Expert Systems with Applications 41 (2014) 7291–7306

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Merging event logs for process mining: A rule based merging method and rule suggestion algorithm

Jan Claes *, Geert Poels

Ghent University, Department of Business Informatics and Operations Management, Tweekerkenstraat 2, 9000 Gent, Belgium

ARTICLE INFO

Article history:
Available online 11 June 2014

Keywords:
Business process management
Process mining
Event log merging
Inter-organizational process modeling

ABSTRACT

In an inter-organizational setting the manual construction of process models is challenging because the different people involved have to put together their partial knowledge about the overall process. Process mining, an automated technique to discover and analyze process models, can facilitate the construction of inter-organizational process models. This paper presents a technique to merge the input data of the different partners of an inter-organizational process in order to serve as input for process mining algorithms. The technique consists of a method for configuring and executing the merge and an algorithm that searches for links between the data of the different partners and that suggests rules to the user on how to merge the data. Tool support is provided in the open source process mining framework ProM. The method and the algorithm are tested using two artificial and three real life datasets that confirm their effectiveness and efficiency.

1. Introduction

The awareness that organizations, in their attempts to optimize business processes, have to look beyond their organizational boundaries, exists in academia (Håkansson & Snehota, 1989; Legner & Wende, 2007) and in practice (Bernabucci, 2008; Grefen, Eshuis, Mehandjiev, Kouvas, & Weichhart, 2009). Numerous inter-organizational integration efforts are supported by the construction and analysis of business process models (Bolstorff & Rosenbaum, 2008; Ghattas & Soffer, 2009; Min & Zhou, 2002) and have proven their value (Höfferer, 2007; Van der Aalst, 1999a; Van der Aalst, 1999b; Van der Aalst, 2000).

When different organizations construct process models of their joint operations, this is called inter-organizational process modeling (Bouchbout & Alimazighi, 2011). In theory, there are many benefits to the alignment of processes across organizational boundaries (Bernabucci, 2008; Bolstorff & Rosenbaum, 2008; Cohen & Roussel, 2005; Håkansson & Snehota, 1989; Stadtler, 2008; Yu, Yan, & Cheng, 2001). Therefore, one would expect organizations to jointly organize their process at a large scale. However, in practice, it rarely happens that business partners share highly strategic data (Simatupang & Sridharan, 2001). Nevertheless, there are several cases where inter-organizational process modeling occurs in reality: (i) between organizations that are not competing (e.g., government, non-profit, healthcare), and (ii) for decentralized end-to-end processes within organizations (e.g., multinationals, shared service centers, multiple labels) (Hoogland, 2012).

The manual modeling of business processes is, however, time-consuming and prone to subjective decision-making, which provides good reasons to adopt automated modeling techniques such as process mining (Rozinat, Mans, Song, & Van der Aalst, 2009; Van der Aalst, 2008; Van der Aalst et al., 2003). Process mining makes use of recorded historical process data in the form of so called event logs to discover and analyze as-is process models (Van der Aalst, 2011). In an inter-organizational setting, these data are distributed over different sources, which each encompass partial information about the overall business processes. Currently available process mining techniques (e.g., Heuristics Miner (Weijters & Van der Aalst, 2006), Fuzzy Miner (Günther & Van der Aalst, 2007), Dotted Chart Analysis (Schonenberg, Weber, Van Dongen, & Van der Aalst, 2008), Conformance Checking (Rozinat & Van der Aalst, 2008), Social Network Mining (Van der Aalst & Song, 2004)[1]) require these data first to be merged into one structured dataset.

The recorded historical process data that are used as input for process mining techniques can be merged at three levels: raw data level (i.e., merging databases and/or files), structured data level (i.e., merging event logs), and model level (i.e., merging process

* Corresponding author. Tel.: +32 9 264 98 31; fax: +32 9 264 42 86.
E-mail addresses: jan.claes@ugent.be (J. Claes), geert.poels@ugent.be (G. Poels).

[1] These are the five most used process mining plug-ins in ProM 6 according to (Claes & Poels, 2013).

http://dx.doi.org/10.1016/j.eswa.2014.06.012
0957-4174/

models). When merging at structured data level, the individual partners of the inter-organizational process are responsible for selecting and structuring the data from their own information systems and by consequence in choosing the appropriate abstraction level and viewpoint of that part of the data. In case the data is merged at raw data level, the operation of structuring these raw data would form a bigger challenge, because then also the required knowledge about the meaning of and relations between the data should be brought together. On the other hand, the choice of a specific process mining technique can be postponed, because the result of merging the data at structured data level is an event log on which all existing process mining techniques (that use an event log as input) can still be applied. In the case of a model level merge, postponing the choice of mining technique to be employed would be impossible. Therefore, in the context of inter-organizational process modeling, the structured data level seems to be the appropriate level for merging the data.

There are, however, to date no techniques for merging the data at structured data level. In practice, process mining users nowadays turn to merging techniques at raw data level or at model level, but they report serious shortcomings in this way of working (Claes & Poels, 2013). Moreover, the shortage of suitable *tool support* for merging event logs (regardless the merge level) appears to be confirmed by a recent survey about the perception of the utility and usability of process mining techniques and tools (Claes & Poels, 2013).

Therefore, the research question that forms the basis for the research presented in this paper, is how to merge the data at structured data level of partners involved in an inter-organizational (business) process in such a way that the benefits of automated process mining are preserved (i.e., speed and objectivity). The significance of the research question is demonstrated by the recently published process mining community manifesto (Van der Aalst et al., 2011). In this manifesto researchers and practitioners in the field of process mining listed eleven important open challenges that need to be addressed to increase the applicability of process mining techniques. Challenge 1 is the merging (and finding and cleaning) of event data. Such merging is required for cross-organizational process mining, which is pointed out in challenge 7 as another important area of research. Two types of cross-organizational process mining are identified: (i) organizations collaborate to handle process instances together and (ii) different organizations execute the same process while sharing experiences, knowledge, or a common infrastructure (Van der Aalst et al., 2011). The contribution of this paper is a new method and algorithm with tool support to merge event logs (i.e., at structured data level) of different sources in support of inter-organizational process modeling.

The merging method that we designed to address the research question comprises two consecutive steps for the merge of two event logs:

(i) discover links between the two event logs to indicate which data in both event logs are considered to belong to the same process instance, and
(ii) based on the configured links, represented by merging rules, merge the data of both event logs to form a new event log.

The research described in this paper was performed using a design science approach (Hevner, March, Park, & Ram, 2004). Hevner et al. define design science research as "*research through the building and evaluation of artifacts designed to meet the identified business need*" (Hevner et al., 2004). The artifacts that are the subject of this paper are (i) an event log merging *method,* (ii) a merging rule suggestion *algorithm*, and (iii) their *implementation* in the process mining tool ProM. The algorithm searches for possible links between event logs and suggests merging rules to the user to support the merging method. Although in theory the method and its implementation (and the algorithm and its implementation) form separate artifacts, the large amount of involved data makes it practically impossible to treat them as separate artifacts that would have to be developed and evaluated separately. Hence, the evaluation of the method and algorithm is performed based on their implementations, i.e., the evaluation concerns the *implemented method* and the *implemented algorithm*.

The knowledge base on which our research is grounded is provided in Section 2. Methodical guidance for our research activities was found in the Design Science Research Methodology (DSRM) for Information System Research (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). This methodology requests the completion of six activities (see Fig. 1). The *Identification of the Problem & Motivation* is addressed by a recent process mining survey paper (Claes & Poels, 2013) that demonstrates the need for method and tool support for merging process mining data, which is also explicitly expressed by the process mining community in Challenge 1 and 7 of the process mining manifesto (Van der Aalst et al., 2011). The *Objectives of a Solution* were already defined in this introductory section. In short, the research aims at developing a way to merge data for process mining at structured data level. To be in line with the overall objectives of process mining, the merge of data in the form of event logs – as a preparatory step for process mining – needs to be accomplished quickly and objectively. Therefore, a high degree of automation is desired. The *Design and Development* of the artifacts is explained in Sections 3 and 4. In Section 3, the two steps of the *method* and its implementation in a well-known process mining tool are explained. Section 4 reports on the development of the rule suggestion *algorithm* and its implementation. The *Demonstration* and the *Evaluation* of the artifacts based on two artificial and three real life datasets is the subject of Section 5. The *Communication* of the research is obviously the objective of this paper, earlier iterations were published in previous work (Claes & Poels, 2011a; Claes & Poels, 2011b). Finally, Section 6 provides a discussion and conclusion.

2. Background

2.1. Process mining

The goal of process mining is "to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's systems" (Van der Aalst, 2011, p. 8). Three types of process mining techniques exist: (i) process discovery, (ii) process conformance, and (iii) process enhancement (Van der Aalst, 2011). Process discovery is concerned about how to construct a process model based on historical process data structured in event logs. Process conformance uses historical process data to check for deviations in the process with respect to a given process model. Process enhancement uses the historical process data to project more information on a provided process model (such as durations of and between activities or decision determinants).

An *event log* is a hierarchically structured file with data about historical process executions. Hence, this file contains data about several (possibly different) executions of the same process, which are used by process mining techniques. Mostly, this file has to be constructed by structuring raw process data that can be found in files or databases (e.g., SAP Audit Trail), into events and traces.

An *event* is the most atomic part of a specific process execution. Event data can typically be found in information systems under the form of status updates (e.g., from 'invoice sent' to 'invoice paid') or activity records (e.g., 'mail sent to customer'). Besides a name, events can have several other attributes to indicate for example

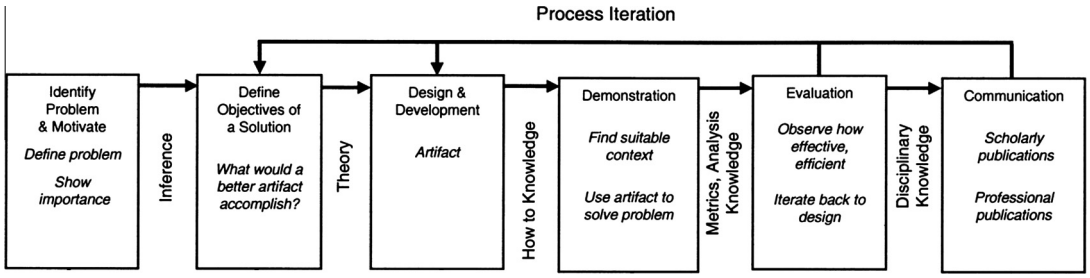


Fig. 1. DSRM process model (adapted from Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007).

the specific timestamp associated with the event, the originator of the event, etc.

A *trace* is a collection of events that belong to the same process execution. For example, all recorded data of accepting and processing a sales order could constitute a trace. Different traces in the event log then accord to different orders for which the process was executed. Traces can also have other attributes (other than a trace identifier): for example an order type (e.g., off-line or online order) can accompany the order id that is used as trace identifier. Table 1 and Extract 1 show an example of the recorded execution data of a sales order processing process, for which five data fields were extracted from a database. *Sales order* is used as trace identifier; *Operation* provides the name for events. Furthermore, the *Date*, *Status* and *User* (i.e., originator) of the operation are stored as attributes of the event.

When constructing an event log, decisions have to be made about which data to include in the analysis (e.g., for performance analysis it is useful to include both start and end time of the events), about the level of detail of events (e.g., several status updates might be combined to result in only one event record in the constructed log), about the level of detail of traces (e.g., information can be grouped in traces per customer or per sales order), etc. The decisions made in this preparation phase might heavily effect the result of the analysis (Van der Aalst, 2011, pp. 95–98).

2.2. Merging historical process data

Merging historical process data at the *structured data level* has not been researched yet. Therefore, it is not possible to compare our solution with similar work. Literature about merging historical process data in preparation of process mining projects is also scarce.

Gerke et al. used a technique at *raw data level*. They examined how RFID data can be used to merge data of supply chain partners (Gerke, Mendling, & Tarmyshov, 2009). The focus is on data that accord to the EPCglobal standard, which makes it possible to merge data of different partners through the mapping onto the concepts of the common standard. At raw data level, the term extract-transform-load (ETL) is used for techniques that take information from a system, transform it to the structure of a new system, and load it into the new system (Agrawal, Chafle, Goyal, Mittal, & Mukherjea, 2008). This way, data from different systems can be merged into the new system.

Table 1
Unstructured historical process data.

Sales Order	Operation	Date	Status	User
SO1-CustA	Order created	2012-9-28 9:50	Complete	Geert
SO1-CustA	Goods delivered	2012-10-1 8:15	Complete	Geert
SO1-CustA	Invoice sent	2012-10-1 10:30	Complete	Jan
SO2-CustA	Order created	2012-9-30 11:05	Complete	Jan
SO2-CustA	Information requested	2012-10-4 10:20	Complete	Jan
SO1-CustB	Order created	2012-10-2 10:25	Complete	Geert

These techniques require to transform the data based on a common standard in order to allow for putting two datasets in the same format in one system. They do not provide support to merge the data that describe complementary properties of the same process execution, as it is needed for inter-organizational process mining. In this context it is not sufficient to put the traces of both event logs together in one aggregated event log, but support is needed to determine which traces need to be merged individually (because they comprise events that belong to the same process execution instance). This support is comprised in the rule based merging method presented in this paper.

La Rosa et al. describe the merge of process data at *process model level* as one of two cases: merged models and digests (La Rosa, Dumas, Uba, & Dijkman, 2013). Merged models are the result of merging different process models that contain different variants of the same process. Digests are defined as a summary of a collection of process models focusing on recurring fragments. The merge of process models can be facilitated by first detecting possible merge points: the nodes where the to be merged process models connect (Sun, Kumar, & Yen, 2006). It is argued that, in order to assure a sound merge, "*one should choose merge points between which a sub-workflow is well structured*" (Sun et al., 2006, p. 844). Li et al. describe a clustering approach for process variants that searches for a process model for which average distance with the variants is minimized. This process model is then presented as the reference model for the process aware information system that supports the process (Li, Reichert, & Wombacher, 2010). Finally, Küster et al. present a tool for detecting and merging changes between different versions of the same process model (Küster, Gerth, Förster, & Engels, 2008).

These techniques are very useful for the merge of process models, but they are less suitable for process mining projects. In such projects various analyses are often performed successively (e.g., using Heuristics Miner to discover the whole process, than using the Fuzzy Miner to examine most frequent behavior, next using LTL Checker to reveal when some deviations occur, etc.) The process model level merge approach would require to merge the result of any individual analysis (and requires a different implementation for the different process model notations used in these analyses). By merging the data at structured data level, as in the proposed rule based merging method, the merge can be performed once; and then the various techniques can be performed sequentially on the merged dataset without further effort.

3. Rule based merging method

This section reports on the design of the rule based merging method (Section 3.1) and its implementation in ProM (Section 3.2).

3.1. Method

The intention of the merging method is to merge historical process data at structured data level. The method can be used to

```
<log xes.version="1.0" xmlns="http://www.xes-standard.org">
  ...
  <trace>
    <string key="concept:name" value="SO1-CustA"/>
    <event>
      <string key="concept:name" value="Order created"/>
      <string key="lifecycle:transition" value="complete"/>
      <string key="org:resource" value="Geert"/>
      <date key="time:timestamp" value="2012-09-28T09:50"/>
    </event>
    <event>
      <string key="concept:name" value="Goods delivered "/>
      ...
```

Extract 1. An event log: structured historical process data.

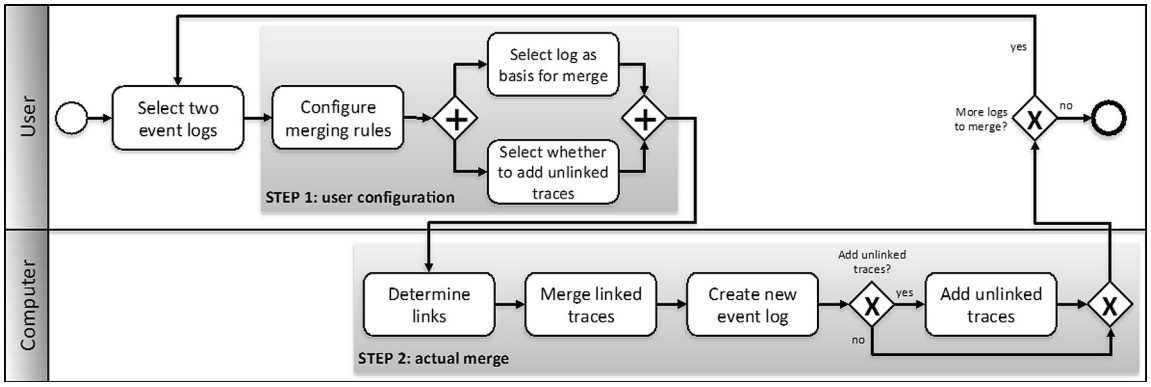


Fig. 2. Steps of the rule based merging method.

merge two event logs at a time. When more event logs need to be merged, one could merge the first two and then, one by one, merge the remaining event logs with the aggregated event log.

The method encompasses two consecutive steps (see Fig. 2). In the first step, a user has to decide *how* both event logs will be merged. The actual merge forms the second step. The benefit of this two-step approach is that the user is in control of the merging process and the merging rules are made explicit beforehand. This increases the transparency of the solution (compared to earlier solutions that ran automatically without feedback about the merge conditions to the user (Claes & Poels, 2011a; Claes & Poels, 2011b)).

In the first step, links between traces of both event logs need to be defined. A *link* between two traces indicates that the two traces are considered to contain information of events of the same process execution. The construction of specific links between traces is the result of the application of merging rules. A *merging rule* is a more general statement declaring the criteria which two traces of different event logs must meet in order to be considered linked traces. These criteria are formulated in terms of relations between attributes of the traces and/or attributes of the events of the traces (see Section 3.1.1).

In the second step, based on the configured merging rules, linked traces are merged into one trace containing the event information of both particular traces. A new event log is created, containing the new, merged traces and optionally also all unlinked original traces from the two event logs (see Section 3.1.2).

3.1.1. Configure merging rules to define links between traces in two event logs

The first step of the method is to configure merging rules. A merging rule is a general statement to indicate which properties of two traces have to be related to be considered traces for the same process execution. In the second step, a link will be created between all pairs of traces of two logs that meet the criterion specified in the merging rule.

While the *structure* of an event log is determined by its division in traces and events, the actual *data* is comprised in the attributes of traces and events. Therefore, merging rules are formulated in terms of *relations* between *attribute values* at a certain *level* (i.e., event and/or trace) in the two event logs.

In our implementation, a merging rule has the following form:

```
Merge all traces where
<select attribute> of <select attribute container>
  in log 1
<select operator>
<select attribute> of <select attribute container>
  in log 2
```

For example: "Merge all traces where <id> of <the trace> in log 1 <equals><reference number> of <event Send Invoice in the trace> in log 2".

There are three types of input that are required to be specified in a merging rule:

- `<select attribute>`: the user can select an attribute name out of a list that contains all present attribute names of traces or events in the specific event log (e.g., "id", "reference number")
- `<select attribute container>`: the user can select an attribute container out of a list that contains
  - o 'the trace',
  - o 'any event in the trace',

- o 'the trace or any event in the trace'.
- o Besides these three options the list is complemented with all present event names in the form 'event X in the trace', where X is replaced by the specific event name.

- `<select operator>`: the user can select a relational operator out of a list that contains:
  - o 'equals', 'is not equal to',
  - o 'is lower than', 'is lower than or equal to',
  - o 'is greater than', 'is greater than or equal to',
  - o 'contains', 'does not contain'.

Different merging rules can be combined with unprioritized 'and' and 'or'. Practically, this means the result of evaluating the first rule is combined with the evaluation result of the second rule using the selected logical operator between first and second rule (i.e., 'and' or 'or'). Next, the result is combined with the evaluation result of the third rule using the selected logical operator between second and third rule, etc.

The method itself does not provide guidance for the selection of merging rules. It is up to the user to gain enough knowledge about the two event logs to know how to link both event logs. However, the method is complemented with a rule suggestion algorithm to support users that lack the requested knowledge (see Section 4).

3.1.2. Merge linked traces based on the configured merging rules

The set of (general) merging rules is then used to construct a set of (specific) links between traces of both event logs. Each possible pair of traces consisting of a trace of the first event log and a trace of the second event log is evaluated based on the merging rules set. Only if the result of this evaluation is *true*, a link between the two traces is added to the set of links (i.e., the information in these traces is considered to belong to the same process execution).

In order to determine the level of detail of the resulting event log, one of the event logs is selected to form the basis for the merge. It is left to the user to make this choice. If for example event log 1 is at customer level (i.e., each trace contains events of a different customer) and event log 2 is at customer order level (i.e., each trace contains events of a different customer order), one can choose the level of detail of the resulting event log by appointing event log 1 or log 2 as basis for the merge. This would result in an event log at customer level or at customer order level respectively.

Further, because traces in both event logs have certain attributes with equal names (but different values) and because each attribute of a certain trace must have a unique name, the attributes of two traces to be merged cannot simply be assigned to the newly formed trace. The attributes of the traces contain the information of the case (i.e., the customer or the customer order in the example). Therefore, it makes sense to only include the attributes of the trace from the event log that serves as basis for the merge into the new trace (i.e., only the data of the customer or of the customer order in the example).

Practically, if event log 1 forms the basis for the merge (for the sake of simplicity, let us call a trace of event log 1 a 'first trace' and a trace of event log 2 a 'second trace'), then

- the new trace as result of the merge of a first trace and a second trace adopts the attributes of the first trace;
- if a first trace is linked to multiple second traces, the merged trace will contain the events of all involved traces;
- if multiple first traces are linked to the same second trace, all involved first traces will be merged with an individual *copy* of the events of that second trace.

Of course, if event log 2 is selected by the user as basis for the merge, the opposite applies.

The actual merge of two or more traces thus consists of (i) constructing a new trace with the attributes of the trace from the event log that forms the basis of the merge, and (ii) adding to this new trace the events and their attributes of all involved traces in chronological order (i.e., the method assumes an accurate and comparable timestamp is present as attribute of each event).

Additionally, for all events two supplementary attributes are added: *origLogID* contains the name of the original event log from where the event was copied into the new event log, and *origTraceID* contains the trace identifier of that event in the original event log. This allows process mining techniques to use or show these attributes and avoids the necessity to change existing attribute values of events to be able to determine the origin of the events.

Finally, the unlinked traces of event log 1 and/or event log 2 can optionally be included in the merged event log. This can be compared to inner and left/right/outer join in a database table join context.

3.2. Implementation

The method was implemented in the well-known academic process mining tool ProM (Van Dongen, De Medeiros, Verbeek, Weijters, & Van der Aalst, 2005). This is an extendable open source framework that allows process mining researchers and developers to implement and test process mining (and related) techniques by developing a plug-in for the tool. The 'Merge two Event Logs using a rule based algorithm' plug-in can be downloaded at our website[2]. It requires two event logs as input and generates the merged event log as output in the framework (see Fig. 3). This means the merged event log can immediately be used as input for all available process mining techniques (that require an event log as input) in the framework.

Fig. 4 shows the parameter settings window of the plug-in for configuring merging rules as part of step 1 of the merging method. The bottom part can be used to add a new rule to the list: It contains the lists of attributes, attribute containers and operators as presented in Section 3.1.1. The middle part contains a list of suggested, preconfigured rules (see Section 4). The top part presents the selected rules as a result of configuring a rule in the bottom part or selecting a rule in the middle part. In the top part these rules can also be combined by changing the order and selecting logical operators.

The other parameter settings are displayed in Fig. 5: The user can select to include all unlinked traces of the first event log, to include all unlinked traces of the second log, and which of both logs to use as basis for the merge.

4. Merging rule suggestion algorithm

This section provides details on the rule suggestion algorithm (Section 4.1) and its implementation (Section 4.2).

4.1. Algorithm

The algorithm helps the user to configure merging rules by suggesting rules to the user. For computational efficacy, it was decided to limit the algorithm to discovering possible merging rules with the 'equals' operator. In our experience with various real life cases (for example see Section 5.2.4 and 5.2.6), it was observed that mostly the configured merging rules use only this operator. Nevertheless, we address the extension of the algorithm to find rules with other operators as future research (see Section 6.3).

[2] Version used in this paper: http://www.janclaes.info/papers/PMMerge/LogMerge.zip Latest version: http://www.janclaes.info/downloads/LogMerge.zip.

Fig. 3. Screenshot of plug-in selector view in ProM.

The algorithm is embedded in the method implementation (see Fig. 6). After selecting two event logs for the merge, the algorithm runs and presents its results to the user. The user can decide to use any of the suggested rules, to use a rule from the suggestion list but change some of its configuration values, or to not use a rule from the list at all.

Every possible merging rule between the two event logs based on the relational operator 'equals' is evaluated by the algorithm and only those rules that would result in linking more than one pair of traces of both event logs are presented. Furthermore, the algorithm sorts the rules according to a score that indicates the assumed probability for finding the correct links between the traces covering data of the same process execution (see Appendix A for a detailed description of the used sort heuristic and score calculation).

The algorithm is composed of three steps:

(i) First, an index is made of attribute values of all attributes in the first log.
(ii) Second, a provisional rule is built for every attribute value in the second log that matches an attribute value of the first log (the rule makes use of the *keys* of the attributes for which the *values* match),
(iii) Third, the rules are sorted according to a given score that specifies the assumed correctness of the rule for merging the two event logs. The sort heuristic assigns a score to each rule that indicates the assumed probability of the rule to be the appropriate rule for merging two event logs correctly. This score is a value between 0 and 1 (see Appendix A).

4.2. Implementation

The rule suggestion list is part of the first configuration screen of the 'Merge two Event Logs using a rule based algorithm' plug-in. The rules are suggested to the user in a sorted list displaying the score of the rule together with a color indication on how good this score is, scaled linearly from red to green according to a score from 0 to 1 (see Fig. 7). The suggested rules are displayed in the middle part of the screen. The user can simply click on the button 'use' to add a suggested rule to the selected rule set.

5. Demonstration and evaluation

In this section seven test scenarios involving two artificially created and three real-life datasets, for demonstration and evaluation of the method (see Section 5.1) and the algorithm (see Section 5.2) are presented. At the end, a brief discussion concludes the findings (see Section 5.3).

5.1. Implemented rule based merging method

5.1.1. Measurement

The rule based merging method is evaluated according to the Method Evaluation Model (MEM) (Moody, 2003). Based on Rescher's epistemic pragmatism (Rescher, 2012), Moody considers a method as having high quality when it is adopted in practice. MEM extends the Technology Acceptance Model (TAM) (Davis, 1989), that proposes to measure *Perceived Usefulness*, *Perceived Ease of Use*, and *Intention to Use*, with three extra variables from Rescher's theory: *Actual Effectiveness*, *Actual Efficiency*, and *Actual Use* (see Fig. 8). Whereas according to TAM, beliefs of usefulness and ease of use are strong determinants of the intention to use a method, the MEM recognizes that these beliefs are at least partly formed by actual experiences with applying the method, as measured by its actual efficacy. Hence, to evaluate the likely adoption of a method, we should measure both actual and perceived efficacy.

Applied to the event log merging method, the Actual Effectiveness of the method (AES_m) is determined by how correctly the two event logs are merged according to the provided merging rule(s). We define the value for this variable as the number of change operations (insertion or deletion) on traces, events or attributes that are needed to convert the merged event log according to the provided

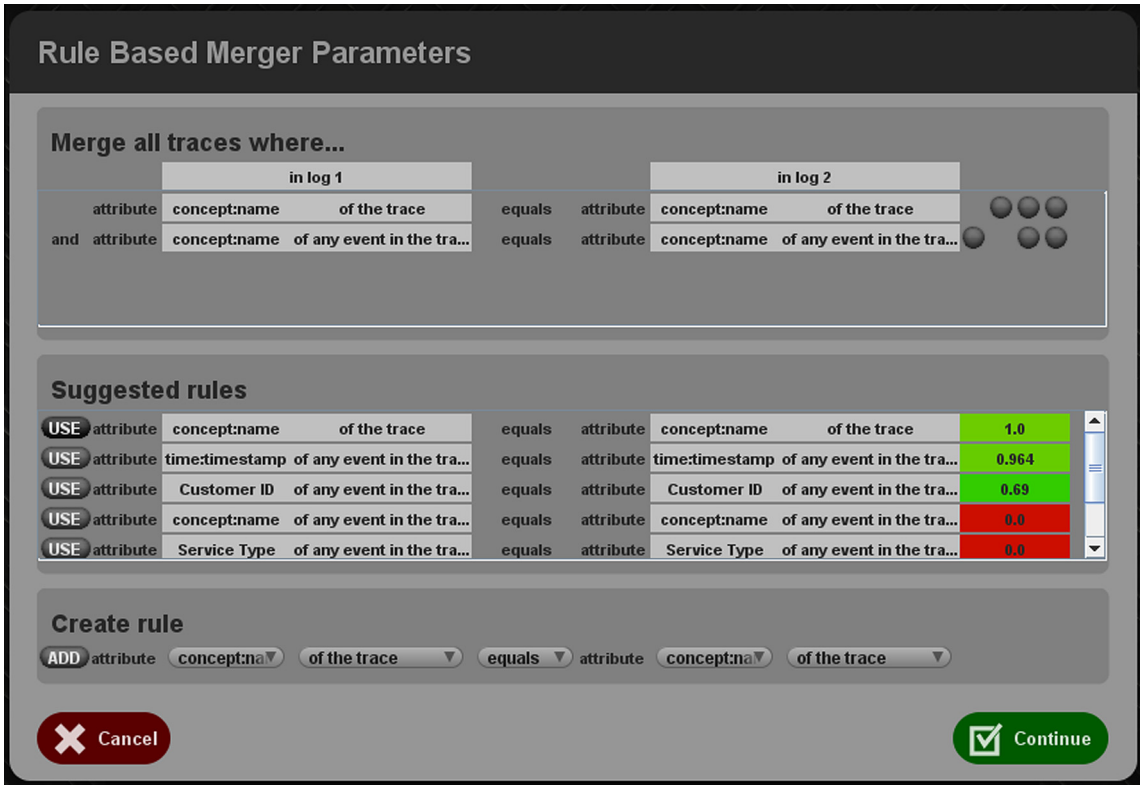


Fig. 4. Screenshot of the rule configurator window in ProM.

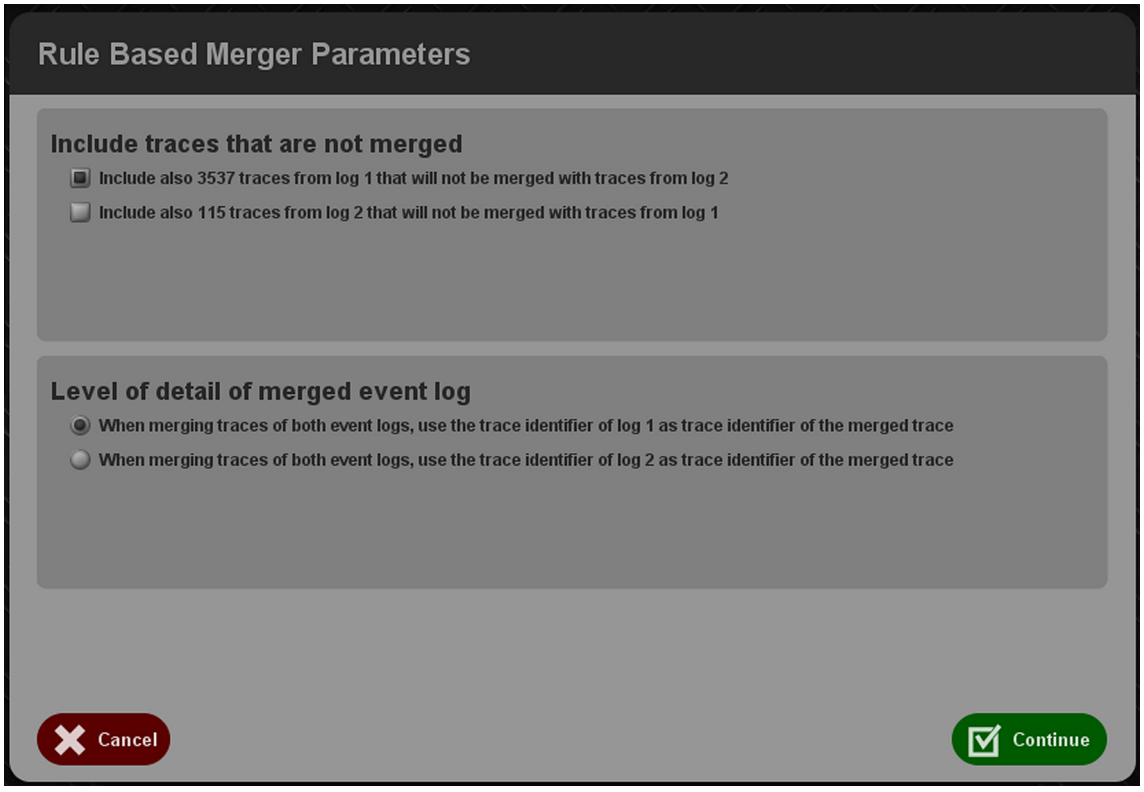


Fig. 5. Screenshot of output parameter window in ProM.

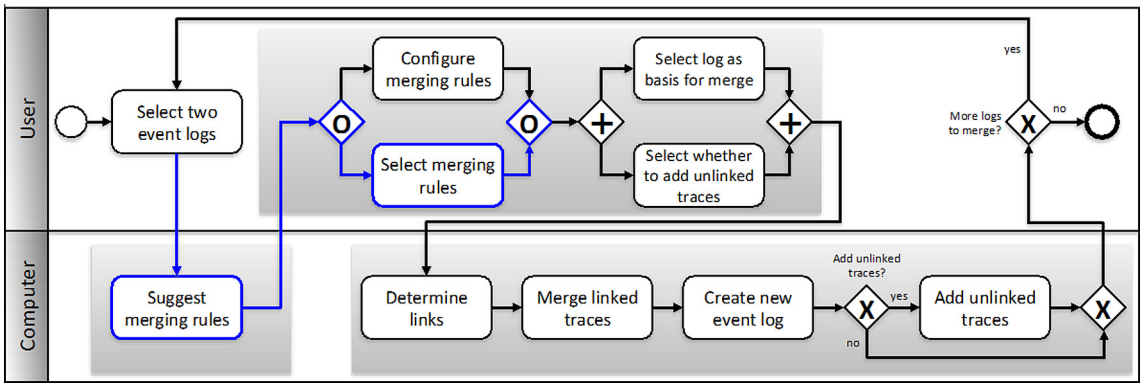


Fig. 6. Steps of the rule based merging method including merging rules suggestion algorithm.

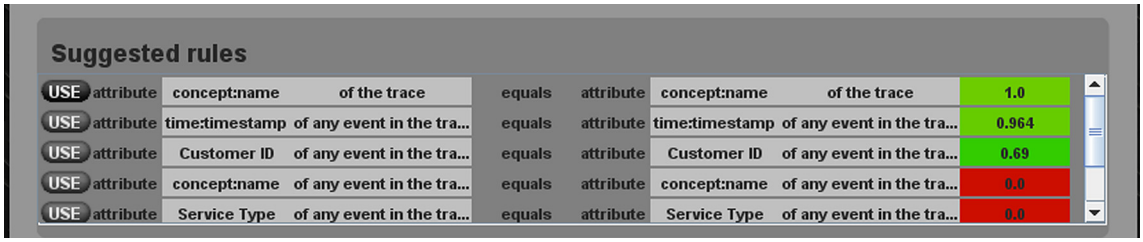


Fig. 7. Screenshot of the rule suggestion panel of the rule configurator window in ProM.

merging rule(s) into the correctly merged event log. The lower this number, the better the actual effectiveness of the method.

The Actual Efficiency of the method (AEy_m) depends on the time, cost and effort to execute the method (Moody, 2003). The time is measured as the time between starting the plug-in in ProM and obtaining the result in the form of a merged event log. It is divided in the time the *computer* takes to build the user interface, to handle the interaction with the user and to merge the event log; and the time the *user* takes to configure the rules and select the appropriate parameters. The cost is zero, because the plug-in is part of an open source project. The effort is measured as the amount of mouse clicks to execute the method as implemented with the ProM plug-in.

Perceived Usefulness (PU_m), Perceived Ease of Use ($PEoU_m$), and Intention to Use (ItU_m) are measured according to the questionnaire presented in Appendix B. The Actual Usage (AU_m) cannot be measured in an experimental setting, but it is offered that the previous variables can be used to predict the actual use of a method (Moody, 2003).

5.1.2. Test scenario 1: artificial case

5.1.2.1. Data. An artificial process description was constructed concerning a process of order handling[3]. It mentions two information systems that support the process. The accounting software contains information about the creation of an order, shipment of goods, sending the invoice and receiving payment. The warehouse software contains the detailed information of the articles on the order, any changes on the order concerning price or amount of ordered articles and price calculation.

Two event logs were constructed as if they were extracted from each of both imaginary systems. The first event log contains information about 6.623 operations (events) on 3.885 sales orders (traces). The second event log describes information about 43.207 operations (events) on 13.102 articles (traces). The sales order identifier, which is used in the first event log as trace id, is included in the attribute 'Sales_Order' of the event 'creation' in each trace of the second event log.

5.1.2.2. Subjects. This test was performed by 6 master students of Business Engineering that have initial experience with process mining techniques and the ProM tool in the preparation of their master thesis. They were provided with (i) a description of the two event logs to be merged, (ii) a description of the ProM plug-in, and (iii) a description of the task to perform. Afterwards, they answered the 16 questions of MEM. We were not able to capture the amount of mouse clicks for the plug-in only, because the students also inspected the event logs during the test.

5.1.2.3. Results. The results are presented in Table 2. In every test, the plug-in succeeded in merging both event logs correctly (i.e., the users selected or configured the correct merging rule and the plug-in used this rule to correctly merge the two event logs). The first step of the method is the configuration by the user, which took between 21 s and 5,4 min. The plug-in took between 4 and 11 s in the second step of the method to merge the two event logs (on a 3,45 GB RAM 2,39 GHz laptop). Every student tended to agree or strongly agree that the plug-in is useful, easy to use, and that they would intend to use it.

5.1.3. Test scenario 2: real life case (Volvo)

5.1.3.1. Data. The data for the second test scenario were provided by Volvo Group. A first dataset provides data about factory orders for the construction of trucks. The second dataset contains customer orders of trucks. It was discovered that the attribute 'ORDERNUMBER' of any event in a trace of the first event log was also displayed in the attribute 'Omnumber' of the event 'Accepted' in the second event log.

5.1.3.2. Subjects. We asked two employees of Volvo IT to use the plug-in to merge the two event logs. They are familiar with process

[3] All the material for this test scenario can be downloaded from http://www.janclaes.info/papers/PMMerge.

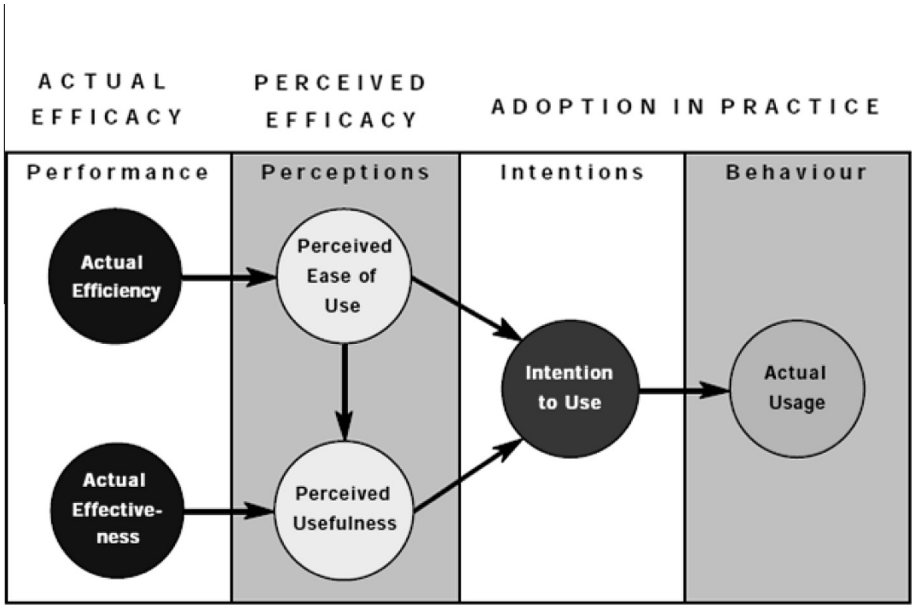


Fig. 8. Variables of the method evaluation model (Moody, 2003).

mining and the ProM tool. Therefore, we only briefly explained the use of the plug-in.

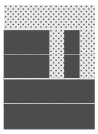
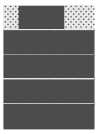
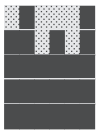
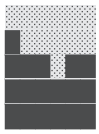
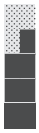
5.1.3.3. Results. During their test, timing and number of mouse clicks were measured. Afterwards they answered the 16 questions of MEM. The results are presented in Table 3. No mistakes were made in the merge of the two event logs (i.e., $AEs_m = 0$). They took less than 2,5 min and less than 22 mouse clicks to configure the plug-in and the actual merge was performed in less than 9 s (on a 3,45 GB RAM 2,39 GHz laptop). Usefulness, ease to use, and intention to use were positively perceived by both participants.

5.2. Implemented merging rule suggestion algorithm

5.2.1. Measurement

For the algorithm, that runs automatically, we only measure effectiveness and efficiency. The algorithm builds a sorted list of suggested rules. The Actual Effectiveness of the algorithm (AEs_a) can only be measured if the correct solution is known (i.e., if the rule set that defines the correct links between both event logs is known). The value for AEs_a is defined as the rank number of the correct rule in the sorted suggestion list. The lower this number, the better. If the correct rule is not available in the suggestion list, the value of AEs_a is set to infinity.

Table 2
Properties and quality measures for test scenario 1: master students use artificial data.

Case properties	Event log 1	Event log 2				
Number of traces	3.885	13.102				
Number of events	6.623	43.207				
Evaluation metrics	Test user 1	Test user 2	Test user 3	Test user 4	Test user 5	Test user 6
Actual effectiveness (AEs_m)	0	0	0	0	0	0
Errors in solution						
Actual efficiency (AEy_m)	6,292 s	5,550 s	4,200 s	5,770 s	10,146 s	3,539 s
Time computerTime user	1,108 min	5,361 min	0,889 min	0,363 min	2,810 min	0,435 min
Effort	*no data*	*no data*	*no data*	*no data*	*no data*	*no data*
Perceived usefulness (PU_m)*						
Strong agree						
Agree						
Neutral						
Disagree						
Strong disagree						
Perceived ease of use ($PEoU_m$)*						
Strong agree						
Agree						
Neutral						
Disagree						
Strong disagree						
Intention to use (ItU_m)*						
Strong agree						
Agree						
Neutral						
Disagree						
Strong disagree						

* Answering '(strongly) disagree' on a negatively formulated question was considered '(strongly) agree' and vice versa.

The Actual Efficiency of the algorithm (AEy_a) can be measured in terms of time, cost and effort. The algorithm runs automatically when starting the plug-in, which is free of charge. Therefore the cost and effort of the implemented algorithm is zero. Hence, the AEy_a is measured only by the time the algorithm ran to construct the suggested rule list. We added some lines of code in the implementation to automatically measure this duration.

5.2.2. Test scenario 1: simulated data

5.2.2.1. Data. The first test uses an artificially generated dataset. The benefit of using simulation is that the correct solution is known (i.e., the content of the merged event log). Another advantage is that properties such as noise can be controlled, whereas in real life data mostly it is not known how many errors exist in the dataset. The model we used in our experiments (see Fig. 9) is based on the same example model as in (Van der Aalst & Weijters, 2004).

Two log files were generated from 100 random executions of the process where the executions of tasks A, E, and F were logged in the first event log and the executions of tasks B, C and D in the second event log. There was no structural unbalance in choosing one or the other path first for the AND-split (B and C) or selecting the path to be followed for the OR-split (A or E). Therefore, the second log ends up with about 50 traces. Time differences between consecutive events were random. Afterwards, more sets of event logs were generated with varying values for number of process executions (i.e., 100, 1.000, 10.000, 100.000) and for noise percentage (i.e., 0%, 10%, 20%, 50%; see Table 4). Noise was introduced in the same way as described in (Weijters & Van der Aalst, 2003). One of four options is selected randomly: (i) delete minimum one and up to one third of events from the beginning of a trace, (ii) from the middle of a trace, (iii) from the end of a trace, or (iv) switch places of two random events in a trace. The noise percentage determines the chance a trace is influenced by noise in one of the four ways.

Table 3
Properties and quality measures for test scenario 2: practitioners use real life data.

Case properties	Event log 1	Event log 2
Number of traces	7.776	3.221
Number of events	24.138	16.848
Evaluation metrics	Test user 1	Test user 2
Actual effectiveness (AEs_m)	0	0
Errors in solution		
Actual efficiency (AEy_m)	8,143 s	8,727 s
Time computer	2,159 min	0,448 min
Time user	19 clicks	21 clicks
Effort		
Perceived usefulness (PU_m)*		
Strong agree		
Agree		
Neutral		
Disagree		
Strong disagree		
Perceived ease of use ($PEoU_m$)*		
Strong agree		
Agree		
Neutral		
Disagree		
Strong disagree		
Intention to use (ItU_m)*		
Strong agree		
Agree		
Neutral		
Disagree		
Strong disagree		

* '(strongly) disagree' on a negatively formulated question was considered '(strongly) agree' and vice versa.

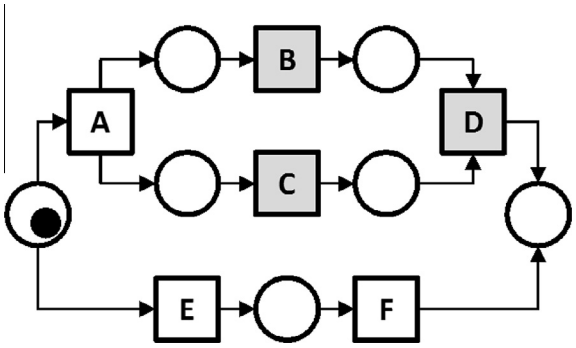


Fig. 9. Process model of the simulated example (from Claes and Poels, 2011b).

5.2.2.2. Results. The results of the tests can be found in Table 4. The score and number of links of the correct rule are shown. Notice that the number of links in the correct solution corresponds with the number of traces in event log 2. This makes sense, since each trace of event log 2 corresponds to a different trace in event log 1 that contains information on the same process execution. Next, it can be observed that the value of AEs_a is 1 for all tests, which means that the correct rule was on top of the suggestion list in every test. Furthermore, the duration of the algorithm stayed under 4 min (on a 3,45 GB RAM 2,39 GHz laptop), even when considering all potential rules with the equals operator between an event log with 100.000 traces and an event log with about 50.000 traces.

5.2.3. Test scenario 2: artificially divided real life data

5.2.3.1. Data. As a second test, the algorithm was applied on a pair of event logs that was artificially constructed out of one single event log from real life[4]. The process data were available in a table format (csv file) and describe the events of a call center support process. The column 'support line' (with value 'frontline' or 'backline') was used to split the event records of the process in two sets to be able to build an event log containing all events of the frontline and a second event log containing all events of the backline.

5.2.3.2. Results. The results are presented in Table 5. Because the algorithm should behave the same way regardless in which order the event logs are provided (the equals operator is symmetric), we provided both event logs in different orders to verify that the result is the same. The score is rather low because of the large difference in number of traces between both event logs. Nevertheless, no merging rule can be made using the equals operator that has a higher score for these two event logs. This illustrates that the value of the score is not important, but the ranking of the rule according to its value separates the better from the worse merging rules (as explained in Appendix A). Note that again the correct merging rule appeared on top of the suggestion list in both cases and the duration remained below two seconds (on a 3,45 GB RAM 2,39 GHz laptop).

5.2.4. Test scenario 3: real life data (Ghent University)

5.2.4.1. Data. The third series of tests was performed using a set of real life event logs that contain partial information about the payroll process at Ghent University for student workers (see Fig. 10). Different users register payroll information in an SAP application (step 1), the data is stored in the SAP database (step 2), data is extracted to transfer files (step 3) that are imported and processed in the old salary calculation system in Oracle (step 4–6) and get

[4] The event log we used is the call center example log from Fluxicon that can be downloaded at http://www.fluxicon.com/disco.

Table 4
Properties and quality measures for test scenario 1 using artificial event logs.

Case	Noise (%)	Num traces1	Num traces2	Score*	Num links	AEs_a	AEy_a
Gen1	0	100	51	0,675	51	1	<1 s
Gen2	10	100	46	0,630	46	1	<1 s
Gen3	20	100	56	0,718	56	1	<1 s
Gen4	50	100	52	0,684	52	1	<1 s
Gen5	0	1.000	480	0,649	480	1	<1 s
Gen6	10	1.000	480	0,649	480	1	<1 s
Gen7	20	1.000	499	0,666	499	1	<1 s
Gen8	50	1.000	505	0,671	505	1	<1 s
Gen9	0	10.000	4.954	0,663	4.954	1	4 s
Gen10	10	10.000	5.002	0,667	5.002	1	3 s
Gen11	20	10.000	5.035	0,670	5.035	1	4 s
Gen12	50	10.000	5.024	0,669	5.024	1	3 s
Gen13	0	100.000	49.969	0,666	44.969	1	2,05 min
Gen14	10	100.000	49.783	0,665	49.783	1	3,27 min
Gen15	20	100.000	50.011	0,667	50.011	1	2,35 min
Gen16	50	100.000	49.957	0,666	49.957	1	2,45 min

* See Appendix A.

Table 5
Properties and quality measures for test scenario 2 using a real life event log that was artificially split into two event logs.

Case	Num traces1	Num traces2	Score*	Num links	AEs_a	AEy_a
FL>BL	3.770	348	0,113	466	1	1 s
BL>FL	348	3.770	0,113	466	1	<1 s

* See Appendix A.

back to the SAP database and applications through other transfer files (step 7–9).

The steps where event logs could be extracted are shown in black circles (i.e., step 2, 3, 5, and 7). The constructed event logs for step 2 (“Audit Trail”), 3 (“Wedde”) and 7 (“Salaris Out”) contain recorded information of 1.032 employees. They can be linked through the trace identifier, which represents the ContractNumber. These event logs are further named Ac, Wc and SOc. Subsequently, the constructed event log for step 5 (“Wedde Berekening”) contains information of 8.242 employees. It has a different trace identifier, i.e., PersonelNumber. We assign the code WBp to the event log. WBp can be linked to the other three event logs because the contract number of each employee is contained in the attribute “Resource” of the events of each trace in the event log.

5.2.4.2. Results. The results are presented in Table 6. The three first rows describe the consecutive merge of the four event logs. First Ac and Wc were merged, which resulted in the creation of event log AcWc. This event log was then merged with WBp to create AcWcWBp, which was ultimately merged with SOc. Because for some event logs we had different versions (e.g., SOc and SOp), we were able to test different variants (see lower part of Table 6).

In 8 of the 9 tests that were performed with these event logs the correct rule was on top of the suggestion list. We examined the case where the correct rule was on rank 2 and we found out that this had to do with two independent attributes that used the same system of ascending numbering, but without the numbers being related. Coincidently matching both attributes provided a higher value for the score and we have to trust upon the user to detect that the proposed attributes in the rule at rank one are not related and therefore that rule should not be selected. All tests took less than 20 s to calculate the merging rule suggestion list (on a 3,45 GB RAM 2,39 GHz laptop).

5.2.5. Test scenario 4: artificial data (student test)

The data from the student tests (see Section 5.1.2) were studied afterwards to evaluate the effectiveness and efficiency of the algorithm. Table 7 encloses the results. The algorithm puts the correct merging rule on top of the suggestion list and it took 5,78 s on average[5] to calculate the suggestion list in the six tests (on a 3,45 GB RAM 2,39 GHz laptop).

5.2.6. Test scenario 5: real life data (Volvo)

Finally, we used the data from the Volvo case (see Section 5.1.3) to evaluate if the algorithm finds the correct rule (according to the rule the employees configured to merge the event logs). The results of this test are presented in Table 8.

Again, the correct rule was presented first in the suggestion list that was constructed after comparing all possible 1.323[5] merging rules using the equals operator. The algorithm took 4,16 s to run on a 3,45 GB RAM 2,39 GHz laptop. In this time 640.482.066[6] attribute containers were compared for each of the 1.323 possible merging rules.

5.3. Discussion of the test results

The artifacts presented in this paper (i.e., the implemented merging method and the implemented rule suggestion algorithm) were developed according to the requirements described in Section 1. The goal was to develop a method and tool support for merging historical process data at structural data level quickly and objectively.

The implemented rules based merging *method* was evaluated in two test scenarios. In both scenarios the method was *effective* (i.e., no errors were made) and *efficient* (i.e., the users took no more than 3 min and used a limited amount of mouse clicks, the actual merge was completed in less than 11 s on a 3,45 GB RAM 2,39 GHz). Moreover, every one of the 8 questioned users agreed that the plug-in is *useful*, *easy to use* and that they *intend to use* it for merging event logs (for these categories each test user answered ‘agree’ or ‘strongly agree’ considerably more than ‘disagree’, no one answered ‘strongly disagree’ on one of the questions)[7].

Five test scenarios were used to evaluate the implemented merging rule suggestion *algorithm*. The scenarios indicate that the algorithm is *effective* (i.e., the correct merging rule was presented first in the list of suggestions in all but one of the 29 cases of the test scenarios) and *efficient* (i.e., the time the algorithm took

[5] (189 different attributes in log 1) × (7 different attributes in log 2) = 1.323 possible rules.

[6] (7.776 cases + 24.138 events) × (3.221 cases + 16.848 events) = 640.482.066 containers.

[7] Answering ‘(strongly) disagree’ on a negatively formulated question was considered ‘(strongly) agree’ and vice versa.

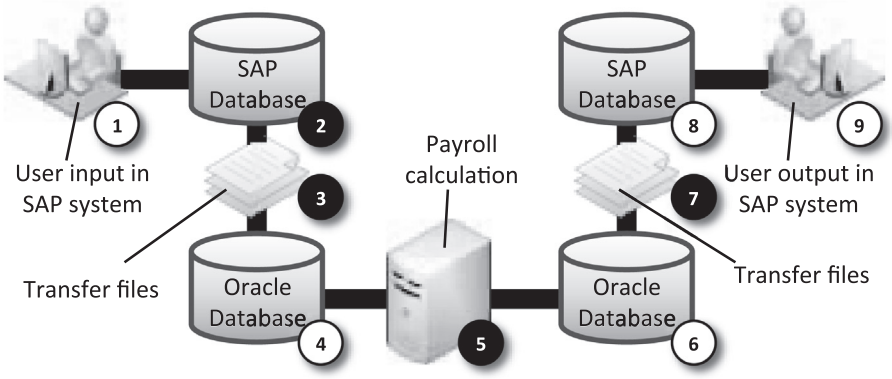


Fig. 10. Main steps in the payroll process (from Claes and Poels, 2011b).

Table 6
Properties and quality measures for test scenario 3 using real life event logs.

Case	Num traces1	Num traces2	Score*	Num links	AEs_a	AEy_a
Ac>Wc	1.032	8.242	0,150	696	1	11 s
AcWc>WBp	8.578	543	0,173	2.064	1	10 s
AcWcWBp>SOc	8.089	1.032	0,119	2.064	1	14 s
Wc>WBp	8.242	543	0,124	1.392	1	3 s
AcWc>WBc	8.578	1.032	0,212	2.064	1	10 s
AcWcWBc>SOp	8.578	3.279	0,364	6.558	1	18 s
AcWcWBp>SOc	8.578	1.032	0,215	2.064	1	16 s
AcWcWBp>SOp	8.089	3.279	0,336	6.558	2	10 s
Wc>WBc	8.242	1.032	0,150	1.392	1	3 s

* See Appendix A.

Table 7
Properties and quality measures for test scenario 4 using real life event logs.

Case	Num traces1	Num traces2	Score*	Num links	AEs_a	AEy_a
Orders	3.885	13.102	0,457	26.204	1	5,78 s[a]

[a] This is the average of the calculation for every student (6,99 s & 4,16 s & 2,72 s & 10,95 s & 3,47 s & 6,36 s).
* See Appendix A.

Table 8
Properties and quality measures for test scenario 5 using real life event logs.

Case	Num traces1	Num traces2	Score*	Num links	AEs_a	AEy_a
Factory orders	7.776	3.221	0,49	5.386	1	4,16 s

* See Appendix A.

to calculate all possible merging rules that use the equals operator was in the order of seconds for event logs with up to 10.000 traces and below 3,5 min for event logs of up to 100.000 traces).

Therefore, in our opinion, the test scenarios demonstrate that the implemented artifacts are quick (i.e., limited CPU time) and objective (i.e., the correct merging rule is presented on top of the suggestion list without user influence).

6. Conclusion

6.1. Discussion

The rule based merging method and rule suggestion algorithm facilitate the merge of historical process data at structured data level, which was a relevant but yet unsolved problem. Both artifacts are implemented in the popular process mining tool ProM in order to make the developed method and algorithm readily available for researchers and practitioners. Test results show a good efficacy, although concerns can be raised about the scalability of the current implementation. The fact that the rule suggestion algorithm only suggests rules with the equals operator is an important drawback, which should be resolved in future research. We believe the two-step approach of the method has resulted in an intuitive user interface of the implementation. It also makes the merge of the event logs transparent to the user, because the merging rules that determine how the event logs will be merged are made explicit (i.e., the user can select or configure the rules in the first step). The ease of use of the implemented method and algorithm is acknowledged by the test users (i.e., five master students and two practitioners).

6.2. Limitations

Current process mining techniques assume the event logs to be "*readily available in today's systems*" (Van der Aalst, 2011, p. 8). In many process mining projects this is not the case. This means the difficulties of selecting the proper data and structuring the data at the proper level of detail to obtain optimal results of process mining analyses, is left out of scope for the process mining techniques ("garbage in is garbage out"). We adopted this assumption, which resulted in two specific characteristics of our technique.

First, if traces in a single event log contain two *different* event records for the same event, existing process discovery techniques will show two activities in the resulting process model for that event. This is no problem, because other techniques can be used to find and correct this situation (i.e., duplicate tasks detection (Li, Liu, & Yang, 2007)). Similarly, if the two event logs to be merged each contain information about the same event execution, using our plug-in will result in a merged event log with two different records for the same event and by consequence two activities for the same event in the resulting process model. The same techniques that are used in other process mining projects can be used to solve this problem. Therefore, the problem of potential overlapping information in both event logs is out of scope of the presented merging method in this paper.

Second, an event log can hold data about a number of events at a high level of detail and of other events at a low level of detail. Existing process discovery techniques do not try to detect this (it might even not be possible to detect this), but the resulting process model will also reflect this difference in level of detail (parts of the process model will contain low level activities, other parts will contain high level activities). Similarly, if the two event logs to

be merged with our plug-in contain information at a different level of detail, this will result in an event log with information at different levels of detail and by consequence in a process model with different levels of detail. The problem of potential differences in level of detail in both event logs is out of scope of the presented merging method. However, the possibility for the user to select the level of detail for the traces for the merged event log (i.e., that of the event log that is chosen as the basis for merge) is included in the method. Further differences in level of detail might be solved by clustering multiple events in the resulting mined process model.

6.3. Future work

In the current implementation of the algorithm, we only consider singular rules based on the equals operator for the suggestion list. Future research will mainly focus on investigating the possibilities of more complex merging rules and an algorithm implementation that searches also for rules with other relational operators or for combined rules. Additionally, the speed and scalability of the method and algorithm need to be improved further by optimizing the program code for the rule discovery and the actual merge.

It should be evaluated how the support for merging event logs can be further improved. For example, it can be examined if the limitations mentioned in Section 6.2 can be resolved or avoided (e.g., support for aligning the abstraction levels of both event logs from within the merge plug-in).

Other future work includes the development for adequate solutions for remaining challenges listed in the process mining manifesto (Van der Aalst et al., 2011).

6.4. Contribution

The research question that forms the basis for the research presented in this paper, is how to merge the data of partners involved in an inter-organizational (business) process in such a way that the benefits of automated process mining are preserved (i.e., speed and objectivity). In the context of inter-organizational process modeling, the appropriate level to merge these data is at the structured data (i.e., event log) level. This way, the individual partners are responsible for selecting the data, the abstraction level, and for structuring the data (which is less convenient at the level of the raw data as recorded in databases or files). On the other hand, all existing process mining techniques can still be used on the data (which is not possible if data is merged at process model level). Up until now, no techniques existed that support the merge of historical process data at structured data level.

We developed a rule based method for merging two event logs at a time, that we implemented in existing process mining software. A merging rule states what conditions between attributes of traces or events in traces in both event logs need to be met to be considered traces for the same process execution. Next, we extended the method with a rule suggestion algorithm that accompanies our implementation to suggest rules that can be used to merge the event logs. The algorithm only discovers potential merging rules that specify that a specific attribute in event log 1 has to be equal to a specific attribute in event log 2 (i.e., only rules with the 'equals' operator). Note, that the described method and algorithm abstract from the inter-organizational context. Therefore, the contributions of this paper can also be used to merge event logs that contain data from two sources within one organization.

Finally, various test scenarios were enrolled to evaluate the implementation of the method and the algorithm. The implemented method was received well by a selection of 6 master students and 2 practitioners, and the calculations in the test scenarios show positive results for effectiveness and efficiency. Similarly, the algorithm succeeded to suggest a proper merging rule at rank 1 or 2 in all test scenarios, and at a fair amount of time. It was not possible to compare our results with existing event log merging techniques because we did not find any of these techniques for the structured data level. We also did not compare with database or model merge techniques, because these techniques are not optimal for inter-organizational process modeling and differ too much in their approach.

Appendix A. Sort heuristic for merging rules

The algorithm presents every possible merging rule to the user that uses the 'equals' operator and that would result in more than one pair of traces to be linked. In order to guide the user in selecting an appropriate merging rule, the list is sorted according to a heuristic that determines the probability of each rule to be the most appropriate merging rule for merging the two event logs. The assumption taken is that a rule which comes closest to a one to one match of traces of both event logs is the correct merging rule. In this attachment, first the assumption is explained and next the heuristic for calculating the probability factor score is presented.

A.1. Assumption taken

Suppose two event logs with 10.000 traces each need to be merged (i.e., they contain partial information about the same process). We assume that a merging rule that links 0 traces from event log 1 to 0 traces from event log 2 is incorrect. In that case, no traces will be merged. Similarly, we assume that a merging rule that links every trace from event log 1 with every trace from event log 2 is also incorrect. In that case, all traces will be merged, resulting in a new log with only one trace that contains all the data. Therefore, we take the explicit assumption that, for this example, a one to one match might be the best choice. If a rule can be built that results in linking every trace of event log 1 with a different trace of event log 2 (i.e., one to one match), it makes sense to assume that this is the correct merging rule (i.e., we assume the chance that this would happen coincidentally is lower than the chance that this is caused by a structural accordance between the two event logs). If no such rule exist, we assume that the rule that results in a match closest to one to one is most probably the right merging rule for the two event logs.

When the two event logs have a different amount of traces, the situation is more complex. Suppose event log 1 has 10.000 traces (e.g., one per customer) and event log 2 has 20.000 traces (e.g., one per customer order). We assume that the merging rule that results in a set of links that is closest to a one to one match is the best possible rule. Notice that we again do not claim that a rule with a one to one match exists, but that the rule that approximates most a one to one match should be on top of the list. In the example this is a rule that links every of the 20.000 traces of event log 2 with one trace of event log 1. Furthermore, we assume that the traces of event log 2 should not all be linked to the same trace of event log 1.

A.2. Calculation of the score

The score for a merging rule is calculated as the product of two factors. The first factor rates the *amount* of links between the two event logs that result from applying the merging rule. The second factor rates the *spread* of these links. The factor score of the rule is then calculated as the product of both factors.

The **first factor** is calculated based on the *average amount of links per trace*. In the previous example we illustrated that for the merge of an event log with 10.000 traces with an event log with 20.000 traces, the desired amount of links is either 10.000 or 20.000 (and the total number of traces in the two event logs is 30.000). Factor 1 is determined in such a way that a score between 0 and 1 is calculated according to next three rules:

(i) if the average is between 0 and 1, this number is used as value for the factor;
(ii) if the average is higher than a threshold t, the value for the factor is set to 0;
(iii) if the average is between 1 and threshold t, the value for the factor is the result of a linear rescaling to a number between 1 and 0 respectively.

$$f1 = f\left(\frac{2 \times number\ of\ links}{number\ of\ traces}\right) \tag{A.1}$$

The threshold t is set to 2 in the current implementation. As can be mathematically proven, with this threshold value, a rule that links all traces from the first event log with one trace from the second event log gets the same score as a rule that links all traces from the second event log with one trace from the first event.

Fig. A.1 shows six examples. The traces are represented by circles. The circles in the left column correspond to event log 1 and the right column corresponds to event log 2. In Fig. A.1a no links are formed and according to previous rules this results in a value of 0 for factor 1. The example in Fig. A.1b shows a single link between one trace in each event log. From the total of 6 traces, 2 traces have 1 link and 4 traces have 0 links, which results in a value for factor 1 of 0,33. Also for the example in Fig. A.1c rule (i) applies and it can easily be calculated that there are on average 0,67 links for each trace. For the example in Fig. A.1d rule (iii) applies. On average 1,33 links per trace exist which results in a value for factor 1 of 0,67. The average of 1,67 links per trace in Fig. A.1e explains the value of 0,33 for factor 1 and according to rule (ii) the average of 2,67 links per trace of Fig. A.1f corresponds with a value of 0 for factor 1. Note that for this example there is no possible solution for which factor 1 is equal to 1, but the solutions with the highest value for factor 1 (i.e., the examples of Figs. A.1c and d) are closest to a one to one match as described above.

The **second factor** is the *number of linked traces related to the total number of traces*. A trace is considered a linked trace when it is involved in at least one link with a trace from the other event log. The value of factor 2 is by definition a number between 0 and 1. Fig. A.2 shows two examples for which factor 1 has the same value (i.e., 0,67). However, the situation in Fig. A.2a is preferred above the example of Fig. A.2b, because of the spread of the links. In Fig. A.2a every trace is a linked trace and therefore the value of factor 2 is 1. Fig. A.2b contains an example where only 4 of the 6 traces are linked, which results in a value for factor 2 of 0,67.

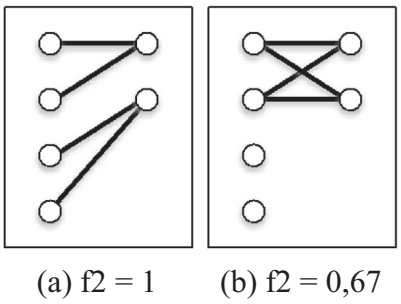


Fig. A.2. Different scenarios of link sets and according value of factor 2.

$$f2 = \frac{number\ of\ linked\ traces}{number\ of\ traces} \tag{A.2}$$

To conclude, the aggregated score of a certain merging rule is calculated as the product of factor 1 and factor 2. Because both factors are a number between 0 and 1 the aggregated factor score also ranges from 0 to 1.Note, that from the example shown in Figs. A.1 and A.2 it can be observed that the example of Fig. A.1d, which is also represented in Fig. A.2a is considered the best solution. However, theoretically it might be possible to construct a set of links with a higher aggregated factor score.

$$score = f1 \times f2 \tag{A.3}$$

Appendix B. Questionnaire

Next questionnaire was used for the assessment of Perceived Usefulness (PU), Perceived Ease of Use (PEoU) and Intention to Use (ItU) according to MEM (Moody, 2003). For each item we used a 5 point Likert scale ('strongly disagree', 'disagree', 'neutral', 'agree', 'strongly agree').

Q1. I found the procedure for applying the method complex and difficult to follow (PEoU1).
Q2. I believe that this method would reduce the effort required to merge event logs (PU1).
Q3. Merging event logs using this method would be more difficult for users to understand (PU4).
Q4. Overall, I found the method difficult to use (PEoU2).
Q5. This method would make it easier for users to bring together process mining data of different sources (PU4).
Q6. I found the method easy to learn (PEoU3).
Q7. Overall, I found the method to be useful (PU4).
Q8. Using this method would make it more difficult to decide which data should be used to determine how both event logs should be merged (PU5).

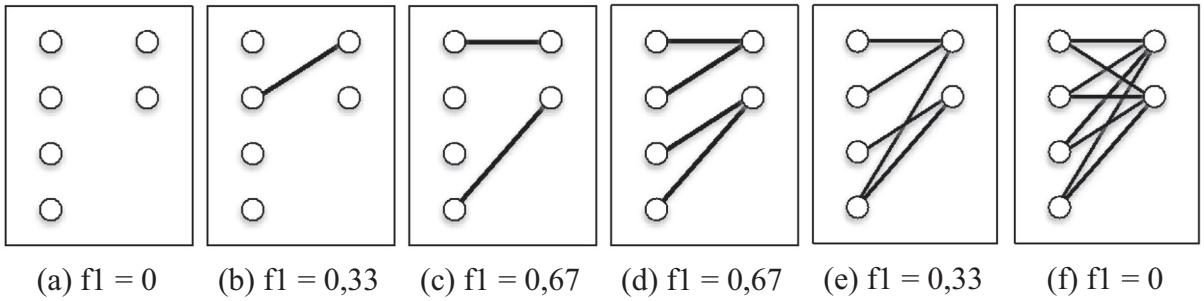


Fig. A.1. Different scenarios of link sets and according value of factor 1.

Q9. I found it difficult to apply the method to the example data (PEoU3).
Q10. I would definitely not use this method to merge event logs for process mining (ItU1).
Q11. I found the rules of the method clear and easy to understand (PEoU5).
Q12. Overall, I think this method does not provide an effective solution to the problem of merging event logs (PU6).
Q13. Using this method would make it easier to merge event logs quickly (PU4).
Q14. I am not confident that I am now competent to apply this method in practice (PEoU6).
Q15. Overall, I think this method is an improvement to the process mining tool set (PU5).
Q16. I intend to use this plug-in in preference to other techniques I know for merging event logs (e.g., using Excel functions) (ITU1).

References

Agrawal, H., Chafle, G., Goyal, S., Mittal, S., & Mukherjea, S. (2008). An enhanced extract-transform-load system for migrating data in Telecom billing. In G. Alonso, J. A. Blakeley, & A. L. P. Chen (Eds.), *Proceedings of the 2008 IEEE 24th international conference on data engineering (ICDE '08)* (pp. 1277–1286). Cancún, México: IEEE Computer Society.

Bernabucci, J. R. (2008). Supply chain gains from integration. *Financial Executive, 24*(3), 46–48.

Bolstorff, P., & Rosenbaum, R. (2008). *Supply chain excellence. A handbook for dramatic improvement using the SCOR model* (2nd ed.). AMACOM Div American Mgmt Assn.

Bouchbout, K., & Alimazighi, Z. (2011). Inter-organizational business processes modelling framework. In J. Eder, M. Bielikova, & A. M. Tjoa (Eds.), *Proceedings II of the 15th East-European conference on advances in databases and information systems (ADBIS '11)* (pp. 45–54). Vienna, Australia: CEUR-WS.

Claes, J., & Poels, G. (2011b). Merging computer log files for process mining: An artificial immune system technique. In F. Daniel, K. Barkaoui, & S. Dustdar (Eds.), *Proceedings of the 9th international conference on business process management workshops (BPM '11), LNBIP 99* (pp. 99–110). Clermont-Ferrand, France: Springer, Berlin Heidelberg.

Claes, J., & Poels, G. (2013). Process mining and the ProM framework: An exploratory survey. In M. La Rosa & P. Soffer (Eds.), *Proceedings of the 10th international conference on business process management workshop/s (BPM '12), LNBIP 132* (pp. 187–198). Tallinn, Estonia: Springer.

Claes, J., & Poels, G. (2011a). Integrating computer log files for process mining: A genetic algorithm inspired technique. In C. Salinesi & O. Pastor (Eds.), *Proceedings of the 23rd international conference on advanced information systems engineering workshops (CAiSE '11), LNBIP 83* (pp. 282–293). London, UK: Springer, Berlin Heidelberg.

Cohen, S., & Roussel, J. (2005). *Strategic supply chain management: The five disciplines for top performance*. McGraw-Hill (p. 316).

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly, 13*(3), 319–340.

Gerke, K., Mendling, J., & Tarmyshov, K. (2009). Case construction for mining supply chain processes. In W. Abramowicz (Ed.), *Proceedings of the 12th international conference on business information systems (BIS '09)* (pp. 181–192). Poznań, Poland: Springer, Berlin Heidelberg.

Ghattas, J., & Soffer, P. (2009). Evaluation of inter-organizational business process solutions: A conceptual model-based approach. *Information Systems Frontiers, 11*(3), 273–291.

Grefen, P., Eshuis, R., Mehandjiev, N., Kouvas, G., & Weichhart, G. (2009). Internet-based support for process-oriented instant virtual enterprises. *IEEE Internet Computing, 13*(6), 65–73.

Günther, C. W., & Van der Aalst, W. M. P. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In G. Alonso, P. Dadam, & M. Rosemann (Eds.), *Proceedings of the 5th international conference on business process management (BPM '07), LNCS 4714* (pp. 328–343). Brisbane, Australia: Springer, Berlin Heidelberg.

Håkansson, H., & Snehota, I. (1989). No business is an island: The network concept of business strategy. *Scandinavian Journal of Management, 5*(3), 187–200.

Hevner, A. R. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75–105.

Höfferer, P. (2007). Achieving business process model interoperability using metamodels and ontologies. In H. Österle, J. Schelp, & R. Winter (Eds.), *Proceedings of 15th European conference on information systems (ECIS '07)* (pp. 1620–1631). St.Gallen, Switzerland: AIS Electronic Library.

Hoogland, J. (2012). BPM Round Table, "The future of BPM technology: mines, green fields, and clouds." Eindhoven. Retrieved February 05, 2013, from http://www.win.tue.nl/bpmroundtable/doku.php?id=news:meeting_05112012:english.

Küster, J., Gerth, C., Förster, A., & Engels, G. (2008). A tool for process merging in business-driven development. In *Proc. of the CAiSE'08 forum, CEUR workshop proceedings* (Vol. 344, pp. 89–92).

La Rosa, M., Dumas, M., Uba, R., & Dijkman, R. M. (2013). Business process model merging: an approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology, 22*(2), 42.

Legner, C., & Wende, K. (2007). The challenges of inter-organizational business process design–a research agenda. In H. Österle, J. Schelp, & R. Winter (Eds.), *Proceedings of the 15th European conference on information systems (ECIS '07)* (pp. 1643–1654). St. Gallen, Switzerland: AIS Electronic Library.

Li, J., Liu, D., & Yang, B. (2007). Process mining: Extending α-algorithm to mine duplicate tasks in process logs. In K. C.-C. Chang, W. Wang, L. Chen, C. A. Ellis, C.-H. Hsu, A. C. Tsoi, & H. Wang (Eds.), *Proceedings of conference on advances in web and network technologies, and information management (APWeb-WAIM '07), LNCS 4537* (pp. 396–407). Huang Shan, China: Springer Berlin Heidelberg.

Li, C., Reichert, M., & Wombacher, A. (2010). The minadept clustering approach for discovering reference process models out of process variants. *International Journal of Cooperative Information Systems, 19*(3 & 4), 159–203.

Min, H., & Zhou, G. (2002). Supply chain modeling: Past, present and future. *Computers & Industrial Engineering, 43*(1–2), 231–249.

Moody, D. L. (2003). The method evaluation model: A theoretical model for validating information systems design methods. In C. U. Ciborra, R. Mercurio, M. de Marco, M. Martinez, & A. Carignani (Eds.), *Proceedings of the 11th European conference on information systems (ECIS '03)* (pp. 1327–1336). Naples, Italy: AIS Electronic Library.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45–77. http://dx.doi.org/10.2753/MIS0742-1222240302.

Rescher, N. (2012). *Pragmatism: The restoration of its scientific roots*. Transaction Publishers (p. 313).

Rozinat, A., Mans, R. S., Song, M. S., & Van der Aalst, W. M. P. (2009). Discovering simulation models. *Information Systems, 34*(3), 305–327. http://dx.doi.org/10.1016/j.is.2008.09.002.

Rozinat, A., & Van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems, 33*(1), 64–95.

Schonenberg, H., Weber, B., Van Dongen, B. F., & Van der Aalst, W. M. P. (2008). Supporting flexible processes through recommendations based on history. In M. Dumas, M. Reichert, & M.-C. Shan (Eds.), *Proceedings of the 6th international conference on business process management (BPM '08), LNCS5240* (pp. 51–66). Milan, Italy: Springer, Berlin Heidelberg.

Simatupang, T. M., & Sridharan, R. (2001). A characterization of information sharing in supply chains. In *Proceedings of 36th annual conference of the operations research society of New Zealand (ORSNZ conference twenty naught one)* (pp. 1–19). Canterbury, New Zealand: ORNZ.

Stadtler, H. (2008). Supply chain management – An overview. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning – Concepts, models, software, and case studies – Part I* (pp. 9–36). Berlin Heidelberg: Springer.

Sun, S., Kumar, A., & Yen, J. (2006). Merging workflows: A new perspective on connecting business processes. *Decision Support Systems, 42*(2), 844–858. http://dx.doi.org/10.1016/j.dss.2005.07.001.

Van der Aalst, W. M. P. (1999a). Interorganizational workflows: An approach based on message sequence charts and Petri nets. *Systems Analysis – Modelling – Simulation, 34*(3), 335–367.

Van der Aalst, W. M. P. (1999b). Process-oriented architectures for electronic commerce and interorganizational workflow. *Information Systems, 24*(8), 639–671.

Van der Aalst, W. M. P. (2000). Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries. *Information & Management, 37*(2), 67–75.

Van der Aalst, W. M. P. (2011). *Process mining: Discovery, conformance and enhancement of business processes*. Springer (p. 368).

Van der Aalst, W. M. P., Adriansyah, A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., et al. (2011). Process mining manifesto. In F. Daniel, K. Barkaoui, & S. Dustdar (Eds.), *Proceedings of the 9th international conference on business process management workshops (BPM '11), LNBIP 99* (pp. 169–194). Clermont-Ferrand, France: Springer, Berlin Heidelberg.

Van der Aalst, W. M. P. (2008). Challenges in business process analysis. In J. Filipe, J. Cordeiro, & J. Cardoso (Eds.), *Proceedings of the 9th international conference on enterprise information systems (ICEIS '07), LNBIP 12* (pp. 27–42). Funchal, Madeira: Springer verlag.

Van der Aalst, W. M. P., & Song, M. S. (2004). Mining social networks: Uncovering interaction patterns in business processes. In J. Desel, B. Pernici, & M. Weske (Eds.), *Proceedings of the 2nd international conference on business process management (BPM '04), LNBIP 3080* (pp. 244–260). Potsdam, Germany: Springer, Berlin Heidelberg.

Van der Aalst, W. M. P., Van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., & Weijters, A. J. M. M. (2003). Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering, 47*(2), 237–267.

Van der Aalst, W. M. P., & Weijters, A. J. M. M. (2004). Process mining: A research agenda. *Computers in Industry, 53*(3), 231–244.

Van Dongen, B. F., De Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., & Van der Aalst, W. M. P. (2005). The ProM framework: A new era in process mining tool support. In G. Ciardo & P. Darondeau (Eds.), *Proceedings of the 26th*

international conference on applications and theory of Petri nets (ICATPN '05), LNCS 3536 (pp. 444–454). Miami, USA: Springer, Berlin Heidelberg.

Weijters, A. J. M. M., & Van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering, 10*(2), 151–162.

Weijters, A. J. M. M., & Van der Aalst, W. M. P. (2006). Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven. Tech. Rep. WP, 166*, 1–34.

Yu, Z., Yan, H., & Cheng, T. (2001). Benefits of information sharing with supply chain partnerships. *Industrial Management & Data Systems, 101*(3), 114–121.