Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing

W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
Phone: +31 40 247.4295/2290. Fax: +31 40 243.2612.
w.m.p.v.d.aalst@tm.tue.nl

Abstract. Increasingly, business processes are being controlled and/or monitored by information systems. As a result, many business processes leave their "footprints" in transactional information systems, i.e., business events are recorded in so-called event logs. Process mining aims at improving this by providing techniques and tools for discovering process, control, data, organizational, and social structures from event logs, i.e., the basic idea of process mining is to diagnose business processes by mining event logs for knowledge. In this paper we focus on the potential use of process mining for measuring *business alignment*, i.e., comparing the real behavior of an information system or its users with the intended or expected behavior. We identify two ways to create and/or maintain the fit between business processes and supporting information systems: *Delta analysis* and *conformance testing*. Delta analysis compares the discovered model (i.e., an abstraction derived from the actual process) with some predefined processes model (e.g., the workflow model or reference model used to configure the system). Conformance testing attempts to quantify the "fit" between the event log and some predefined processes model. In this paper, we show that Delta analysis and conformance testing can be used to analyze business alignment as long as the actual events are logged and users have some control over the process.

Key words: Process Mining, Business Process Management, Business Alignment, Workflow Management, Delta Analysis, Conformance Testing.

1 Introduction

In many organizations new processes are emerging and existing processes are changing ("The only constant is change"). Therefore, the *alignment of business processes and information systems* requires continuous attention. To "maintain the fit" it is important to detect changes over time, i.e., deviations of the described or prescribed behavior. Rare deviations will always happen, but should not be regarded as a symptom of a change. However, if considerable amount of process instances deviate from the prescribed pattern, some action should be

undertaken to align the business process and the supporting information system. Similarly, to "create the fit" it may be worthwhile to monitor the actual behavior of the people in the organization before configuring/installing the (new) information system. Both for creating and maintaining the fit (i.e., securing alignment) we propose *process mining* techniques which use event logs to discover the actual process.

Today, many enterprise information systems store relevant events in some structured form. For example, workflow management systems typically register the start and completion of activities. ERP systems like SAP log all transactions, e.g., users filling out forms, changing documents, etc. Business-to-business (B2B) systems log the exchange of messages with other parties. Call center packages but also general-purpose CRM systems log interactions with customers. These examples show that many systems have some kind of *event log* often referred to as "history", "audit trail", "transaction log", etc. [1, 2]. The event log typically contains information about events referring to an *activity* and a *case*. The case (also named process instance) is the "thing" which is being handled, e.g., a customer order, a job application, an insurance claim, a building permit, etc. The activity (also named task, operation, action, or work-item) is some operation on the case. Typically, events have a *timestamp* indicating the time of occurrence. Moreover, when people are involved, event logs will typically contain information on the person executing or initiating the event, i.e., the *originator*. Based on this information several tools and techniques for process mining have been developed.

Process mining is useful for at least two reasons. First of all, it could be used as a tool to find out how people and/or procedures really work. Consider for example processes supported by an ERP system like SAP (e.g., a procurement process). Such a system logs all transactions but in many cases does not enforce a specific way of working. In such an environment, process mining could be used to gain insight in the actual process. Another example would be the flow of patients in a hospital. Note that in such an environment all activities are logged but information about the underlying process is typically missing. In this context it is important to stress that management information systems provide information about key performance indicators like resource utilization, flow times, and service levels but *not* about the underlying business processes (e.g., causal relations, ordering of activities, etc.). Second, process mining could be used for *Delta analysis*, i.e., comparing the actual process with some predefined process. Note that in many situations there is a descriptive or prescriptive process model. Such a model specifies how people and organizations are assumed/expected to work. By comparing the descriptive or prescriptive process model with the discovered model, discrepancies between both can be detected and used to improve the process. Consider for example the so-called reference models in the context of SAP. These models describe how the system should be used. Using process mining it is possible to verify whether this is the case. In fact, process mining could also be used to compare different departments/organizations using the same ERP system. Instead of doing a Delta analysis it is also possible to do *conformance testing*. The basic idea of conformance testing is to directly compare the log with

the descriptive or prescriptive process model, i.e., do the log and the model "fit" together.

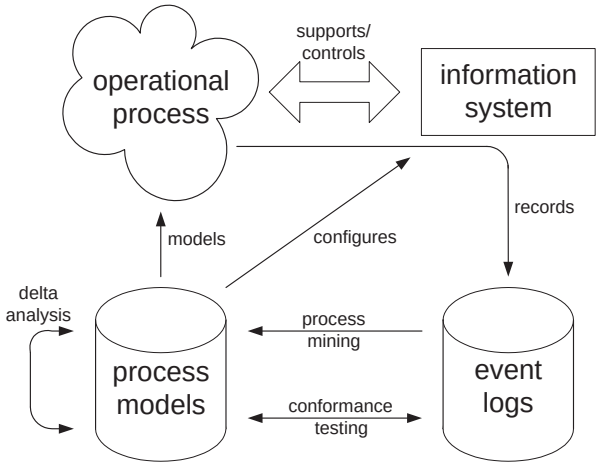


Fig. 1. Tackling business alignment (i.e., the "fit" between the operational process and the information system) using Delta analysis and conformance testing.

Figure 1 illustrates the way process mining, delta analysis, and conformance testing can be used to analyze business alignment. Clearly, the operational process and the information system interact, i.e., the information system is used to support or even control the operational process. The information system is often configured on the basis of a process model as Figure 1 shows. This model is often some abstraction of the operational process. In fact it is interesting to note that similar models can be used to analyze the operational process and to configure the information system. People using workflow management systems and simulation tools often note the many similarities between the workflow specification and the simulation model. As indicated, many information systems log events referring to actions in the operational process. These logs can be used for process mining. As Figure 1 shows, process mining derives a process model from an event log. The resulting process model can be compared with the original model (Delta analysis). It is however also possible to compare the event log directly with the original model (conformance testing).

This paper is a spin-off of the BPMDS 2004 workshop (cf. [3, 4]). Here the focus was on business alignment and not necessarily on requirements engineering. For the readership of the *Requirements Engineering* journal it is interesting to put process mining, delta analysis, and conformance testing in the context of the software development process. There are two possible ways these techniques can be applied in this context. First of all, the software development process is also a process and in some cases the activities in this process are also logged. There are similarities to for example the application of process mining in the context

of Product Data Management (PDM) systems. Here the objects are design documents (e.g., a technical drawing) that can be indifferent states. Artifacts in the software development process (e.g., designs and executable code) also go through different states. A version control system such as Concurrent Versions System (CVS) can be used for process mining and the actual processes can be compared with a normative process model. Second, the aim of requirement engineering is to produce software that "fits". For example, co-development, i.e., concurrently designing business processes and business software applications, aims at aligning business processes and software. Here process mining, delta analysis, and conformance testing can be used to compare designs (e.g., UML models) with real-life scenarios. Every audit trail can be seen as a scenario and using the techniques presented in this paper a set of scenarios (e.g., event log) can be compared with artifacts in the software development process at different levels of granularity (from high-level designs to executable code).

The remainder of this paper is organized as follows. Section 2 introduces the concept of business process mining followed by a short description of a case study in Section 3. Section 4 discusses process mining as a tool for Delta analysis, i.e., measuring the business alignment by comparing event logs with descriptive or prescriptive (process) models. Section 5 discusses metrics for conformance testing, i.e., quantifying the fit. Finally, Section 6 discusses related work and Section 7 concludes the paper.

2 Business Process Mining: An overview

The goal of process mining is to extract information about processes from transaction logs [1]. We assume that it is possible to record events such that (i) each event refers to an *activity* (i.e., a well-defined step in the process), (ii) each event refers to a *case* (i.e., a process instance), (iii) each event can have a *performer* also referred to as *originator* (the person executing or initiating the activity), and (iv) events have a *timestamp* and are totally ordered.[1] Table 1 shows an example of a log involving 19 events, 5 activities, and 6 originators. In addition to the information shown in this table, some event logs contain more information on the case itself, i.e., data elements referring to properties of the case. For example, the case handling systems FLOWer logs every modification of some data element.

Event logs such as the one shown in Table 1 are used as the starting point for mining. We distinguish three different perspectives: (1) the process perspective, (2) the organizational perspective and (3) the case perspective. The *process perspective* focuses on the control-flow, i.e., the ordering of activities. The goal of

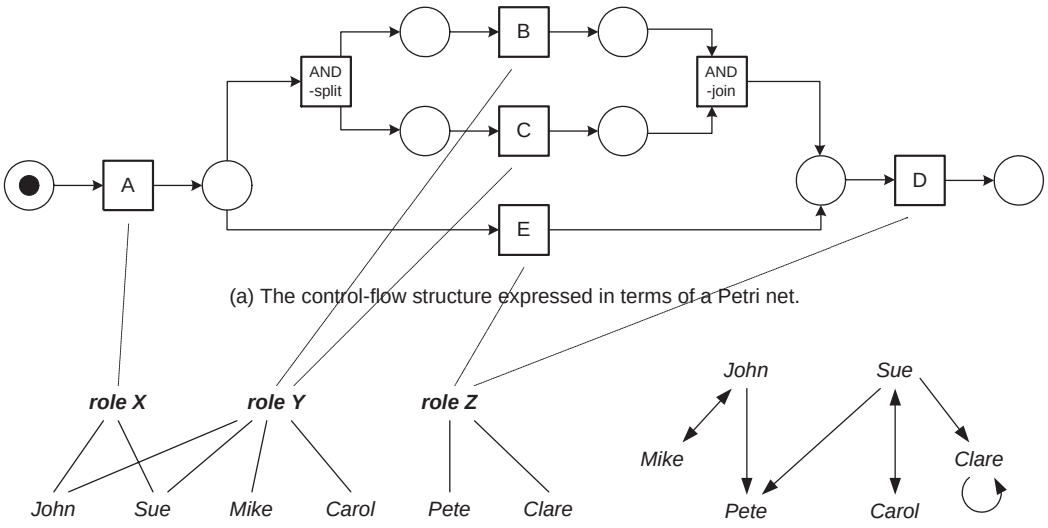
[1] Note that in Table 1 we abstract from *event types*, i.e., we consider activities to be atomic. In real logs events typically correspond to the start or completion of an activity. This way it is possible to measure the duration of activity and to explicitly detect parallelism. Moreover, there are other event types related to failures, scheduling, delegations, etc. For simplicity we abstract from this in this paper. However, in our process mining tools we take event types into account.

case id	activity id	originator	timestamp
case 1	activity A	John	9-3-2004:15.01
case 2	activity A	John	9-3-2004:15.12
case 3	activity A	Sue	9-3-2004:16.03
case 3	activity B	Carol	9-3-2004:16.07
case 1	activity B	Mike	9-3-2004:18.25
case 1	activity C	John	10-3-2004:9.23
case 2	activity C	Mike	10-3-2004:10.34
case 4	activity A	Sue	10-3-2004:10.35
case 2	activity B	John	10-3-2004:12.34
case 2	activity D	Pete	10-3-2004:12.50
case 5	activity A	Sue	10-3-2004:13.05
case 4	activity C	Carol	11-3-2004:10.12
case 1	activity D	Pete	11-3-2004:10.14
case 3	activity C	Sue	11-3-2004:10.44
case 3	activity D	Pete	11-3-2004:11.03
case 4	activity B	Sue	11-3-2004:11.18
case 5	activity E	Clare	11-3-2004:12.22
case 5	activity D	Clare	11-3-2004:14.34
case 4	activity D	Pete	11-3-2004:15.56

Table 1. An event log.

mining this perspective is to find a good characterization of all possible paths, e.g., expressed in terms of a Petri net or Event-driven Process Chain (EPC). The *organizational perspective* focuses on the originator field, i.e., which performers 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 relation between individual performers (i.e., build a social network). The *case perspective* focuses on properties of cases. Cases can be characterized by their path in the process or by the originators working on a case. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represent a replenishment order it is interesting to know the supplier or the number of products ordered.

The process perspective is concerned with the "How?" question, the organizational perspective is concerned with the "Who?" question, and the case perspective is concerned with the "What?" question. To illustrate the first two consider Figure 2. The log shown in Table 1 contains information about five cases (i.e., process instances). The log shows that for four cases (1, 2, 3, and 4) the activities A, B, C, and D have been executed. For the fifth case only three activities are executed: activities A, E, and D. Each case starts with the execution of A and ends with the execution of D. If activity B is executed, then also activity C is executed. However, for some cases activity C is executed before activity B. Based on the information shown in Table 1 and by making some assumptions about the completeness of the log (i.e., assuming that the cases are representative and a sufficient large subset of possible behaviors is observed), we



(a) The control-flow structure expressed in terms of a Petri net.

(b) The organizational structure expressed in terms of an activity-role-performer diagram.

(c) A sociogram based on transfer of work.

Fig. 2. Some mining results for the process perspective (a) and organizational (b and c) perspective based on the event log shown in Table 1.

can deduce the process model shown in Figure 2(a). The model is represented in terms of a Petri net [5]. (In fact it is a workflow net, see also Section 4 for more details.) The Petri net starts with activity A and finishes with activity D. These activities are represented by transitions. After executing A there is a choice between either executing B and C in parallel or just executing activity E. To execute B and C in parallel two non-observable activities (AND-split and AND-join) have been added. These activities have been added for routing purposes only and are not present in the event log. Note that for this example we assume that two activities are in parallel if they appear in any order. By distinguishing between start events and complete events for activities it is possible to explicitly detect parallelism.

Figure 2(a) does not show any information about the organization, i.e., it does not use any information on the people executing activities. However, Table 1 shows information about the performers. For example, we can deduce that activity A is executed by either John or Sue, activity B is executed by John, Sue, Mike or Carol, C is executed by John, Sue, Mike or Carol, D is executed by Pete or Clare, and E is executed by Clare. We could indicate this information in Figure 2(a). The information could also be used to "guess" or "discover" organizational structures. For example, a guess could be that there are three roles: X, Y, and Z. For the execution of A role X is required and John and Sue have this role. For the execution of B and C role Y is required and John, Sue, Mike and Carol have this role. For the execution of D and E role Z is required and Pete and Clare have this role. For five cases these choices may seem arbitrary but for larger data sets such inferences capture the dominant roles in an organization. The resulting "activity-role-performer diagram" is shown in Figure 2(b). The three "discovered" roles link activities to performers. Figure 2(c) shows another view on the organization based on the transfer of work from one individual to another, i.e., not focus on the relation between the process and individuals but on relations among individuals (or groups of individuals). Consider for example Table 1. Although Carol and Mike can execute the same activities (B and C), Mike is always working with John (cases 1 and 2) and Carol is always working with Sue (cases 3 and 4). Probably Carol and Mike have the same role but based on the small sample shown in Table 1 it seems that John is not working with Carol and Sue is not working with Carol. These examples show that the event log can be used to derive relations between performers of activities, thus resulting in a sociogram. For example, it is possible to generate a sociogram based on the transfers of work from one individual to another as is shown in Figure 2(c). Each node represents one of the six performers and each arc represents that there has been a transfer of work from one individual to another. The definition of "transfer of work from A to B" is based on whether there for the same case an activity executed by A is directly followed by an activity executed by B. For example, both in case 1 and 2 there is a transfer from John to Mike. Figure 2(c) does not show frequencies. However, for analysis proposes these frequencies can added. The arc from John to Mike would then have weight 2. Typically, we do not use absolute frequencies but weighted frequencies to get relative values be-

tween 0 and 1. Figure 2(c) shows that work is transferred to Pete but not vice versa. Mike only interacts with John and Carol only interacts with Sue. Clare is the only person transferring work to herself.

Besides the "How?" and "Who?" question (i.e., the process and organization perspectives), there is the case perspective that is concerned with the "What?" question. Figure 2 does not address this. In fact, focusing on the case perspective is most interesting when also data elements are logged but these are not listed in Table 1. The case perspective looks at the case as a whole and tries to establish relations between the various properties of a case. Note that some of the properties may refer to the activities being executed, the performers working on the case, and the values of various data elements linked to the case. Using clustering algorithms it would for example be possible to show a positive correlation between the the size of an order or its handling time and the involvement of specific people.

Orthogonal to the three perspectives (process, organization, and case), the result of a mining effort may refer to *logical* issues and/or *performance* issues. For example, process mining can focus on the logical structure of the process model (e.g., the Petri net shown in Figure 2(a)) or on performance issues such as flow time. For mining the organizational perspectives, the emphasis can be on the roles or the social network (cf. Figure 2(b) and (c)) or on the utilization of performers or execution frequencies.

To address the three perspectives and the logical and performance issues we have developed a set of tools including EMiT, Thumb , and MinSoN sharing a common XML format. Recently, these tools have been merged into the *ProM framework* (see http://www.processmining.org for more details). Figure 3 shows a screenshot of the ProM tool while analyzing the event log shown in Table 1. Note that indeed the results shown in Figure 2 are obtained.

3 Case study

We have applied our mining techniques in several organizations. In this section, we briefly show some results for one of these organizations, i.e., the processes of a Dutch governmental institution responsible for fine-collection.[2] A case (process instance) is a fine that has to be paid. There may be more fines related with the same person. However, each fine corresponds to an independent case. This process has the particularity that as soon as the fine is paid, the process stops. In total there are 99 distinct activities which can be either manually or automatically executed. We selected the fines information for 130136 cases. We constructed the process log and we applied to this log our process discovery method that can handle noisy data [6, 7].

Figure 4 (top-left) shows a fragment of the log containing 130136 cases. This log is generated by an information system specifically constructed for the Dutch

[2] The name of the organization is not given for reasons of confidentiality. We want to thank L. Maruster, R. Dorenbos, H.J. de Vries, H. Reijers, and A. in 't Veld for their valuable support.

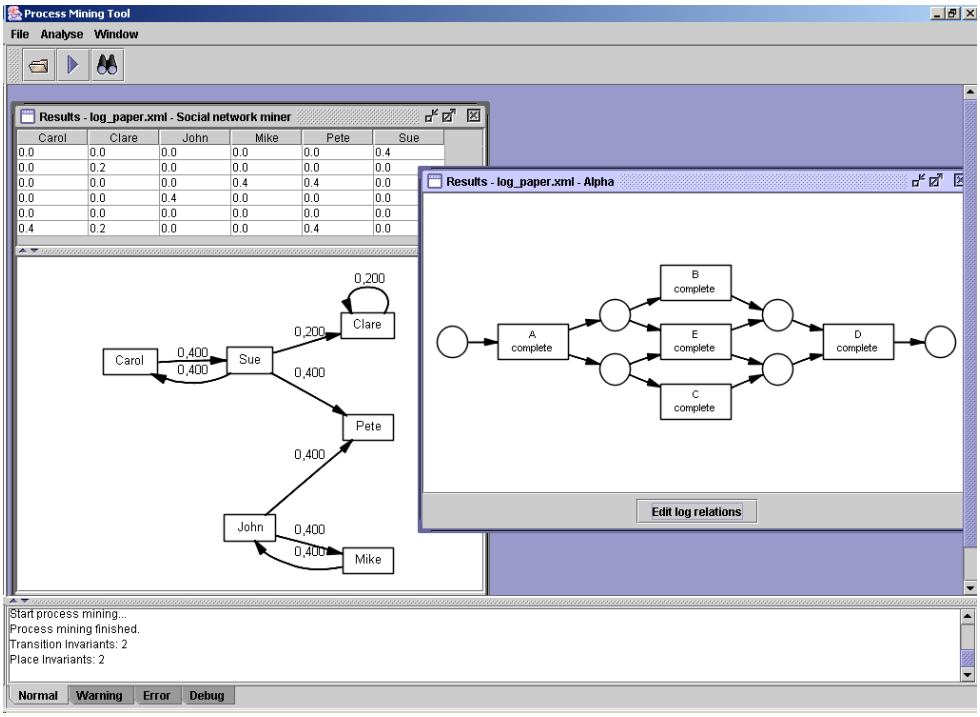


Fig. 3. Some mining results obtained using ProM based on the event log shown in Table 1.

governmental institution. (The institution has been in the process of using standard workflow technology but this process has been put "on hold".) The top-right screen shows a screenshot of our mining tool EMiT while analyzing the log. The bottom screenshot shows the whole process obtained through application of the process mining techniques. The discovered models have been inspected by the domain experts. They concluded that our discovered models were able to grasp the important aspects of the process. Moreover, the discovered models revealed aspects that are often questioned when discussing the process model. These experiences showed that process discovery can provide useful insights into the current practice of a process and highlight difference between the actual process and the prescriptive/descriptive model [7].

We have also applied our process mining techniques to a health-care process where the flow of multi-disciplinary patients is analyzed. We have analyzed event logs (visits to different specialists) of patients with peripheral arterial vascular diseases of the Elizabeth Hospital in Tilburg and the Academic Hospital in Maastricht. Patients with peripheral arterial vascular diseases are a typical example of multi-disciplinary patients. We have preliminary results showing that process mining is very difficult given the "spaghetti-like" nature of this process. Only by focusing on specific tasks and abstracting from infrequent tasks we are

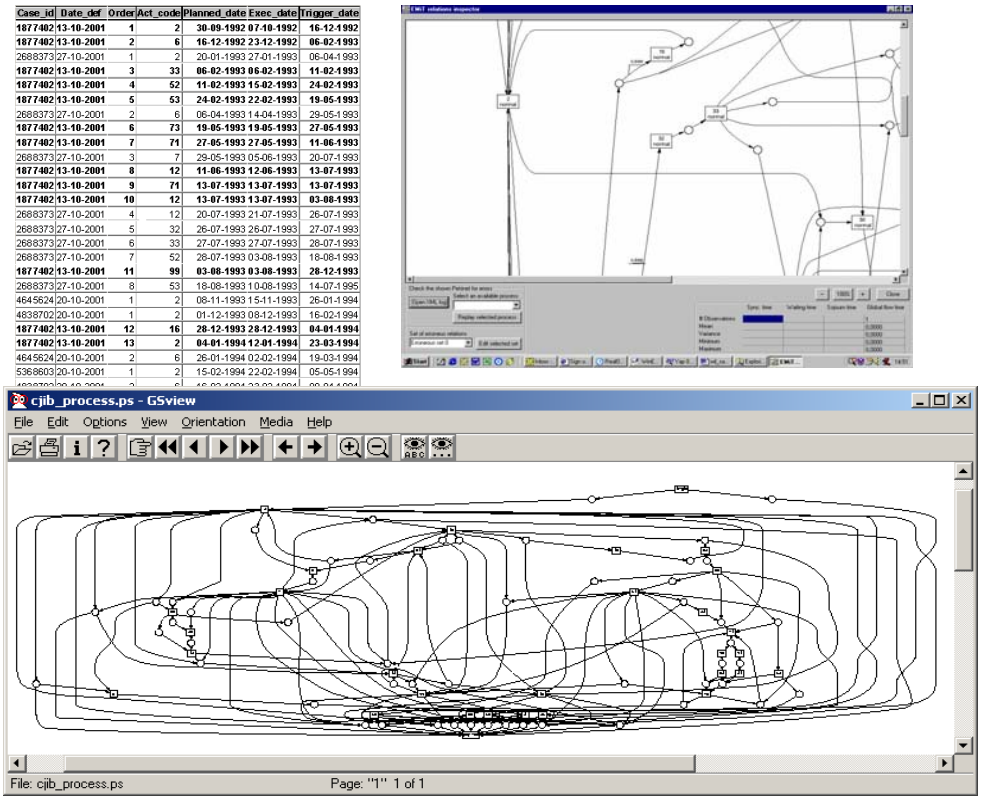


Fig. 4. A fragment of the log of a Dutch governmental institution responsible for fine-collection and the corresponding process mining result.

able to successfully mine such processes. Given this experience we are now focusing on processes have more structure. For example, environments using case handling system like FLOWer (the workflow product of Pallas Athena), e.g., the Employee Insurance Implementing Body (Uitvoering Werknemersverzekeringen, or UVW).

The case studies show that in many situations there is a discrepancy between the predefined process (i.e., a descriptive or prescriptive process model) and the real process (i.e., the process model obtained through mining). From the viewpoint of business alignment these discrepancies are of particular interest since they may indicate a misfit between the information system (based on an unrealistic or incorrect descriptive or prescriptive process model) and the real business process. In the remainder we explore two techniques to analyze or measure discrepancies between some predefined process and the real process: (1) *Delta analysis* and (2) *Conformance testing*.

4 Delta analysis

Process mining can be used for *Delta analysis*, i.e., comparing the actual process, represented by a process model obtained through process mining, with some predefined process representing the information system. Note that in many situations there is a *descriptive* or *prescriptive* process model. Such a model specifies how people and organizations are assumed/expected to work. By comparing the descriptive or prescriptive process model with the discovered model, discrepancies between both can be detected and used to improve the process. Figure 5 shows the basic idea of Delta analysis: two process models are compared, i.e., differences between the discovered process model and the descriptive/prescriptive process model are analyzed.

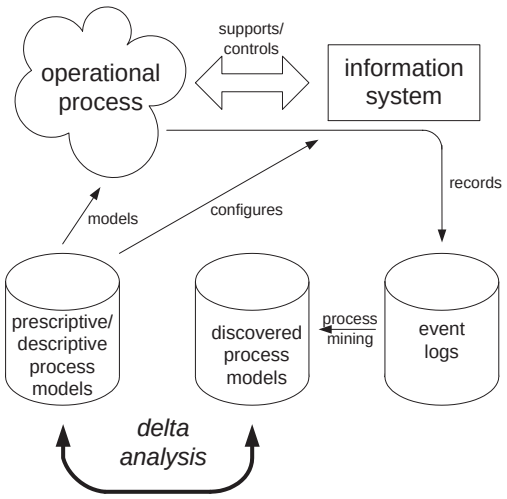


Fig. 5. Delta analysis as a means to analyze business alignment.

Before discussing techniques for Delta analysis, we first illustrate the existence of descriptive/prescriptive process model in real-life situations. Increasingly, health-care workers are using medical protocols, also named medical guidelines (to emphasize support) or pathways (to emphasize prediction), are used to treat patients [8, 9]. This is reflected by the abundance of languages for describing medical protocols, e.g., Asbru, EON, GLIF, GUIDE, PRODIGY and PROforma [10–13]. These languages are comparable to typical process modeling languages in the business domain. There are two ways medical protocols are used: (1) actively and (2) passively. If medical protocols are used actively, the medical information systems is either trying to control the careflow or to automatically suggest pathways. If medical protocols are used only passively, the health-care worker can consult them (i.e., the information system is not taking the initia-

tive). Both the active and passive use of medical protocols entails process models (e.g., flowcharts) that are of a prescriptive nature. Clearly, health-care worker can (and should!) deviate from the protocol when needed. Nevertheless, it is interesting to know how well the protocol "fits". Workflow models in workflow management systems are also prescriptive and even enforce parts of the process to be executed in a certain way. Note that even in processes "controlled" by a workflow system, workers typically still have quite some operational freedom, i.e., the system cannot force workers to execute tasks. For example, the order of execution and the exact work distribution are decided by the users. The current trend in workflow management is to develop systems that allow for even more flexibility (cf. case handling systems such as FLOWer [14]). This suggests that it is increasingly more interesting to compare predefined process models with "discovered" models. In this context it is also interesting to consider the so-called reference models of SAP [15]. These reference models can be used both in a descriptive and prescriptive manner. In the context of SAP reference models are typically expressed in terms of Event-driven Process Chains (EPC). Such an EPC can be used to describe how the SAP system should be used. However, in practice people may use the system differently or only use a subset of the system. Consider for example the two EPCs shown in Figure 6. The left EPC represents the full process, the right EPC shows the parts that are actually being used. Note that tools like the SAP Reverse Business Engineer (RBE) can be used to monitor how frequent parts of the SAP system are being used. Unfortunately, tools like RBE do not consider the order of activities nor other aspects such as the organizational and case perspective.

Figure 6 illustrates the goal of Delta analysis: using process mining techniques we want to compare the discovered model (i.e., the real process) with some predefined process. For clarification, we also show a smaller example expressed in terms of a Petri net. Figure 7(a) shows an example of an order handling process modeled in terms of a so-called workflow net [17]. Workflow nets form a subclass of the classical Petri-nets. The squares are the active parts of the model and correspond to tasks. The circles are the passive parts of the model and are used to represent states. In the classical Petri net, the squares are named transitions and the circles places. A workflow net models the life-cycle of one case. Examples of cases are insurance claims, tax declarations, and traffic violations. Cases are represented by tokens and in this case the token in *start* corresponds to an order. Task *register* is a so-called AND-split and is enabled in the state shown. The arrow indicates that this task requires human intervention. If a person executes this task, the token is removed from place *start* and two tokens are produced: one for *c1* and one for *c2*. Then, in parallel, two tasks are enabled: *check_availability* and *send_bill*. Depending on the eagerness of the workers executing these two tasks either *check_available* or *send_bill* is executed first. Suppose *check_availability* is executed first. If the ordered goods are available, they can be shipped by executing task *ship_goods*. If they are not available, either a replenishment order is issued or not. Note that *check_availability* is an OR-split and produces one token for *c3*, *c4*, or *c5*. Suppose that not all ordered goods are available, but the appro-

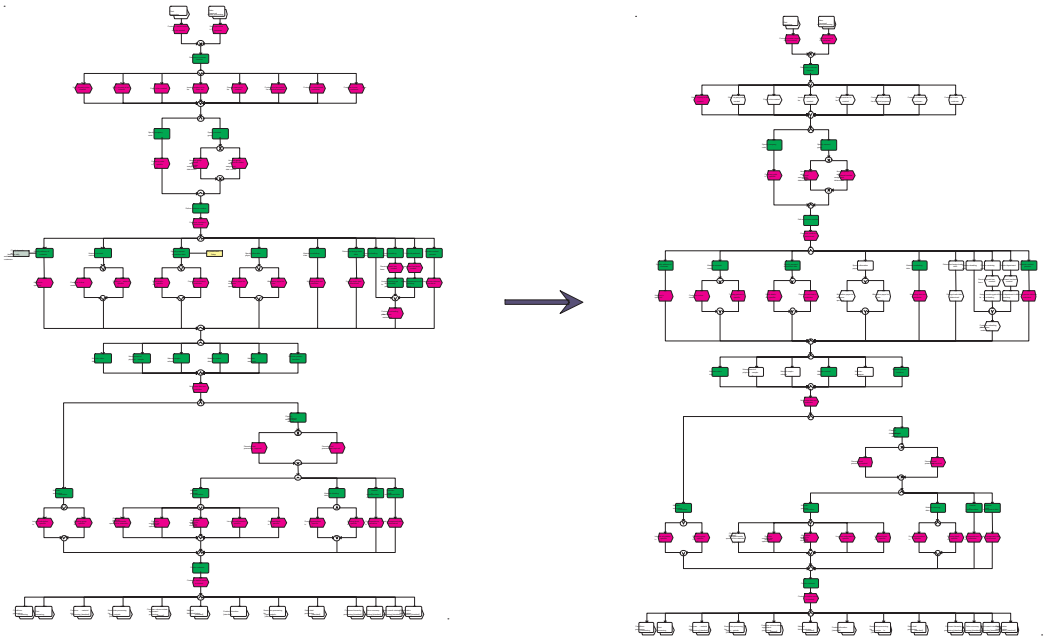
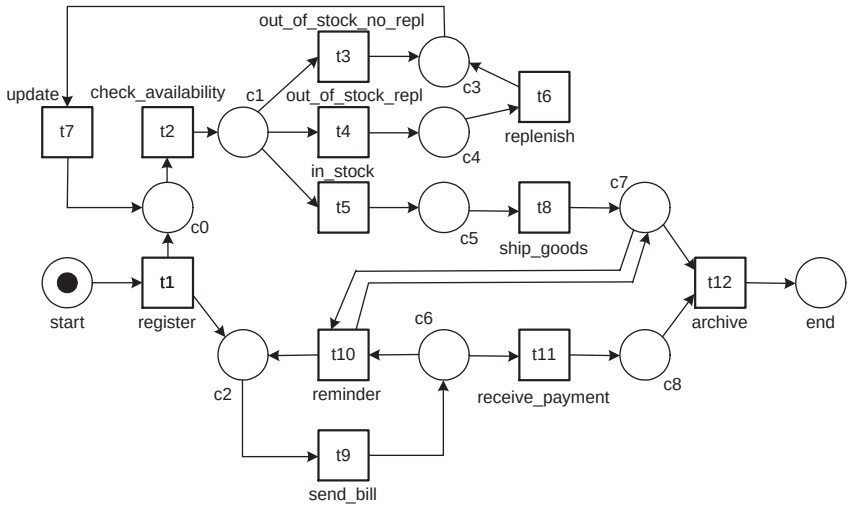
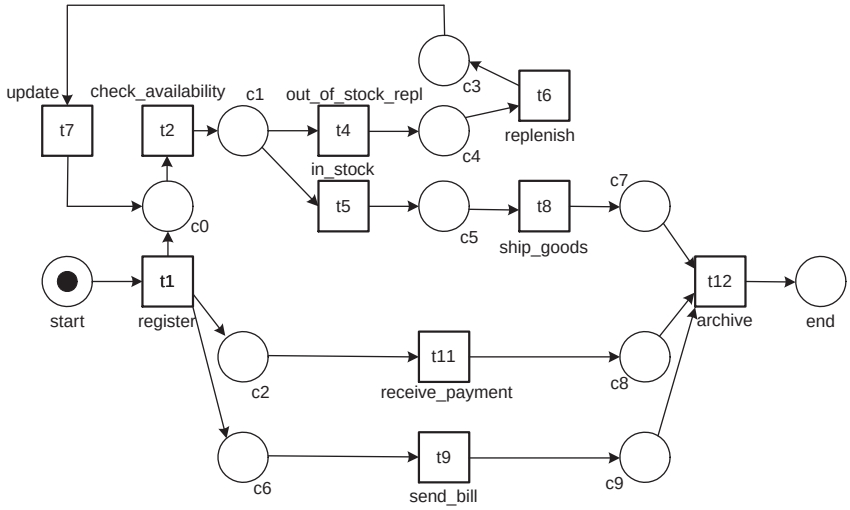


Fig. 6. Two EPCs: the full EPC (left) and EPC really being used (right) [16].

priate replenishment orders were already issued. A token is produced for *c3* and task *update* becomes enabled. Suppose that at this point in time task *send_bill* is executed, resulting in the state with a token in *c3* and *c6*. The token in *c6* is input for two tasks. However, only one of these tasks can be executed and in this state only *receive_payment* is enabled. Task *receive_payment* can be executed the moment the payment is received. Task *reminder* is an AND-join/AND-split and is blocked until the bill is sent and the goods have been shipped. Note that the reminder is sent only after a specified period. However, it is only possible to send a reminder if the goods have been actually shipped. Assume that in the state with a token in *c3* and *c6* task *update* is executed. This task does not require human involvement and is triggered by a message of the warehouse indicating that relevant goods have arrived. Again *check_availability* is enabled. Suppose that this task is executed and the result is positive. In the resulting state *ship_goods* can be executed. Now there is a token in *c6* and *c7* thus enabling task *reminder*. Executing task *reminder* again enables the task *send_bill*. A new copy of the bill is sent with the appropriate text. It is possible to send several reminders by alternating *reminder* and *send_bill*. However, let us assume that after the first loop the customer pays resulting in a state with a token in *c7* and *c8*. In this state, the AND-join *archive* is enabled and executing this task results in the final state with a token in *end*.



(a) Prescriptive/descriptive process model



(b) Discovered process model

Fig. 7. Comparing the discovered order processing model with the predefined process model.

Suppose that Figure 7(a) represents the process as it *is perceived* by management. To see whether this model fits reality, we can apply process mining on a log with events such as in Table 1. Using tools such as EMiT or ProM we can discover the workflow net shown in Figure 7(b). We do not list the complete log that would lead to this model. Instead we just show a fragment, cf. Table 2.

case id	activity id	originator	timestamp
case 1	register	Pete	14-3-2004:14.33
case 2	register	Carol	14-3-2004:15.22
case 3	register	Chris	14-3-2004:16.43
case 3	send bill	Michael	15-3-2004:16.17
case 1	receive payment	Jorge	15-3-2004:18.05
case 1	check availability	Pete	16-3-2004:9.43
case 2	send bill	Mike	16-3-2004:10.24
case 4	register	Sue	16-3-2004:10.35
...	...	...	...

Table 2. A fragment of the event log used to discover Figure 7(b).

When comparing Figure 7(a) and Figure 7(b) it is easy to see that there are four basic differences: (1) in the discovered process an out-of-stock situation always results in a replenishment order, (2) in the discovered process the sending of the bill and receiving the payment are in parallel (i.e., sometimes customers apparently pay before receiving the bill) , (3) in the discovered process it is not possible to send multiple bills, and (4) in the discovered process it is possible to send the bill before shipping to goods. Each of these differences provides interesting insights in discrepancies between the real process as it has been observed and the process model assumed by management. If the information system is configured based on Figure 7(a) and people really work as suggested by Figure 7(b), there is a misalignment. These problems can be addressed by either updating the prescriptive/descriptive model based on the discovered model or motivating people to stick to the original prescriptive/descriptive model.

For large processes it may be difficult to compare the prescriptive/descriptive model and the discovered model. There are many ways to highlight differences between two models in a graphical fashion. However, most of such approaches will consider simple "node mapping techniques" rather than compare differences in *behavior*, i.e., the focus is on syntactical differences rather than semantical differences. From a theoretical point of view there are at least two approaches that also incorporate behavior. The first approach is to use *inheritance of behavior* [18, 19]. The second approach is to calculate *change regions* [20, 21].

Inheritance is a well-know concept from object-orientation but typically not well-understood for processes. Based on the inheritance notions defined in [19] we have developed the notion of the *Greatest Common Divisor* (GCD) and *Least Common Multiple* (LCM) of two or more processes [18]. The inheritance relations

defined in [19] yield a partial order that can be used to reason about a common super- or subclass of the two models (i.e., predefined and discovered processes). The GCD is the common superclass which preserves as much information about the behavior of the Petri nets as possible, i.e., it is not possible to construct a more detailed Petri net which is also a superclass of both models. The GCD describes the behavior all variants agree on. The LCM is the most compact Petri net which is still a subclass of all variants. If we apply these notions to the two models shown in Figure 7 we can get the GCD and LCM of the predefined and discovered processes. For example, the GCD would only show the activities both processes agree on.

The change region is determined by comparing the two process models and extending the regions that have changed directly by the parts of the process that are also affected by the change of going from one process to the other, i.e., the syntactical affected parts of the processes are extended with the semantically affected parts of the processes to yield change regions. Due to the possibly complex mixture of different routing constructs (choice, synchronization, iteration, etc.), it is far from trivial to compute the actual change regions as is shown in [20, 21]. If we apply these notions to the two models shown in Figure 7, we can highlight the parts of the processes affected by the differences between the predefined and discovered models.

5 Conformance testing

In the previous section, we presented an approach (Delta analysis) that requires a predefined model and a discovered model (i.e., Delta analysis). This approach has two drawbacks. First of all, there may not be enough events to actually discover the process model.[3]

As a result, the discovered model may be flawed or not representative. Second, Delta analysis does not provide quantitative measures for the "fit" between the prescriptive/descriptive model and the log. Therefore, we propose an alternative approach in addition to Delta analysis: *conformance testing*.

Figure 8 shows the basic idea of conformance testing: the event log is directly compared with the predefined process model. This can easily be done. Compare for example case 1 in Table 2. This case starts with step register (*t1*), followed by receive payment (*t11*). This part of the case fits the discovered model Figure 7(b) but not the original model Figure 7(a) because *t11* is not enabled after firing *t1*. Given a complete case, represented as a sequence of activities (i.e., transitions names in Petri-net terms), it is possible to indicate whether a case fits or not. Consider for example Figure 2 and assume that the AND-split and AND-join are visible in the log. Assume these are recorded as S and J. The case AED fits. So

[3] An interesting question is now many events are required to correctly discover the process model. There is not a simple answer to that a-priori. However, there are simple techniques to assess the completeness of the result, e.g., K-fold cross validation which splits the event log into K parts and for each part it is verified whether adding it changes the result.

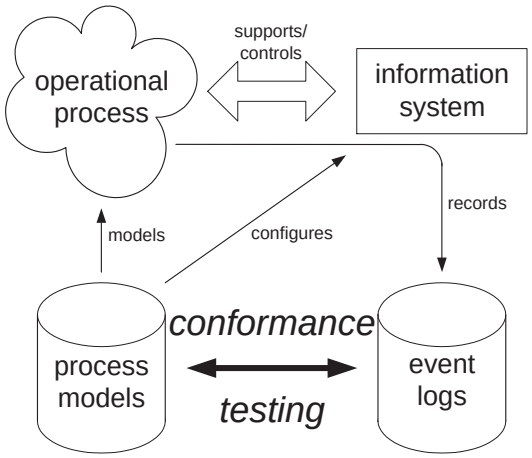
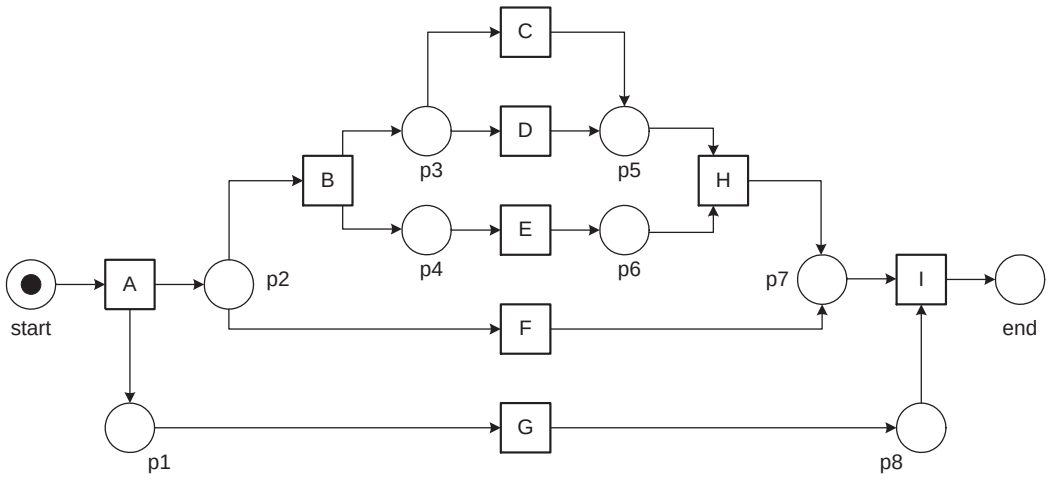


Fig. 8. Conformance testing as a means to quantify business alignment.

do the cases $ASBCJD$ and $ASCBJD$. However, the case ABD clearly does not fit. In other words the log consisting of only cases of the form AED, $ASBCJD$ and $ASCBJD$ *conforms* to the model while any log containing a case of the form ABD will not. In the remainder of this section we will further elaborate on the notion of conformance and introduce some metrics.

Consider Figure 9 where some predefined model and four logs are given. In the four logs we removed timestamps, performer, etc. and grouped the events in one sequence per case. Event log *L1* shows five cases. For the first case (*case 1*) four events are logged: A, G, F, and I. Clearly this sequence of events is possible in Figure 9(a), i.e., *case 1* seems to fit. The same holds for the other four cases in log *L1*. This can be checked by playing the "token game" in the Petri net shown Figure 9(a). Each case in log *L1* corresponds to a possible firing sequence of the Petri net with one initial token in the source place. Similarly, each case in event log *L2* corresponds to a possible firing sequence of Figure 9(a). However, it is striking to see that only events corresponding to the lower part of the Petri net appear in the log. Now consider event log *L3* shown Figure 9(d). This log does not seem to fit. For example, *case 1* does not correspond to a possible firing sequence because after executing A and F, place *p8* is still unmarked and I cannot be executed according to the model. It seems that an event involving G is missing. Also *case 4* in event log *L3* seems to suffer from this problem. Also event log *L4* shown Figure 9(e) does not fit completely, i.e., E seems to be impossible in first two cases of this log.

Figure 9 illustrates that there are two issues when checking the conformance of a predefined process model based on an actual event log. The first problem is *underfitting*, i.e., the log contains behavior not possible in the predefined model. Event logs *L3* and *case 4* suffer from underfitting because in each log two cases do not "fit". The second problem is *overfitting*, i.e., the log contains behavior



(a) Some descriptive/prescriptive process model.

case 1: AGFI case 2: AFGI case 3: AGBCEHI case 4: ABEDGHI case 5: ABDGEHI	case 1: AFGI case 2: AFGI case 3: AGFI case 4: AFGI case 5: AGFI	case 1: AFI case 2: AFGI case 3: AGBCEHI case 4: ABEDHI case 5: ABDGEHI	case 1: AGEFI case 2: AFEGI case 3: AGBCEHI case 4: ABEDGHI case 5: ABDGEHI
(b) Event log L1.	(c) Event log L2.	(d) Event log L3.	(e) Event log L4.

Fig. 9. A predefined process model and some event logs used to illustrate conformance testing.

possible in the predefined model but parts of the process model are not addressed by the log.[4] Event log *L2* seems to indicate overfitting, i.e., the upper half of the process model is not used according to the log. Given a fixed set of activities X it is easy to construct a Petri net that fits all logs, simply construct the Petri net with one place p and one transition per activity. Then connect each transition with this p such that p is both and input and output place. The resulting net fits on any log with just activities of X. Therefore, the predicative value of the resulting Petri net is negligible.

Clearly both underfitting and overfitting are highly relevant. However, it is difficult to quantify overfitting. For example, certain paths in the process model may only be used for specific classes of cases (e.g., rush orders) or during specific periods (e.g., during the peak season). The fact that these are not used in some log does not indicate that the process model is necessarily incorrect. However, underfitting can easily be detected as shown by event logs *L3* and *L4*. Therefore, we only consider underfitting in the context of conformance testing.

To conclude this section, we consider some metrics for conformance testing. All metrics presented are based on measuring the degree of underfitting. Let P be a process model and L a log. A case c in L can be successfully parsed if and only if c corresponds to a firing sequence of P and after executing this sequence only the sink place is marked (i.e., one token in place *end*).

The first metric *fit1(P,L)* simply counts the number of cases in L that can be parsed by P divided by the number of cases. For example, *fit1(P,L1) = 1* while *fit1(P,L3) = 0.6* (only three of the five cases can be parsed). This metric does not take into account the number of events. For large process models this may be too simplistic, i.e., the metric does not distinguish between cases that are close to the model and cases that are completely unrelated. Fortunately, it is also possible to simply count the number of events that can be executed. This can be done using blocking or non-blocking semantics. If we assume blocking semantics, the parsing of the case stops after the first problem is detected. If we use non-blocking semantics, even disabled transitions are fired so that the parsing of the case can continue. Consider for example, *case 1* in event log *L4*. If we use blocking semantics, the parsing stops after executing G because E is not enabled in the Petri net, i.e., only two of the five events can be parsed. If we use blocking semantics, the parsing does not stop after executing G and four of the five events can be parsed (all except E). The second metric *fit2(P,L)* uses blocking semantics to simply count the number of events in L that can be parsed by P divided by the total number of events in the log. Again *fit2(P,L1) = 1*, but *fit2(P,L3) = 25/27 = 0.93* and *fit2(P,L3) = 25/31 = 0.81*. The third metric *fit3(P,L)* uses non-blocking semantics to count the faction of events in L that can be parsed by P. Now *fit3(P,L3) = 25/27 = 0.93* and *fit3(P,L3) = 27/31 = 0.87*. Clearly, *fit3(P,L)* is a better metric than *fit2(P,L)* because the blocking

[4] The term "overfitting" may seem incorrect in this context because we are not really constructing a model. However, in this section we use it as an antonym for the term "underfitting", i.e., the model allows for more behavior than the behavior that actually occurs in real-life.

semantics is very sensitive for the position of the anomaly in the case, i.e., "early problems" outweigh "late problems". Note that for *fit2(P,L)* and *fit3(P,L)* we did not specify penalties for tokens remaining in the net after parsing the case. It is possible to define alternative metrics to also take this into account. Table 3 summarizes the results for the three metrics.

event log	*fit1(P,L)*	*fit2(P,L)*	*fit3(P,L)*
L1	1.00	1.00	1.00
L2	1.00	1.00	1.00
L3	0.60	0.93	0.93
L4	0.60	0.81	0.87

Table 3. Measuring conformance using three metrics based on underfitting.

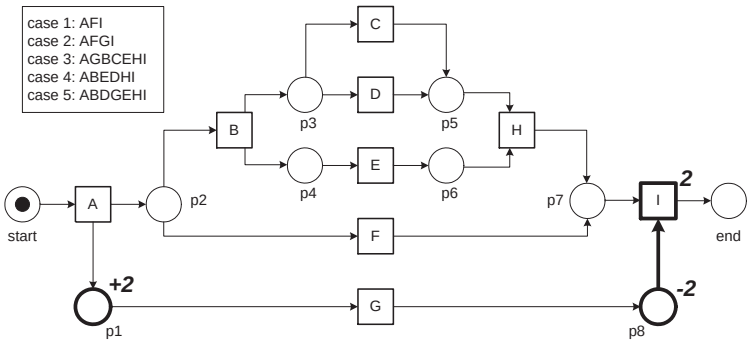


Fig. 10. Some diagnostics showing the mismatch between Figure 9(a) and event log *L3*.

By trying to parse cases using the predefined model it is also possible to give diagnostics that may help to locate the source of the misalignment. For this purpose, we use the non-blocking semantics. While parsing the log it is possible to count how many times an activity should have been executed according to the log but could not be executed according to the predefined model. Figure 10 shows that for event log *L4* activity *I* did not fit twice. (Note the "2" next to activity *I*.) This was caused by the fact that in there was a token missing in place *p8*. This is indicated by the "-2" next to place *p8*. For two cases a token was missing in place *p8* because activity *G* did not occur after executing the initial activity *A*. Therefore, there were two cases where a token was left in place *p1*. This is indicated by the "+2" next to place *p1*. Similar diagnostics are shown for event log *L4* in Figure 11. Here activity *E* could not be parsed for the first two cases. In both cases a token was missing in place *p4* (see "-2") and because of the

non-blocking semantics two tokens remained in place *p6* (see "+2"). Figures 10 and 11 show that we can pinpoint the parts of the process where there appears to be a misalignment.

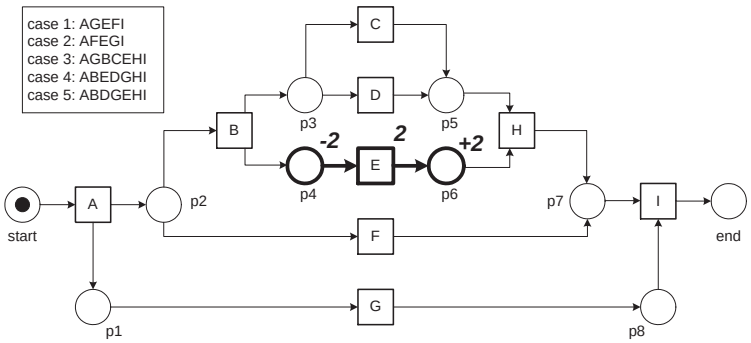


Fig. 11. Some diagnostics showing the mismatch between Figure 9(a) and event log *L4*.

In this section, we demonstrated that conformance testing (i.e., comparing the log with some predefined process model) can be used to measure the fitness of the model based on a log. Moreover, the same mechanisms can be used to locate the misalignment.

6 Related work

This paper extends our BPMDS 2004 workshop paper (cf. [4]) in several ways. For example, the current version includes concrete techniques for Delta analysis and conformance testing. (In fact, in [4] conformance testing is not considered.)

Recently many authors focused on business alignment [3]. An example is [22] where the author examines the techniques that eight organizations have used to both monitor and improve the alignment and performance of their IS functions.

The idea of process mining is not new [23, 1, 6, 24–30] and has been mainly aiming at the control-flow perspective. For more information on process mining we refer to a special issue of Computers in Industry on process mining [2] and a survey paper [1]. In this paper, it is impossible to do justice to the work done in this area. However, we would like to highlight [31] where Cook and Wolf provide a measure to quantify discrepancies between a process model and the actual behavior as registered using event-based data.

The work in [23, 1, 6, 24–30] is primarily focusing on the process perspective. However, there are clear links with sociometry, and Social Network Analysis (SNA) in particular. Since the early work of Moreno [32] SNA has been an active research domain. There is a vast amount of textbooks, research papers, and

tools available in this domain [33–38, 32, 39–41]. There have been many studies analyzing workflow processes based on insights from social network analysis. However, these studies typically have an ad-hoc character and sociograms are typically constructed based on questionnaires rather than using a structured and automated approach as described in this paper. Most tools in the SNA domain take sociograms as input. MiSoN and ProM are only some of the few tools that generate sociograms as output. The only comparable tools are tools to analyze e-mail traffic, cf. BuddyGraph (http://www.buddygraph.com/) and MetaSight (http://www.metasight.co.uk/). However, these tools monitor unstructured messages and cannot distinguish between different activities (e.g., work-related interaction versus social interaction).

For Delta analysis one needs to compare processes. This is not as trivial as is may seem. Judging by the vast number of equivalence notions [42] researchers do not even agree on the equivalence of processes. One of the ways to establish relations between processes is through notions of "process inheritance" [19]. By using such notions it is possible to calculate what two processes have in common [18]. In [43] there is a more elaborate discussion on Delta analysis based on notions of inheritance.

Finally, we would like to point out that the notion of conformance is related to security as is discussed in [44].

7 Conclusion

This paper discussed the application of process mining to business alignment. For this purpose two fundamental assumptions are made. First, we assume that events are actually logged by some information system. Note that this assumption is valid in some, but definitely not all, situations. However, the current focus on "Corporate Governance" and governmental regulations such as Sarbanes-Oxley Act trigger the development of IT systems that log events. Also new technologies such as RFID, Bluetooth, WLAN, etc. and the omnipresence of small computing devices (e.g., mobile phones and PDAs) enable ubiquitous and/or mobile systems that can be used to record human behavior and business processes in detail. The second fundamental assumption is that people are not completely "controlled" by the system, i.e., process mining does not give any insight if all decisions are made by the system and users cannot deviate from the default path. Although the degree of freedom is limited by some systems (e.g., traditional production workflow systems) the trend is towards more flexible systems.

If we assume that events are logged and that users are able to deviate from the default path, then process mining techniques can be used to investigate the alignment of the process models driving the information system and the actual business process. We discussed two approaches: Delta analysis and conformance testing. Delta analysis can be used to compare the prescriptive/descriptive model and the discovered model. Conformance testing can be used to quantify the fit between the prescriptive/descriptive model and the actual events. Moreover, we demonstrated that conformance testing can be used to locate the misalignment.

A clear limitation of the approach presented in this paper is that we need to assume that events are actually logged by some information system. This way misalignment may be hidden (e.g., manual or unrecorded activities). Therefore, an interesting approach could be to combine the quantitative process log data with qualitative data gathered from other sources (e.g., interviews and questionnaires). This combination can help to understand potential problems more effectively. Further research is needed to clarify this.

Acknowledgements
The author would like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song, Laura Maruster, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, Anne Rozinat and Peter van den Brand for their on-going work on process mining techniques. Parts of this paper have been based on earlier papers with these researchers.

References

1. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
2. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
3. I. Bider, G. Regev, and P. Soffer, editors. *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*. Riga Technical University, Latvia, 2004.
4. W.M.P. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*, pages 138–145. Riga Technical University, Latvia, 2004.
5. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.
6. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
7. L. Maruster. *A Machine Learning Approach to Understand Business Processes*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2003.
8. J. Bemmel and M.A. Musen. *Handbook of Medical Informatics*. Springer-Verlag, 1997.
9. E. Coiera. *Guide to Health Informatics*. Arnold Publishers, 2003.
10. M. Peleg, , S. Tu, J. Bury, P. Ciccarese, J. Fox, R.A. Greenes, R. Hall, P.D. Johnson, N. Jones, A. Kumar, Silvia Miksch, S. Quaglini, A. Seyfang, E.H. Shortliffe, and M. Stefanelli. Comparing Computer-interpretable Guideline Models: A Case-study Approach. *Journal of the American Medical Informatics Association*, 10(1):52–68, 2003.

11. J. Fox, N. Johns, and A. Rahmanzadeh. Disseminating Medical Knowledge: The PROforma Approach. *Artificial Intelligence in Medicine*, 14(1):157–182, 1998.
12. S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa. Flexible Guideline-based Patient Careflow Systems. *Artificial Intelligence in Medicine*, 22(1):65–80, 2001.
13. P.A. de Clerq, J.A. Blom, H.H. Korsten, and A. Hasman. Approaches for Creating Computer-Interpretable Guidelines that Facilitate Decision Support. *Artificial Intelligence in Medicine*, 31(1):1–27, 2004.
14. Pallas Athena. *Flower User Manual*. Pallas Athena BV, Apeldoorn, The Netherlands, 2002.
15. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
16. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. QUT Technical report, FIT-TR-2003-05, Queensland University of Technology, Brisbane, 2003.
17. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
18. W.M.P. van der Aalst and T. Basten. Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance. In J.M. Colom and M. Koutny, editors, *Application and Theory of Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 32–52. Springer-Verlag, Berlin, 2001.
19. T. Basten and W.M.P. van der Aalst. Inheritance of Behavior. *Journal of Logic and Algebraic Programming*, 47(2):47–145, 2001.
20. W.M.P. van der Aalst. Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change. *Information Systems Frontiers*, 3(3):297–317, 2001.
21. C.A. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pages 10 – 21, Milpitas, California, August 1995. ACM SIGOIS, ACM Press, New York.
22. Y.E. Chan. Why Haven't we Mastered Alignment?: The Importance of the Informal Organization Structure. *MIS Quarterly Executive*, 1(2):95–110, 2002.
23. W.M.P. van der Aalst and B.F. van Dongen. Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, volume 2480 of *Lecture Notes in Computer Science*, pages 45–63. Springer-Verlag, Berlin, 2002.
24. 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.
25. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
26. 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.
27. IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.

28. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
29. M. Sayal, F. Casati, U. Dayal, and M.C. Shan. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.
30. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
31. J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology*, 8(2):147–176, 1999.
32. J.L. Moreno. *Who Shall Survive?* Nervous and Mental Disease Publishing Company, Washington, DC, 1934.
33. A.A. Bavelas. A Mathematical Model for Group Structures. *Human Organization*, 7:16–30, 1948.
34. H.R. Bernard, P.D. Killworth, C. McCarty, G.A. Shelley, and S. Robinson. Comparing Four Different Methods for Measuring Personal Social Networks. *Social Networks*, 12:179–216, 1990.
35. R.S. Burt and M. Minor. *Applied Network Analysis: A Methodological Introduction*. Sage, Newbury Park CA, 1983.
36. M. Feldman. Electronic Mail and Weak Ties in Organizations. *Office: Technology and People*, 3:83–101, 1987.
37. L.C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40:35–41, 1977.
38. L.C. Freeman. Centrality in Social Networks: Conceptual Clarification. *Social Networks*, 1:215–239, 1979.
39. H. Nemati and C.D. Barko. *Organizational Data Mining: Leveraging Enterprise Data Resources for Optimal Performance*. Idea Group Publishing, Hershey, PA, USA, 2003.
40. J. Scott. *Social Network Analysis*. Sage, Newbury Park CA, 1992.
41. S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, 1994.
42. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
43. W.M.P. van der Aalst. Inheritance of Business Processes: A Journey Visiting Four Notorious Problems. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri Net Technology for Communication Based Systems*, volume 2472 of *Lecture Notes in Computer Science*, pages 383–408. Springer-Verlag, Berlin, 2003.
44. W.M.P. van der Aalst and A.K.A. de Medeiros. Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance. In N. Busi, R. Gorrieri, and F. Martinelli, editors, *Second International Workshop on Security Issues with Petri Nets and other Computational Models (WISP 2004)*, pages 69–84. STAR, Servizio Tipografico Area della Ricerca, CNR Pisa, Italy, 2004.