Configurable Services in the Cloud: Supporting Variability While Enabling Cross-Organizational Process Mining

Wil M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
w.m.p.v.d.aalst@tue.nl

Abstract. The *Software as a Service* (SaaS) paradigm is particularly interesting in situations where many organizations need to support similar processes. For example, municipalities, courts, rental agencies, etc. all need to support highly similar processes. However, despite these similarities, there is also the need to allow for local variations in a controlled manner. Therefore, cloud infrastructures should provide configurable services such that products and processes can be customized while sharing commonalities. Configurable and executable process models are essential for realizing such infrastructures. This will finally transform reference models from "paper tigers" (reference modeling à la SAP, ARIS, etc.) into an "executable reality". Moreover, "configurable services in the cloud" enable *cross-organizational process mining*. This way, organizations can learn from each other and improve their processes.

Keywords: Configurable process models, Process Mining, Business Process Management, YAWL, ProM

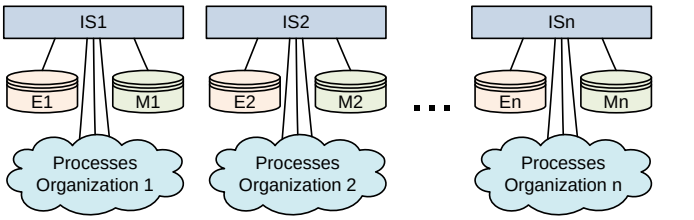
1 Motivation

Cloud computing is not a new idea. In 1961, in a speech given to celebrate MIT's centennial, John McCarthy stated "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility. The computer utility could become the basis of a new and important industry." In 1961, even ARPANET, the predecessor Internet, did not exist and it is remarkable that people like John McCarthy, who received the Turing Award in 1971 for his work on AI, could predict that computing would become a utility as is signified today by Gmail, Google Apps. Salesforce.com, Amazon EC2/S3, etc. *Cloud computing* is typically defined as Internet-based computing, whereby shared resources, software, and information are provided on demand, like the electricity grid. The term is closely related to the notion of *Software as a Service* (SaaS). SaaS refers to a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. The terms SaaS and cloud computing are strongly related. People talking about cloud computing tend to emphasize the computing infrastructure and

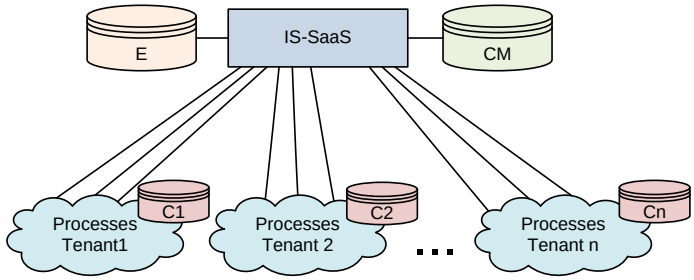
combine this with a broad vision on computing as a utility. The term SaaS tends to emphasize the role of services that are provided and consumed. SaaS service providers typically offer a subscription model where service consumers do not pay for software but only pay for the actual use of software. A well-known example of a SaaS provider that is using a cloud infrastructure is SalesForce.com. This company allows organizations to outsource the IT support of standard functionality such as sales, customer relationship management, etc. without worrying about scalability and maintenance. Another example is the conference management system EasyChair that is currently probably the most commonly used system to host conferences and to manage the reviewing of scientific papers. To organize a conference, there is no need to install any software as everything is hosted and managed centrally.

Cloud computing and SaaS have in common that multiple organizations, often called *tenants*, are sharing the same infrastructure/software. This provides many advantages: lower costs (only pay for actual use), reduced setup times, reduced maintenance and management efforts, etc. However, it also creates the *challenge of dealing with variability across organizations*. It is not realistic to enforce "one size fits all" as tenants may have different needs and preferences. In this paper, we focus on the process perspective and suggest using so-called *configurable process models* to support variability [1, 3, 14–17, 23, 24]. The basic idea of configurable process models is that one model does not represent a single process, but a family of processes. By configuring a configurable process model one obtains a concrete process model that can be executed within an organization. Configurability is essential for the success of SaaS software. Ideally, tenants are provided with a multitude of options and variations using a single code base, such that it is possible for each tenant to have a unique software configuration. Related to variability, there are other concerns raised by multi-tenancy. For example, how to ensure correctness of all possible configurations? It is not sufficient to guarantee the correct operation of a single process. Instead one needs to ensure the correctness of a process family and all of its configurations. Another concern is security; How to make sure that data and processes of different tenants are isolated while using the same code base and infrastructure?

Besides these challenges, there are also many opportunities. Besides the obvious "economies of scale" achievable from consolidating IT resources, there is the possibility to carefully analyze software usage and process executions across different organizations. In the situation where customized enterprise information systems are running inside organizations, the software vendor has little insight into the actual use of its software. Moreover, it is impossible to analyze differences between organizations. In this paper, we suggest using *cross-organizational process mining* using multi-tenancy environments provided through SaaS and cloud computing. The goal of process mining is to use event data to extract process-related information, e.g., to automatically discover a process model by observing events recorded by some enterprise system [5, 4, 6, 9, 11, 12, 18, 26]. Where data mining focuses on relatively simple tasks such as classification (e.g., decision trees), clustering, and regression that aim at analyzing *data*, process mining



(a) Traditional situation with, per process, one model, and a dedicated system and database for each organization



(b) New situation using SaaS technology based on configurable services

Fig. 1. The traditional situation where each organization has it's own IT infrastructure (a) and the situation where each organization is a tenant of a "shared configurable cloud" (b). (IS = Information System, M = Process Model, CM = Configurable Process Model, E = Event Log, and C = Configuration.)

focuses on *operational processes*, i.e., identifying causal dependencies between activities, visualizing bottlenecks inside the discovered process, measuring conformance, detecting deviations, predicting cycle times, etc.

We will use Figure 1 to illustrate the above. Figure 1(a) shows the traditional situation where each organization uses its own infrastructure, process models $(M_1, M_2, \ldots M_n)$, event logs $(E_1, E_2, \ldots E_n)$, and information system $(IS_1, IS_2, \ldots IS_n)$. Note that we assume that organizations are using a *Process-Aware Information System* (PAIS) [10], i.e., a system that is driven by some process model and that is recording events. Note that Workflow Management (WfM) systems, Business Process Management (BPM) systems, Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM) systems, etc. are all examples of such systems. Some of these systems are driven by explicit graphical process models and record events in a structured manner, while others are supporting processes in a more implicit manner and some efforts are needed to extract event logs suitable for analysis. In any case, processes are being supported and it is possible to extract the information required for process mining. Each organization has complete freedom to change processes

provided that they have the resources needed to re-configure or replace parts of the information system.

Figure 1(b) shows the situation where the n organizations have become tenants of a shared SaaS system. In this paper, we propose using *one configurable model* (CM) per service and each tenant uses a particular configuration for this service ($C_1, C_2, \ldots C_n$).[1] The configurable model needs to be able to support meaningful variations of the same process required by the different organizations. In the new situation events are recorded in a unified manner. This allows for comparing processes within different organizations. We have identified two possible use cases for *cross-organizational process mining*.

- The *service provider* can use the event logs of all tenants to *improve its services* and *provide guidance* to tenants when configuring their processes. Note that the focus can be on the software or on the processes supported by the software. For example, the service provider may note that there are situations where the system has poor response times or even fails. Using process mining it is possible to identify possible causes for such problems. Moreover, the usability of the system can be analyzed, e.g., measuring the time to complete a task or the number of retries. However, the service provider can also analyze differences in the processes supported by these systems, e.g., comparing flow times of different organizations. These can be used to give advice to individual organizations. Note that data privacy is an important factor. However, the service provider can do many types of analysis without looking into sensitive data, e.g., the identities of individual workers or customers are irrelevant for most analysis questions and no information about one tenant is shared with other tenants.
- Another use case is where multiple comparable *organizations wish to share information*. In this case, cross-organizational process mining is used to benchmark different organizations and differences in performance are analyzed. This is of course only possible in a non-competitive environment, e.g., different branches of some multinational organization, franchises, municipalities, courts, etc. The goal is to let organizations learn from each other and establish proven best practices. Of course privacy issues may again complicate such analysis, however, it may be sufficient to compare things at an aggregate level or to anonymize the results.

The goal of this paper is to discuss the opportunities and challenges provided by cloud computing and SaaS. In particular, we focus on the need for process configuration and the opportunities provided by cross-organizational process mining. This will be illustrated by a short description of the *CoSeLoG* (Configurable Services for Local Governments) project. Ten municipalities and two software organizations are involved in this project. The goal of CoSeLoG is to develop and analyze configurable services for local municipalities while using the SaaS paradigm, process configuration, and process mining.

[1] We assume that any service is characterized by an interface that describes the information exchanged. However, in this paper we focus on the process-related aspects of a service. Therefore, we will use the terms "service" and "process" interchangeably.

Before introducing the CoSeLoG project in Section 4, we first present our ideas related to process configuration in the cloud (Section 2) followed by an introduction to cross-organizational process mining (Section 3).

2 Turning "Paper Tigers" into an "Executable Reality"

In this section we focus on configurable process models in a SaaS setting. These enable service providers to support variability among different organizations in a structured manner.

2.1 The Need for Configurable Process Models

Although large organizations support their processes using a wide variety of Process-Aware Information Systems (PAISs) [10], the majority of business processes are still not directly driven by explicit process models. Despite the success of Business Process Management (BPM) thinking in organizations, Workflow Management (WfM) systems — today often referred to as *BPM systems* — are not widely used. One of the main problems of BPM technology is the "lack of content", that is, providing just a generic infrastructure to build process-aware information systems is insufficient as organizations need to support specific processes. Organizations want to have "out-of-the-box" support for standard processes and are only willing to design and develop system support for organization-specific processes. Yet most BPM systems expect users to model basic processes from scratch. Enterprise Resource Planning (ERP) systems such as SAP and Oracle, on the other hand, focus on the support of these common processes. Although all main ERP systems have workflow engines comparable to the engines of BPM systems, the majority of processes are not supported by software that is directly driven by process models. For example, most of SAP's functionality is not grounded in their workflow component, but hard-coded in application software. ERP vendors try to capture "best practices" in dedicated applications designed for a particular purpose. Such systems can be configured by setting parameters. System configuration can be a time consuming and complex process. Moreover, configuration parameters are exposed as "switches in the application software", thus making it difficult to see the intricate dependencies among certain settings.

A model-driven process-oriented approach toward supporting business processes has all kinds of benefits ranging from improved analysis possibilities (verification, simulation, etc.) and better insights, to maintainability and ability to rapidly develop organization-specific solutions. Although obvious, this approach has not been widely adopted thus far, probably because BPM vendors have failed to provide content and ERP vendors suffer from the "Law of the handicap of a head start". ERP vendors manage to effectively build data-centric solutions to support particular tasks. However, the complexity and large installed base of their products makes it impossible to refactor their software and make it truly process-centric.

Based on the limitations of existing BPM and ERP systems, we propose to use *configurable process models*. A configurable process model represents a *family of process models*, that is, a model that through configuration can be customized for a particular setting. Configuration is achieved by *hiding* (i. e., bypassing) or *blocking* (i. e., inhibiting) certain fragments of the configurable process model [14]. In this way, the desired behavior is selected. From the viewpoint of generic BPM software, configurable process models can be seen as a mechanism to add content to these systems. By developing comprehensive collections of configurable models, particular domains can be supported. From the viewpoint of ERP software, configurable process models can be seen as a means to make these systems more process-centric, although in the latter case quite some refactoring is needed as processes are hidden in table structures and application code.

Various configurable languages have been proposed as extensions of existing languages (e. g., C-EPCs [24], C-iEPCs, C-WF-nets [1], C-SAP, C-BPEL) but few are actually supported by enactment software (e. g., C-YAWL [16]). In this paper, we are interested in the latter class of languages, which, unlike traditional reference models [8, 7, 13], are executable after they have been configured. In this paper, we focus on configurable services offered over the Internet. Therefore, the models need to be executable to be of any use.

2.2 An Example: C-YAWL

As an example of a configurable language we briefly describe C-YAWL [16, 17]. YAWL is a process modeling notation and workflow environment based on Petri nets but extended with powerful features for cancelation, OR-joins, etc. It has been developed with the aim to provide a notation with formal semantics that supports all desired workflow patterns [19]. The YAWL system is open-source and supports the execution and work distribution of workflows depicted in such models even in production environments. Thus, although originally developed as a proof of concept, the YAWL system can be used for practical applications [19].

Figure 2 depicts a simple YAWL model for the process executed by municipalities when a man registers that he will become the father of a not-yet-born child although he is not married to the mother [17]. In this model tasks are depicted as rectangles while circles represent conditions like the initial and final condition in this example. Conditions mark the states between tasks but can be omitted for simplicity (like in the example). Composite tasks enable the hierarchical specification of sub-processes while split and join types of tasks allow the specification of how the process should proceed in case a task splits or joins the process's control flow. For this, YAWL distinguishes an XOR-split (as in the example in Figure 2) allowing the triggering of only one of the subsequent paths, an AND-split requiring the triggering of all subsequent paths, and an OR-split requiring the triggering of at least one subsequent path but allowing also for path combinations. Similarly, a task with an XOR-join can be executed as soon as one of its incoming paths is triggered, an AND-join requires that all incoming

arcs are triggered, and a task with an OR-join allows for the execution of the task as soon as no further incoming paths can potentially be triggered at any future point in time (see [19] for further details).

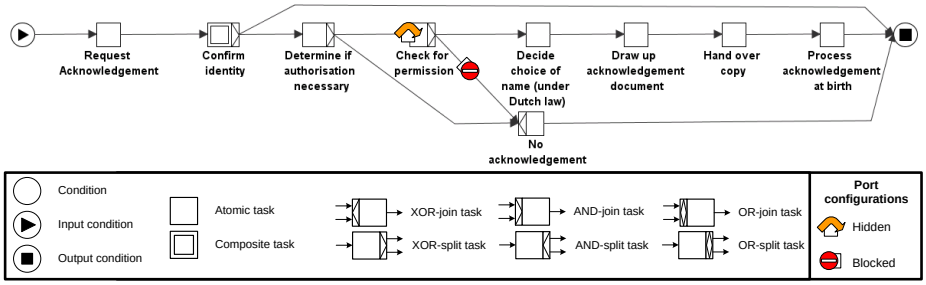


Fig. 2. A YAWL process model for acknowledging an unborn child [17]. The input port of *check permission* is configured as hidden and one output port is blocked.

This routing behavior can be restricted by *process configuration*. For this purpose, *input ports* are assigned to each task depicting how the task can be triggered and *output ports* are assigned to depict which paths can be triggered after the completion of the task. A task with an XOR-join can be triggered via each of its incoming paths. Thus, it has a dedicated input port for each of these paths. Tasks with AND-joins and OR-joins can only be executed if all paths (that can potentially be triggered) are triggered, i.e. there is only one way these tasks can be triggered and thus there is only one input port. A task with an XOR-split has an output port for each subsequent path as each of these paths can be triggered individually while a task with an AND-split has only one output port as all subsequent tasks must always be triggered. A task with an OR-split can trigger a subset of the outgoing paths, i.e. in this case a separate output port exists for each of these combinations.

The process flow can be restricted at these ports. A *blocked* port prevents the process flows through it, i.e. a blocked input port prevents the triggering of the task through the port while a blocked output port prevents that the corresponding output paths can be triggered. In the model in Figure 2, we blocked the output port from *Check for permission* to *No acknowledgement*. Thus, the task *Check for permission* must always be followed by the task *Decide choice of name (under Dutch law)* as the path to the task *No acknowledgement* can no longer be triggered. Input ports can not only be blocked but also be configured as *hidden*. Similarly, the subsequent task can then not be triggered through this port anymore. However, in this case the process flow is not completely blocked, but only the execution of the corresponding task is skipped. The process execution continues afterwards. In Figure 2 the input port of the task *Check for permission* is hidden. Thus, the execution of this task is skipped which also explains why we blocked one of the task's output ports: the configuration results

in skipping the check. Hence, it can no longer fail and the process must continue normally. Further details on configurable YAWL can be found in [16, 19].

As we can observe from this example, the configurations of ports are often not independent from each other and require extensive domain knowledge. In [23] it is shown how domain knowledge can be taken into account and used to drive the configuration process. In [1, 3] different techniques are provided for ensuring the correctness of the resulting models.

2.3 Challenges

There are many challenges related to process configuration. First of all, there is a need to develop *complete collections of high-quality configurable models*. Often the ideas and the technology are in place, but the actual "content" is missing or of very low quality (see for example the many errors in SAP's well-known reference model [21]). It is important to develop a *sound methodology* to extract best practise models. Process mining can help to find out what the actual processes are and how they perform. A second challenge is how to *extract a manageable configurable process from a set of concrete models*. As shown in [17] techniques from process mining can be adapted for this purpose. However, the resulting models are rather spaghetti-like. In [20] another approach, more related to ad-hoc change, is used. Here a reference model is chosen that requires the least number of edit operations. Unfortunately, one needs to manually modify the reference model to create a selected variant. Finally, there are many issues related to multi-tenancy, flexibility and change. How to change a configurable model used by many tenants? How to ensure privacy and isolation? How to accommodate exceptional requests that do not fit the configurable model?

3 Cross-Organizational Process Mining

This section first provides a high-level overview of process mining techniques. Subsequently, we introduce the concept of cross-organizational process mining and discuss the corresponding challenges.

3.1 Process Mining in One Organization

More and more information about (business) processes is recorded by information systems in the form of so-called "event logs" (e.g., transaction logs, audit trails, databases, message logs). IT systems are becoming more and more intertwined with the processes they support, resulting in an "explosion" of available data that can be used for analysis purposes. Cloud computing and SaaS will fuel this development even more.

To illustrate the role that event logs can play, let us first explain Figure 3. We assume the existence of a collection of information systems that are supporting a "world" composed of business processes, people, organizations, etc. The *event data* extracted from such systems are the starting point for *process mining*. Note

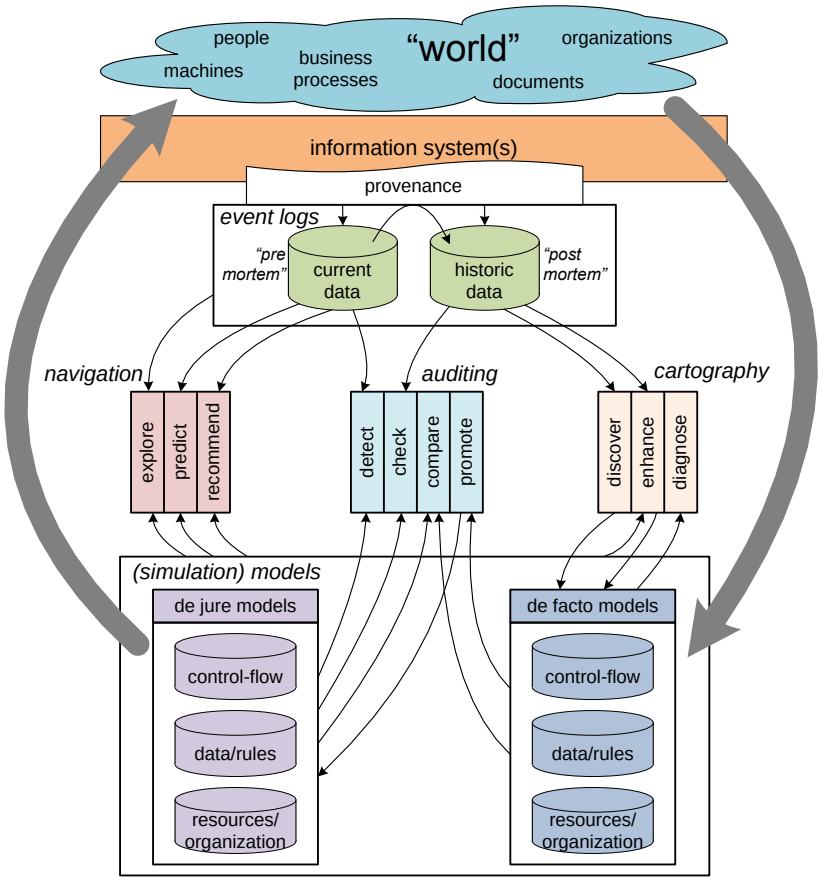


Fig. 3. Overview of various process mining activities.

that Figure 3 distinguishes between *current data* and *historic data*. The former refers to events of cases (i.e., process instances) that are still actively worked on ("pre mortem"). The latter refers to events of completed cases, i.e., process instances that cannot be influenced anymore ("post mortem"). The historic data ("post mortem") can be any collection of events where each event refers to an instance (i.e., case), has a name (e.g., activity name), and has a timestamp.

The collection of event data is becoming more important. One the one hand, more and more event data are available. On the other hand, organizations depend on such data; not only for performance measurement, but also for auditing. We use the term *business process provenance* to refer to the systematic collection of the information needed to reconstruct what has actually happened. The term signifies that for most organizations it is vital that "history cannot be rewritten or obscured". From an auditing point of view the systematic, reliable, and trust-

worthy recording of events is essential. Fortunately, cloud computing and SaaS can assist in the systematic and unified collection of event data.

The lower part of Figure 3 shows two types of models: *de jure models* are normative models that describe a desired or required way of working while *de facto models* aim to describe the actual reality with all of its intricacies (policy violations, inefficiencies, fraud, etc.). Both types of models may cover one or more perspectives and thus describe control-flow, time, data, organization, resource, and/or cost aspects. For process mining one can focus on a particular perspective. However, when the goal is to build simulation models, all factors influencing performance need to be taken into account (e.g., when measuring utilization and response times, it is not possible to abstract from resources and focus on control-flow only). Models can also be based on a mixture of "de jure" and "de facto" information. The key idea of process mining is to not simply rely on de jure models that may have little to do with reality. Therefore, the goal is to shift more to "de facto models"; this will save time and increase the quality of analysis results.

In Figure 3 three main categories of activities have been identified: *cartography*, *auditing*, and *navigation*. The individual activities are briefly described below.

1. **Discover.** The discovery of good process models from events logs - comparable to geographic maps - remains challenging. Process discovery techniques can be used to discover process models (e.g., Petri nets) from event logs [4, 5].
2. **Enhance.** Existing process models (either discovered or hand-made) need to be related to events logs such that these models can be enhanced by making them more faithful or by adding new perspectives based on event data. By combining historic data and pre-existing models, these models can be repaired (e.g., a path that is never taken is removed) or extended (e.g., adding time information extracted from logs).
3. **Diagnose.** Models (either de jure or de facto) need to be analyzed using existing model-based analysis techniques, e.g., process models can be checked for the absence of deadlocks or simulated to estimate cycle times. Probably the most widely used model-based analysis technique is simulation.
4. **Detect.** For on-line auditing, de jure models need to be compared with current data (events of running process instances) and deviations of such partial cases should to be detected at runtime. By replaying the observed events on a model, it is possible to do conformance checking while the process is unfolding.
5. **Check.** Similarly, historic "post mortem" data can be cross-checked with de jure models. For this conformance checking techniques are used that can pinpoint deviations and quantify the level of compliance [25].
6. **Compare.** De facto models can be compared with de jure models to see in what way reality deviates from what was planned or expected.
7. **Promote.** Based on an analysis of the differences between a de facto model and a de jure model, it is possible to promote parts of the de facto model to

a new de jure model. By promoting proven "best practises" to the de jure model, existing processes can be improved. For example, a simulation model may be improved and calibrated based on elements of a de facto model.

8. **Explore.** The combination of event data and models can be used to explore business processes. Here new forms of interactive process visualization can be used (visual analytics).
9. **Predict.** By combining information about running cases with models (discovered or hand-made), it is possible to make predictions about the future, e.g., the remaining flow time and the probability of success. Here techniques such as simulation and regression analysis can be used.
10. **Recommend.** The information used for predicting the future can also be used to recommend suitable actions (e.g. to minimize costs or time). The goal is to enable functionality similar to the guidance given by navigation systems like TomTom, but now in the context of BPM.

3.2 Example: Control-Flow Discovery

It is impossible to give concrete examples for all process mining techniques referred to in Figure 3. Therefore, we only illustrate the first activity, i.e., process discovery. Input for process discovery and any other process mining technique is an event log. 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 characteristically contain information on the person executing or initiating the event, i.e., the *performer*. Also any other data can be attached to events. Various process mining techniques depend on subsets of this information. For example, techniques focusing on performance take timestamps into account, techniques focusing on decision points take data attributes into account, techniques focusing on the organizational perspective take performers into account. Figure 4 shows the minimal input required for applying the so-called α algorithm [5]. The left-hand side represents cases as sequences of activities, also referred to a traces. Every sequence corresponds to a case. For example for the first case, A, B, C, and D are executed. For the second case, the same activities are executed but B and C are reversed. Etc. The traces in Figure 4 suggest that the process always starts with A and always ends with D. In-between A and D either E or B and C are executed. The α algorithm analyzes the log for particular patterns and deduces the Petri net model shown on the right-hand side of Figure 4. Note that the Petri net indicates that after A either just E or both B and C are executed. Activities B and C are put in parallel. Activity D either waits for the completion of E or needs to wait until both B and C complete.

The α algorithm is able to identify all of the common control-flow patterns (AND/XOR-split/join, loops, etc.) [5]. However, it has many limitations when applied to real-life logs. Fortunately, many more mature techniques exist [4, 18,

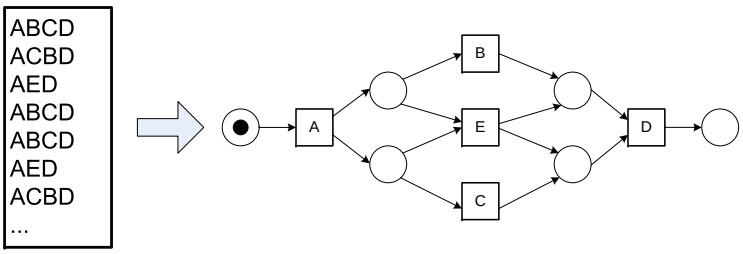


Fig. 4. Process discovered using the α algorithm [5].

26]. The α algorithm also only uses a subset of the information available and is restricted to the control-flow perspective.

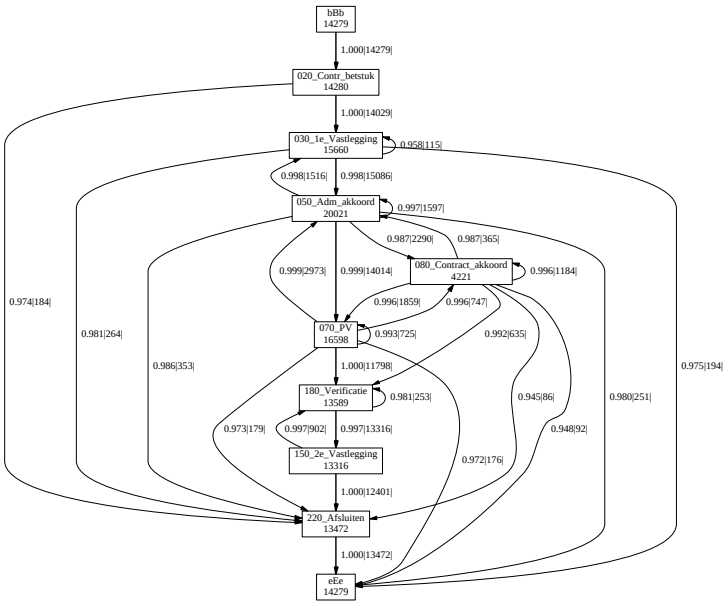


Fig. 5. Discovered process for invoice payments in RWS [4].

Figure 5 shows an example of a real life process discovered through process mining. It is the invoice payment process of one of the twelve provincial offices of "Rijkswaterstaat", the Dutch national public works department, often abbreviated as "RWS". The process was discovered based on the event logs of RWS's information system. The goal was to find out what the real process was and use this information to improve and streamline it. Figure 5 shows a particular view

on the control-flow of the RWS process. We also discovered models for other perspectives such as the organizational perspective, the time perspective, etc. See [4] for a detailed analysis. We have been applying process mining in over 100 organizations. Typically, we see that processes are less structured than people think. Moreover, conformance checking typically reveals many deviations and inefficiencies.

3.3 Process Mining in Multi-Tenancy Environments

Thus far process mining research mainly focused on the analysis of a single process typically residing in one organization. Some authors have investigated interactions between web services [2, 11, 22], however, the focus is always on a single process. In a multi-tenancy environment provided by a cloud or SaaS infrastructure, there will be *many variants of the same process* running in parallel. This creates many interesting challenges.

Assume that there are n configured processes $P_1, P_2, \ldots, P_n$ that are all variants of some configurable model CM. Each of these processes has a configuration C_k and a set of process instances (cases) I_k (with $k \in \{1, 2, \ldots n\}$). Using conventional techniques, one can derive a model for every variant, e.g., model M_k is derived from I_k using some process discovery algorithm. It is also possible to derive a model based on all instances; model M^* is derived from $I^* = \bigcup_k I_k$. M^* can be seen as the "least common multiple" of all variants. If no configurable model is given and only the variants are given, then M^* can serve as a starting point for constructing CM.

The challenge is to compare the different process variants and their performance. Note that different processes may share the same configuration but operate under different circumstances. For example, two tenants may use the same configuration, but one has a only a few customers while the other has many. Each of the configured processes has a set of *features*. These features are based on properties of the process model M_k, properties of the configuration C_k, and performance related properties such as average flow times, average response times, service levels, frequencies, etc. Using *clustering* one can group process variants into coherent clusters. Cluster analysis or clustering is "the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense". Using *classification* one tries to explain one feature in terms of other features, e.g., processes with a particular configuration tend to have a better performance. A common technique is decision tree learning. Clustering is sometimes referred to as unsupervised learning while classification is referred to as supervised learning. The large scale adoption of multi-tenancy environments will enable machine learning techniques such as clustering and classification. This way cross-organizational process mining comes into reach.

Cross-organizational process mining is an unexplored area. One of the reasons is that this requires comparable event logs, i.e., events need to be recorded in a consistent manner across multiple organizations. Fortunately, this can easily be achieved in SaaS and cloud infrastructures. Even if data is collected in a unified

manner across different organizations, there are still several challenges. First of all, there is the concern that enough variants of the same process should be available to enable learning. Second, there is the problem of *concept drift*.[2] The same process variant may operate under different circumstances. For example, there may be seasonal effects affecting the features of a process. The same process may have long flow times in December and short flow times in January due to differences in workload. This should be taken into account when comparing variants. In fact, the analysis of concept drift in processes is related to cross-organizational process mining. Instead of comparing different processes operating in the same time period, one can also compare different episodes of the same process.

4 CoSeLoG Project

In this section, we briefly introduce the CoSeLoG project and present an example showing that municipalities form an interesting application domain for the ideas presented in this paper.

4.1 Overview

Since there are 430 municipalities in the Netherlands and they are all providing similar services and are executing similar processes, the use of SaaS technology could potentially be very beneficial for these local governments. The CoSeLoG project was established to exploit this observation. The goal of the project is to create a *cloud infrastructure for municipalities*. More precisely: we want to transition from situation depicted in Figure 1(a) to the situation depicted in Figure 1(b) in a prototypical setting involving several municipalities. Such a cloud infrastructure for municipalities would offer services for handling various types of permits, taxes, certificates, and licences. Although municipalities are similar, their internal processes are typically different. Within the constraints of national laws and regulations, municipalities can differentiate because of differences in size, demographics, problems, and policies. Therefore, the cloud should provide *configurable services* such that products and processes can be customized while sharing a common infrastructure. The CoSeLoG project aims at the development and analysis of such services using the results described in sections 2 and 3.

The following (end-)user organizations are participating in the CoSeLoG proposal: Pallas Athena, D!MPACT, and 10 Dutch municipalities (Bergeijk, Bladel, Coevorden, Eersel, Emmen, Gemert-Bakel, Hellendoorn, Noordoostpolder, Reusel

[2] In machine learning, concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes. In the context of process mining one is not investigating a single variable but a complete process model. This makes it more difficult to properly define this notion.

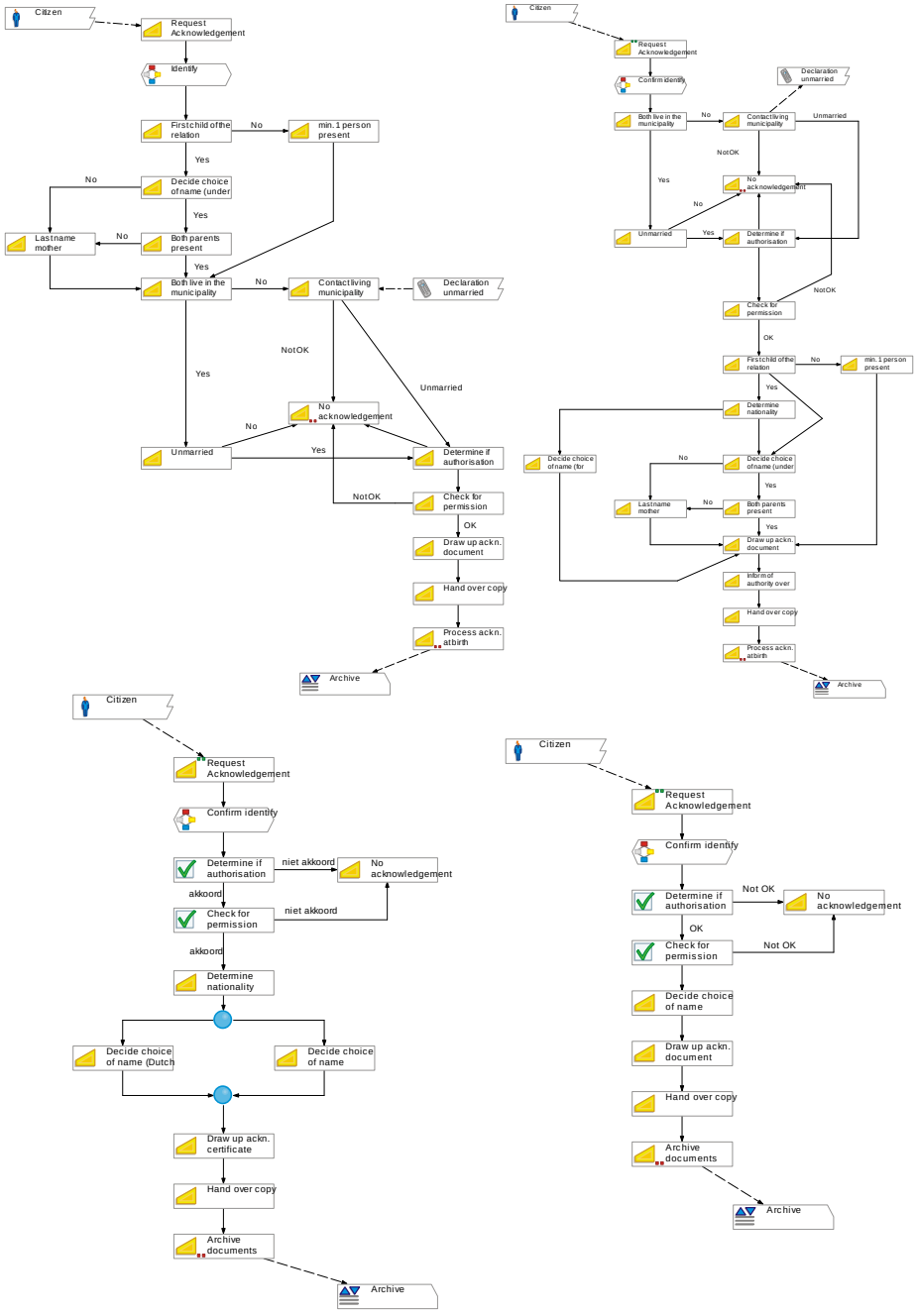


Fig. 6. The process "acknowledgement of an unborn child" in four municipalities [17].

de Mierden, and Zwolle). The project started in May 2010 and is supported by the Jacquard program `www.jacquard.nl` which aims to promote SaaS research.

Municipalities provide an *ideal setting for SaaS, configurable models, and cross-organizational mining*. In principle all 430 municipalities need to offer the same services to their citizens, and need to manage similar collections of processes. However, due to demographics and political choices, municipalities are handling things differently. Sometimes these differences are unintentional, however, often these differences can be easily justified by the desired "Couleur Locale". Hence, it is important to support variability. Interestingly, municipalities are *not* in direct competition with one another. Therefore, cross-organizational process mining is not a threat and municipalities are eager to share information and experiences and learn from each other. Therefore, a widely used cloud infrastructure for municipalities can help to establish best practices based on evidence obtained through cross-organizational process mining.

4.2 Example

In [17] we analyzed four of the most frequently executed processes in municipalities: (a) acknowledging an unborn child, (b) registering a newborn child, (c) marriage, and (d) issuing a death certificate. Any municipality has these processes, however, as we found out, these processes are implemented and executed differently among municipalities. In [17] we compared the processes of four municipalities and the reference model provided by the NVVB (Nederlandse Vereniging Voor Burgerzaken). For example, Figure 6 shows four variants of the process related to "acknowledging an unborn child". Each of the four municipalities is using a specific variant of the processes. Moreover, the NVVB reference model (not shown in Figure 6) is yet another variant of the same process. Based on a detailed analysis of the differences we derived a *configurable process model*, i.e., a model that captures all variants observed. By setting the configuration parameters, one can reconstruct each of the original process models (and many more). The study reported in [17] revealed that: (a) it is possible to construct (or even generate) configurable process models for the core processes in municipalities, (b) municipalities use similar, but at the same time really different, processes, and (c) the comparison of the same process in multiple municipalities provides interesting insights and triggers valuable discussions.

Figure 6 illustrates that it is a challenge to merge different models into one configurable model. As shown in [17] the resulting configurable model tends to be rather complex and difficult to manage. Moreover, it is questionable whether the models shown in Figure 6 adequately reflect the real processes. Using process mining, more realistic models can be discovered and compared across municipalities. The CoSeLoG project will research these problems and, hopefully, provide solutions.

5 Conclusions

In this paper, we discussed configurable services that run in a cloud/SaaS infrastructure where multiple organizations need support for variants of the same process. We showed that supporting variability is one of the main challenges. Moreover, we discussed the potential of process mining techniques in such environments. We believe that "configurable services in the cloud" enable a new kind of process mining, coined "cross-organizational process mining" in this paper. The CoSeLoG project, presented in Section 4, aims to address the challenges presented in this paper.

Acknowledgments. The author would like to thank all the people that contributed to the development of ProM and (C-)YAWL. Special thanks go to Florian Gottschalk and Marcello La Rosa for their seminal work on process configuration and Boudewijn van Dongen and Eric Verbeek for driving the process mining work at TU/e. We thank the Jacquard program for their support and Joos Buijs and Jan Vogelaar for their efforts within the CoSeLoG project.

References

1. W.M.P. van der Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. La Rosa, and J. Mendling. Preserving Correctness During Business Process Model Configuration. *Formal Aspects of Computing*, 22(3):459–482, 2010.
2. W.M.P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H.M.W. Verbeek. Conformance Checking of Service Behavior. *ACM Transactions on Internet Technology*, 8(3):29–59, 2008.
3. W.M.P. van der Aalst, N. Lohmann, M. La Rosa, and J. Xu. Correctness Ensuring Process Configuration: An Approach Based on Partner Synthesis. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management (BPM 2010)*, LNCS, vol. 6336, pages 95–111. Springer-Verlag, 2010.
4. W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32(5):713–732, 2007.
5. 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.
6. 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.
7. J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In J. Becker and P. Delfmann, editors, *Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models*, pages 27–58. Springer, 2007.
8. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
9. A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.

10. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
11. S. Dustdar and R. Gombotz. Discovering Web Service Workflows Using Web Services Interaction Mining. *International Journal of Business Process Integration and Management*, 1(4):256–266, 2006.
12. D.R. Ferreira and D. Gillblad. Discovering Process Models from Unlabelled Event Logs. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, BPM 2009, LNCS, vol. 5701, pages 143–158. Springer-Verlag, Berlin, 2009.
13. P. Fettke and P. Loos. Classification of Reference Models - A Methodology and its Application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.
14. F. Gottschalk, W.M.P. van der Aalst, and H.M. Jansen-Vullers. Configurable Process Models: A Foundational Approach. In J. Becker and P. Delfmann, editors, *Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models*, pages 59–78, Springer, 2007.
15. F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers. SAP WebFlow Made Configurable: Unifying Workflow Templates into a Configurable Model. In G. Alonso, P. Dadam, and M. Rosemann, editors, BPM 2007, LNCS, vol. 4714, pages 262–270. Springer-Verlag, 2007.
16. F. Gottschalk, W.M.P. van der Aalst, M.H Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *International Journal of Cooperative Information Systems*, 17(2):177–221, 2008.
17. F. Gottschalk, T. Wagemakers, M.H. Jansen-Vullers, W.M.P. van der Aalst, N. Sidorova, and M. La Rosa. Configurable Process Models: Experiences From a Municipality Case Study. In P. van Eck, J. Gordijn, , and R. Wieringa, editors, CAiSE'09, LNCS, vol. 5565, pages 486–500. Springer-Verlag, 2009.
18. C.W. Günther and W.M.P. van der Aalst. Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In G. Alonso, P. Dadam, and M. Rosemann, editors, BPM 2007, LNCS, vol. 4714, pages 328–343. Springer-Verlag, 2007.
19. A. ter Hofstede, W. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its Support Environment*. Springer-Verlag, 2010.
20. C. Li, M. Reichert and A. Wombacher. Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, BPM 2009, LNCS, vol. 5701, pages 344–362. Springer-Verlag, 2009.
21. J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data and Knowledge Engineering*, 64(1):312–329, 2008.
22. H.R. Motahari Nezhad, R. Saint-Paul, B.Benatallah, and F. Casati. Deriving Protocol Models from Imperfect Service Conversation Logs. *IEEE Transactions on Knowledge and Data Engineering*, 20(12):1683–1698, 2008.
23. M. La Rosa, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Questionnaire-based Variability Modeling for System Configuration. *Software and Systems Modeling*, 8(2):251–274, 2009.
24. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.
25. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
26. 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.