Lecture Notes in Computer Science, H. Mouratidis and C. Rolland, Eds., vol. 6741. Springer-Verlag, Berlin, 2011, pp. 391–405.

[23] W. van der Aalst, P. Barthelmess, C. Ellis, and J. Wainer, "Proclets: A Framework for Lightweight Interacting Workflow Processes," *International Journal of Cooperative Information Systems*, vol. 10, no. 4, pp. 443–482, 2001.

[24] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, "Towards Formal Analysis of Artifact-Centric Business Process Models," in *International Conference on Business Process Management (BPM 2007)*, ser. Lecture Notes in Computer Science, G. Alonso, P. Dadam, and M. Rosemann, Eds., vol. 4714. Springer-Verlag, Berlin, 2007, pp. 288–304.

[25] W. Pauw, M. Lei, E. Pring, L. Villard, M. Arnold, and J. Morar, "Web Services Navigator: Visualizing the Execution of Web Services," *IBM Systems Journal*, vol. 44, no. 4, pp. 821–845, 2005.

[26] H. Montahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah, "Event Correlation for Process Discovery from Web Service Interaction Logs," *VLBD Journal*, vol. 20, no. 3, pp. 417–444, 2011.

[27] ACSI, "Artifact-Centric Service Interoperation (ACSI) Project Home Page," www.acsi-project.eu.

[28] D. Fahland, M. de Leoni, B. van Dongen, and W. van der Aalst, "Conformance Checking of Interacting Processes with Overlapping Instances," in *Business Process Management (BPM 2011)*, ser. Lecture Notes in Computer Science, S. Rinderle, F. Toumani, and K. Wolf, Eds., vol. 6896. Springer-Verlag, Berlin, 2011, pp. 345–361.

[29] D. Fahland, M. Leoni, B. van Dongen, and W. van der Aalst, "Behavioral Conformance of Artifact-Centric Process Models," in *Business Information Systems (BIS 2011)*, ser. Lecture Notes in Business Information Processing, A. Abramowicz, Ed., vol. 87. Springer-Verlag, Berlin, 2011, pp. 37–49.

[30] A. Barros, G. Decker, M. Dumas, and F. Weber, "Correlation Patterns in Service-Oriented Architectures," in *Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering (FASE 2007)*, ser. Lecture Notes in Computer Science, M. Dwyer and A. Lopes, Eds., vol. 4422. Springer-Verlag, Berlin, 2007, pp. 245–259.

[31] S. Dustdar, R. Gombotz, and K. Baina, "Web Services Interaction Mining," Technical Report TUV-1841-2004-16, Information Systems Institute, Vienna University of Technology, Wien, Austria, 2004.

[32] S. Dustdar and R. Gombotz, "Discovering Web Service Workflows Using Web Services Interaction Mining," *International Journal of Business Process Integration and Management*, vol. 1, no. 4, pp. 256–266, 2006.

[33] R. Gombotz and S. Dustdar, "On Web Services Mining," in *BPM 2005 Workshops (Workshop on Business Process Intelligence)*, ser. Lecture Notes in Computer Science, C. Bussler et al., Ed., vol. 3812. Springer-Verlag, Berlin, 2005, pp. 216–228.

[34] H. Nezhad, R. Saint-Paul, B. Benatallah, and F. Casati, "Deriving Protocol Models from Imperfect Service Conversation Logs," *IEEE Transaction on Knowledge and Data Engineering*, vol. 20, no. 12, pp. 1683–1698, 2008.

[35] J. Simmonds, Y. Gan, M. Chechik, S. Nejati, B. Farrell, E. Litani, and J. Waterhouse, "Runtime Monitoring of Web Service Conversations," *IEEE Transactions on Services Computing*, vol. 2, no. 3, pp. 223–244, 2009.

[36] G. Zheng and A. Bouguettaya, "Service Mining on the Web," *IEEE Transactions on Services Computing*, vol. 2, no. 1, pp. 65–78, 2009.

[37] W. Gaaloul, S. Bhiri, and M.Rouached, "Event-Based Design and Runtime Verification of Composite Service Transactional Behavior," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 32–45, 2010.

[38] W. van der Aalst, "Configurable Services in the Cloud: Supporting Variability While Enabling Cross-Organizational Process Mining," in *OTM Federated Conferences, 18th International Conference on Cooperative Information Systems (CoopIS 2010)*, ser. Lecture Notes in Computer Science, R.Meersman, T. Dillon, and P. Herrero, Eds., vol. 6426. Springer-Verlag, Berlin, 2010, pp. 8–25.

[39] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin, 2007.

Wil van der Aalst Prof.dr.ir. Wil van der Aalst is a full professor of Information Systems at the Technische Universiteit Eindhoven (TU/e). Currently he is also an adjunct professor at Queensland University of Technology (QUT) working within the BPM group there. His research interests include workflow management, process mining, Petri nets, business process management, process modeling, and process analysis. He is also editor/member of the editorial board of several journals, including the Distributed and Parallel Databases, the International Journal of Business Process Integration and Management, the International Journal on Enterprise Modelling and Information Systems Architectures, Computers in Industry, Business & Information Systems Engineering, IEEE Transactions on Services Computing, Lecture Notes in Business Information Processing, and Transactions on Petri Nets and Other Models of Concurrency. He is also a member of the Royal Holland Society of Sciences and Humanities (Koninklijke Hollandsche Maatschappij der Wetenschappen) and the Academy of Europe (Academia Europaea).