Mining Inter-Organizational Business Process Models from EDI Messages: A Case Study from the Automotive Sector*

Robert Engel[1], Wil van der Aalst[2], Marco Zapletal[1], Christian Pichler[1], and Hannes Werthner[1]

[1] Vienna University of Technology
Institute of Software Technology and Interactive Systems
Electronic Commerce Group
{engel,marco,pichler,werthner}@ec.tuwien.ac.at
[2] Eindhoven University of Technology
Department of Computer Science
w.m.p.v.d.aalst@tue.nl

Abstract. Traditional standards for *Electronic Data Interchange* (EDI), such as EDIFACT and ANSI X12, have been employed in Business-to-Business (B2B) e-commerce for decades. Due to their wide industry coverage and long-standing establishment, they will presumably continue to play an important role for some time. EDI systems are typically not "process-aware", i.e., messages are standardized but processes simply "emerge". However, to improve performance and to enhance the control, it is important to understand and analyze the "real" processes supported by these systems. In the case study presented in this paper we uncover the inter-organizational business processes of an automotive supplier company by analyzing the EDIFACT messages that it receives from its business partners. We start by transforming a set of observed messages to an event log, which requires that the individual messages are correlated to process instances. Thereby, we make use of the specific structure of EDIFACT messages. Then we apply process mining techniques to uncover the inter-organizational business processes. Our results show that inter-organizational business process models can be derived by analyzing EDI messages that are exchanged in a network of organizations.

Keywords: process mining, inter-organizational business processes, BPM, EDI, event correlation

1 Introduction

Recent academic research on inter-organizational business processes has mainly focused on Web service choreographies and related XML-based technologies.

* This research has been conducted in the context of the *EDImine* project and has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT10-010.

Nevertheless, many inter-organizational systems are still realized by traditional Electronic Data Interchange (EDI) [12, 7, 16] standards, such as EDIFACT [2] or ANSI X12. Presumably they will continue to be the primary data formats in Business-to-Business (B2B) interaction for years to come [24].

However, such traditional EDI systems are usually process-unaware, meaning that they are solely responsible for sending and receiving messages. When companies intend to analyze their inter-organizational business processes they generally have to rely on a-priori models, if models documenting the business processes exist at all. In case there are models, those may describe the business processes as they were planned, which is not necessarily in sync with the real-world business processes. To address this shortcoming, we seek to derive models of inter-organizational business processes from EDI message exchanges [8]. Thereby we employ and extend existing *process mining* [19, 21] techniques, which so far have concentrated on business processes within single organizations.

In this paper we present a case study where we analyze the EDI-based inter-organizational business processes of an automotive supplier company. These processes include activities for placing and changing orders as well as for the management of a just-in-time supply chain. Our results show that inter-organizational business process models can be derived by analyzing EDI messages that are exchanged between companies. This implies that we can apply a wide range of process mining techniques. Besides discovering the "real" processes, we can discover bottlenecks, discover deviations from some predefined behavior (conformance checking), predict performance (e.g., predict the remaining flow time for running instances), etc. [19].

To apply process mining techniques we need to convert a collection of EDI messages to an event log. In such an event log, each sent or received EDI message represents an event. Process mining algorithms generally presuppose that individual events can be assigned to process instances (i.e., cases) [19]. However, in practice many legacy implementations of EDI systems do not provide case identifiers in individual messages. This is also the case for the data set under consideration in this case study. Hence, in order to enable the application of process mining techniques in such settings, it is necessary to *correlate individual EDI messages to process instances*. Consequently, our approach for the analysis of EDI messages for deriving inter-organizational business process models comprises two main steps: (i) correlation of individual, unlinked EDI messages to process instances in order to attain an event log suitable for process mining and (ii) application of a suitable process mining algorithm on the event log to derive the inter-organizational business process models.

The remainder of this paper is organized as follows. In Section 2, related work is discussed. In Section 3, we describe our approach of correlating individual EDI messages to process instances as a necessary prerequisite for the application of process mining algorithms on the data set used in this case study. In Section 4, the actual case study as well as an interpretation of the results are provided. Finally, in Section 5, a critical discussion of the results and an outlook on future work are given.

2 Related Work

In [8] we introduce our overall approach of extending process mining techniques for inter-organizational business processes by means of examining EDI message exchanges. Process mining techniques [19, 21, 1, 3, 4] extract knowledge about business processes by analyzing event logs. So far, the main focus has been on the analysis of processes inside single organizations. The few publications on process mining in an inter-organizational context tend to focus on the area of Web services [20, 22, 6, 13, 15, 17]. For example, in [20] conformance checking techniques are applied to the message logs of Oracle BPEL. Another example may be found in [22] where process mining techniques are applied in the context of IBM's WebSphere.

Dustdar et al. [6] proposed techniques for services interaction mining, i.e., applying process mining techniques to the analysis of service interactions, and in [14, pp. 30-32], the suitability of general data mining methods [11] for correlating messages is discussed, where, however, several serious limitations are identified. In [18], an algorithm for mining sequential patterns from databases is presented.

Nezhad et al. [13, 15] developed techniques for event correlation and process discovery from web service interaction logs. The authors introduce the notion of a "process view" which is the result of a particular event correlation. However, they argue that correlation is subjective and that multiple views are possible. A collection of process views is called the "process space". With regard to message correlation, Nezhad et al. [14] also describe *correlation rule patterns*, a formalization of corresponding correlation rules as well as heuristics to derive such rules for correlating a set of messages to conversations. The correlation approach presented in the aforementioned work can be seen as a generalization of the mechanism for message correlation used in this paper. For example, *key/value-correlation*, as introduced in this paper, corresponds to *key-based correlation* in [14]; the *synonymy* and *concatenation* rules introduced in this paper can also be expressed with *reference-based correlation* rules as described in [14]. However, the correlation mechanism presented in this paper - while potentially applicable to other domains - was derived from specific observations on real-world EDI messages.

In [17], a technique is presented for correlating messages with the goal to visualize the execution of Web services.

In practice, however, neither explicit choreography modeling nor Web services are widely employed in electronic business transactions. Rather, traditional approaches to EDI such as EDIFACT [2] still play an overwhelmingly dominant role [24]. Therefore, we developed techniques for the correlation of EDI messages.

The problem of recognizing process instances from a set of EDI messages corresponds to the requirement of *event correlation* in process mining [19, p. 113]. Literature on this topic is generally sparse with the notable exception of [9], where a probabilistic approach for correlating events to cases without any a-priori information is proposed. However, in the case of EDI messages one can work with richer information by examining the actual content of messages sent and received by EDI systems. This is a new approach to process mining, since it

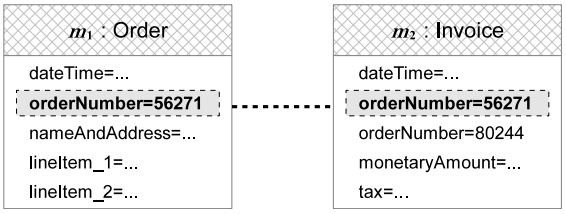

Fig. 1: Example for correlating EDI messages by matching the `orderNumber` data elements and their values

considers also the "content" within the execution of a process instance instead of treating the sending and receiving of messages as opaque events.

Related to mining process models from exchanged messages is mining from I/O operations as described in [5]. In [10], an approach is presented that accounts for the identification of different variants of processes by clustering traces that share similar behavior patterns.

3 Correlation of EDI Messages to Process Instances

Contrary to process-based approaches such as BPEL, traditional EDI messages usually do not contain explicit *case identifiers* that map individual messages to process instances. However, traditional EDI standards such as EDIFACT define data elements that may contain back references to previously sent or received messages of the same business case. For example, the *Order* and *Invoice* messages of the purchase order transaction shown in Fig. 1 both contain the data element `orderNumber`. Additionally, both messages contain the same value in this data element (*56271*). Hence, in this case the order number can be seen as a case identifier that allows for the conclusion that both messages belong to the same process instance. Typically, it is not as simple as shown in Fig. 1. Therefore, we discuss the basic correlation mechanisms and then present a concrete algorithm.

3.1 Key/Value-Correlation

We define a *trace* as a set of EDI messages that represents a process instance or a fragment thereof. *Note that although in process mining the order of events is generally crucial, we define a trace as an unordered set of messages because the order is given implicitly in the timestamps of the messages and can thus be recovered after correlation.* Furthermore, we define a *correlator* as a key for a specific data element defined in one ore more EDI message types that can be used for correlating messages to traces by matching the values contained in the data element. Messages are assigned to a trace when each of them contains the data element denoted by the correlator and the values in the data elements match. We refer to this basic mechanism as *key/value-correlation*.

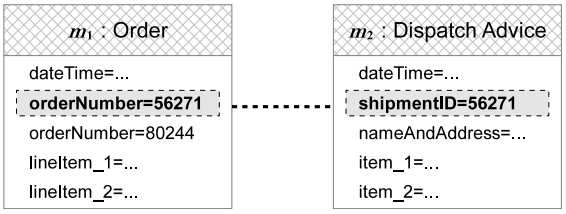

Fig. 2: Example for synonymous correlators

Note that not all data elements can serve as correlators. Some data elements may contain repeating values across multiple messages by coincidence, or the contained information may be unrelated to actual process instances (e.g., an address). We define a *set of potential correlators* $\mathbb{C}$ as a set of keys for data elements that can be potentially used for correlation in some EDI standard. In the case of the EDIFACT standards, a number of different *segments* is defined that group related data elements. For example, the RFF (for *Reference*) segment is intended to hold information that references some other message, business document or other business entity. This can be exploited for correlating EDIFACT messages. Observations on the sample data set used in this case study reveal that many data elements in the RFF segment[3] represent indeed potentially effective correlators (e.g., order numbers are usually contained in RFF segments). Hence, one may use the various types of references defined in the RFF segment as a starting point for building a set of potential correlators (cf. Section 4). In practice, however, varying implementations of EDI standards require further narrowing of the set of potential correlators to a subset that suffices for the correlation of a specific EDI implementation. We refer to such a subset as a *correlator set* $\mathcal{C} \subseteq \mathbb{C}$.

3.2 Synonymous Keys

As will be shown in detail in Section 4, the data set used for this case study reveals that distinct data elements sometimes contain identical information. For example, a seller receives an *Order* message (m_1) from a customer with a specific order number (*56271*) (cf. Fig. 2). The seller replies with a *Dispatch Advice* message (m_2) indicating that ordered goods have been shipped. To facilitate the alignment of orders with shipments, the seller references the order numbers received from the customer in shipment IDs of corresponding shipments. The shipment ID (*56271*) of the *Dispatch Advice* matches with the order number that was originally assigned by the customer in the *Order* message (*56271*). If correlators `orderNumber` and `shipmentID` are defined as *synonymous*, then they are regarded as identical keys in key/value-correlation, i.e., m_1 and m_2 are correlated. Note that every correlator is reflexively synonymous, i.e., every correlator is synonymous to itself.

[3] see `http://live.unece.org/trade/untdid/d10b/tred/tred1153.htm`

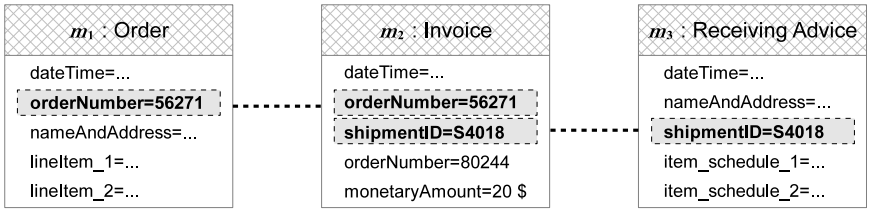

Fig. 3: Example for concatenation of traces

3.3 Concatenation of Traces

We also consider that results may be further improved by concatenating traces at points where they intersect (i.e., overlap) with respect to individual messages. Consider, for example, the three messages shown in Fig. 3 representing a purchase order transaction. An *Order* message (m_1) of a customer is followed by a corresponding *Invoice* message (m_2) sent by the seller. Because both messages refer to the same `orderNumber` (*56271*), the trace $\langle m_1, m_2 \rangle$ can be recognized by applying key/value-correlation. However, in the *Invoice* message, the seller assigns a new `shipmentID` (*S4018*) to a shipment of goods corresponding to the order. When the customer receives the shipment, it advises the seller by sending a *Receiving Advice* message (m_3) containing the same value in the `shipmentID` data element (*S4018*). Hence, $\langle m_2, m_3 \rangle$ can also be recognized as a trace by key/value-correlation. From a business point of view, all three messages may be regarded as part of the same process instance since they belong to the same purchase order transaction. The larger trace $\langle m_1, m_2, m_3 \rangle$ can be reconstructed by concatenating the traces $\langle m_1, m_2 \rangle$ and $\langle m_2, m_3 \rangle$ as a result of recognizing m_2 as the intersection of the message sets. In this example the overlap occurs with respect to two different correlators (`orderNumber` and `shipmentID`), but overlaps may also occur in traces of more than two correlators or a single correlator in case the corresponding data element occurs multiple times in individual EDI messages.

Note that this correlation rule may lead to undesired results in cases where concatenation is not appropriate. This implies that subsets of the correlator set need to be selected for which the concatenation rule should be applied. Such *concatenation groups* of correlators specify groups of correlators. When any combination of one or more correlators in a concatenation group is encountered in a single message that is contained in multiple traces resulting from key/value-correlation, the respective traces are concatenated. This means that a new trace is built from the union of the respective message sets and the original traces get discarded. This also includes cases where just one of the correlators from the concatenation group occurs repeatedly in one message that is contained in multiple traces.

3.4 Correlation Algorithm

In the following we present the algorithm for correlating EDI messages to process instances that is used in the case study presented in Section 4. It applies key/value-correlation and selectively the rules for synonymous correlators and concatenation of traces on a set of observed messages. The algorithm has to be parameterized with a correlator set and a corresponding specification of relationships between the correlators with respect to synonymies and concatenation. These relationships are modeled by means of *synonymy sets* and *concatenation sets* as described below.

Potential Correlators, Correlator Set. $\mathbb{C}$ is the set of potential correlators, i.e., all keys for data elements that can potentially be used for correlating EDI messages. $\mathcal{C} \subseteq \mathbb{C}$ is the selected set of correlators. The keys in the set $\mathbb{C} \setminus \mathcal{C}$ will be discarded when correlating messages.

Synonymy Group, Synonymy Set. A *synonymy group* $\mathcal{SG} \subseteq \mathcal{C}$ is a set of correlators that are considered to be synonymous. For example, $\mathcal{SG} = \{\texttt{orderNumber}, \texttt{shipmentID}\}$ is a synonymy group for the transaction shown in Fig. 2. A synonymy group may be a singleton set, e.g., $\mathcal{SG} = \{\texttt{nameAndAddress}\}$. This implies that correlator `nameAndAddress` is only synonymous with itself.

A *synonymy set* $\mathcal{SS} = \{\mathcal{SG}_1, \mathcal{SG}_2, \ldots \mathcal{SG}_m\} \subseteq \mathcal{P}(\mathcal{C})$ is a set of synonymy groups partitioning $\mathcal{C}$, i.e., $\mathcal{SG}_i \cap \mathcal{SG}_j \neq \emptyset$ implies that $i = j$ (pairwise disjoint) and $\bigcup \mathcal{SS} = \mathcal{C}$.[4]

Concatenation Group, Concatenation Set. A *concatenation group* $\mathcal{CG} \subseteq \mathcal{SS}$ is a set of synonymy groups for which the traces that result from correlation with these synonymy groups are to be concatenated in case of overlaps. For example, $\mathcal{CG} = \{\{\texttt{orderNumber}\}, \{\texttt{shipmentID}\}\}$ is a concatenation group for the transaction shown in Fig. 3. A concatenation group may contain only a single synonymy group; in this case concatenation of traces is applied with regard to overlaps in traces that were correlated by this synonymy group. This may be useful when an EDI message contains multiple instances of the same data element (e.g., multiple order numbers).

A *concatenation set* $\mathcal{CS} = \{\mathcal{CG}_1, \mathcal{CG}_2, \ldots \mathcal{CG}_n\} \subseteq \mathcal{P}(\mathcal{SS})$ is a set of concatenation groups. Unlike synonymy sets, a concatenation set $\mathcal{CS}$ does not necessarily contain all correlators from $\mathcal{C}$; synonymy groups that are inappropriate for the concatenation rule are left out of concatenation sets. This is to prevent the concatenation of traces of single synonymy groups where this behavior is undesired.

Messages, Strings, Values. $\mathbb{M}$ is the universe of EDI messages, i.e., all potential messages. $\mathbb{S}$ is the set of all strings, i.e., possible data values. $\mathcal{MS} \subseteq \mathbb{M}$ is a concrete set of messages. For modeling actual values in data elements of EDI messages we define the *values function* $val : \mathbb{M} \times \mathcal{SS} \rightarrow \mathcal{P}(\mathbb{S})$. The values function returns the values from a specific message and synonymy group. $val(m, \mathcal{SG})$ is a set of strings corresponding to message $m \in \mathbb{M}$ and synonymy group $\mathcal{SG} \in \mathcal{SS}$. As certain data elements may occur repeatedly in a single EDI message, the

[4] $\mathcal{P}(X)$ denotes the powerset of some set X. $\bigcup X$ denotes the union of the subsets in a set of sets X.

values function might also yield multiple values for a particular message and synonymy group. Each of these values has to be considered for correlation. Thus, the values function returns a set of strings[5]. For example, for messages m_1 and m_2 shown in Fig. 2 and a synonymy group $\mathcal{SG} = \{\texttt{orderNumber}, \texttt{shipmentID}\}$, both $val(m_1, \mathcal{SG}) = \{\text{"56271"}, \text{"80244"}\}$ and $val(m_2, \mathcal{SG}) = \{\text{"56271"}\}$ hold.

Correlation. For computing the traces that result from the application of key/value-correlation and the synonymy rule using a single synonymy group $\mathcal{SG} \in \mathcal{SS}$ on a set of observed messages $\mathcal{MS} \subseteq \mathbb{M}$, we define the *correlation function* $corr : \mathcal{P}(\mathbb{M}) \times \mathcal{SS} \to \mathcal{P}(\mathcal{P}(\mathbb{S}) \times \mathcal{P}(\mathbb{M}))$ as

$$corr(\mathcal{MS}, \mathcal{SG}) = \Big\{ (V, \mathcal{MS}') \in \mathcal{P}(\mathbb{S}) \times \mathcal{P}(\mathcal{MS}) \;\Big|\; V = \bigcap_{m \in \mathcal{MS}'} val(m, \mathcal{SG}) \setminus \bigcup_{m \in \mathcal{MS} \setminus \mathcal{MS}'} val(m, \mathcal{SG}) \neq \emptyset \Big\}.$$

$(V, \mathcal{MS}') \in corr(\mathcal{MS}, \mathcal{SG})$ represents a trace[6] containing messages $\mathcal{MS}'$ correlated using the set of values V. Note that the resulting traces, which are sets of messages, need not be disjoint: as a single message may yield multiple values for a single synonymy group, it might be contained in multiple traces of this synonymy group as well.

The set of all traces with respect to a synonymy set $\mathcal{SS}$ and message set $\mathcal{MS}$ can be obtained as follows:

$$\mathcal{T}_{\mathcal{SS},\mathcal{MS}} = \bigcup_{\mathcal{SG} \in \mathcal{SS}} corr(\mathcal{MS}, \mathcal{SG}).$$

Concatenation. Let $\mathcal{CS}$ be the concatenation set used to concatenate traces. Two traces $(V_1, \mathcal{MS}_1), (V_2, \mathcal{MS}_2) \in \mathcal{T}_{\mathcal{SS},\mathcal{MS}}$ are related, denoted $(V_1, \mathcal{MS}_1) \sim_{\mathcal{CS}} (V_2, \mathcal{MS}_2)$, if and only if:

$$\exists_{CG \in CS}\ \exists_{\mathcal{SG}_1, \mathcal{SG}_2 \in CG}\ \exists_{m \in \mathcal{MS}_1 \cap \mathcal{MS}_2}\ V_1 \subseteq val(m, \mathcal{SG}_1)\ \wedge\ V_2 \subseteq val(m, \mathcal{SG}_2).$$

$\sim^*_{\mathcal{CS}}$ is the reflexive transitive closure of $\sim_{\mathcal{CS}}$, i.e., two traces $(V_1, \mathcal{MS}_1)$ and $(V_2, \mathcal{MS}_2)$ are related, i.e. $(V_1, \mathcal{MS}_1) \sim^*_{\mathcal{CS}} (V_2, \mathcal{MS}_2)$, if $(V_1, \mathcal{MS}_1) = (V_2, \mathcal{MS}_2)$ or there exists a trace $(V, \mathcal{MS}) \in \mathcal{T}_{\mathcal{SS},\mathcal{MS}}$ such that $(V_1, \mathcal{MS}_1) \sim_{\mathcal{CS}} (V, \mathcal{MS})$ and $(V, \mathcal{MS}) \sim^*_{\mathcal{CS}} (V_2, \mathcal{MS}_2)$.

All traces related through $\sim^*_{\mathcal{CS}}$ are merged into larger traces. The set of traces resulting from the application of key-/value-correlation and selectively the rules for synonymous correlators and concatenation of traces with respect

[5] However, if the same value occurs repeatedly in the same data element in a single message, this will not be reflected by the result of the values function as the function's codomain is a set rather than a multiset.

[6] Note that earlier we described a trace as a set of messages (the ordering can be derived based on the timestamps). Here, we refer to a trace as a pair $(V, \mathcal{MS}')$ where $\mathcal{MS}'$ is the set of time-ordered messages and V is the set of strings used to correlate the messages in the trace.

to a message message set $\mathcal{MS}$, a synonymy set $\mathcal{SS}$ and a concatenation set $\mathcal{CS}$ can be obtained as follows:

$$\mathcal{T}^{\mathcal{CS}}_{\mathcal{SS},\mathcal{MS}} = \Big\{ \bigcup \Big\{ \mathcal{MS}_2 \mid (V_2, \mathcal{MS}_2) \in \mathcal{T}_{\mathcal{SS},\mathcal{MS}} \wedge (V_1, \mathcal{MS}_1) \sim^*_{\mathcal{CS}} (V_2, \mathcal{MS}_2) \Big\} \Big| (V_1, \mathcal{MS}_1) \in \mathcal{T}_{\mathcal{SS},\mathcal{MS}} \Big\}.$$

An event log suitable for the subsequent application of process mining techniques can be obtained by generating an event for each message from $\mathcal{MS}$. There, the messages are grouped to process instances according to the traces in $\mathcal{T}^{\mathcal{CS}}_{\mathcal{SS},\mathcal{MS}}$ and sorted according to their timestamps.

4 Case Study

For the case study at hand we examined a data set of 410 EDIFACT messages that has been supplied by an automotive supplier company. The data set consists of 180 DELFOR (*Delivery schedule message*), 75 DELJIT (*Delivery just in time message*), 28 GENRAL (*General purpose message*), 28 ORDCHG (*Purchase order change request message*), 2 ORDERS (*Purchase order message*) and 97 RECADV (*Receiving advice message*) messages received by the company in a period of approximately two months. The set does not contain any messages sent by the company. Note that we do not presume any further knowledge about the company's processes nor availability of any other meta-information about the data set, in particular with regard to completeness.

4.1 Methodology

We analyzed the data set with regard to the process models that can be mined by applying existing process mining algorithms on sets of process instances that have been recognized using the approach presented in Section 3. For interpreting the mined process models we referred to the EDIFACT standards and the therein specified purposes of particular message types. For evaluating the results we discussed the mined process models as well as our interpretation of these models with representatives from the company that has supplied the data set.

For performing the case study we implemented the correlation algorithm described in Section 3 in a plug-in[7] for ProM 6[8] [23]. We will further on refer to the plug-in as *correlation tool*. The correlation tool comes with a pre-defined set of 778 potential correlators for EDIFACT messages. These potential correlators were automatically derived from the code list that defines the possible values an

[7] The plug-in is publicly available in the *EDImine* package of ProM 6 nightly builds as of November 2011.

[8] ProM is developed at the Eindhoven University of Technology and currently regarded as the most prevalent tool in the area of process mining. `http://www.processmining.org`

Table 1: Potential correlators occurring in the data set with average trace lengths resulting from key/value-correlation

Correlator	Avg. trace length [msgs.]
Profile_number	30.0
Order_document_identifier__buyer_assigned	14.7
Delivery_schedule_number	9.6
Previous_delivery_instruction_number	4.5
SID__Shipper_s_identifying_number_for_shipment	1.8
Release_number	1.0
Previous_delivery_schedule_number	1.0
Customer_reference_number	1.0
Contract_number	1.0
Shipment_reference_number	1.0

RFF (*Reference*) segment can hold according to the UN/EDIFACT standard, Release D10B[9]. We use this set of 778 potential correlators in our case study[10].

4.2 Correlation

A first examination of the messages in the data set reveals that ten of the 778 potential correlators derived from the possible RFF segment qualifiers actually occur in the examined data set. These ten correlators are shown in Table 1 ordered by average length of the traces (i.e., number of contained messages/activities) that result from applying key/value-correlation on the data set using the corresponding correlator. The average length of the traces may give an indication for the suitability of the corresponding correlator; however, this metric can also be misleading. For example, adding `Profile_number` and `Order_document_identifier__buyer_assigned` to the correlator set and applying the correlation algorithm without any non-reflexive synonymies and an empty concatenation set results in the set of 18 recognized traces visualized in Fig. 4. These 18 traces comprise 250 of the 410 messages. The visualization reveals that `Profile_number` is not a suitable correlator because the data element it denotes contains always the same value and is only present in ORDER and ORDCHG messages that also contain order numbers (denoted by `Order_document_identifier__buyer_assigned`). Thus, it can be removed from the correlator set without losing any traces.

A closer look at the data set reveals that most DELFOR (68%) and all RECADV messages contain the data element denoted by `Order_document_identifier__buyer_assigned`, and all DELJIT messages contain the data element denoted by `Delivery_schedule_number`. Thus, the correlator `Delivery_schedule_number`, which is rated third in terms of average trace

[9] see `http://live.unece.org/trade/untdid/d10b/trsd/trsdrff.htm`

[10] However, the correlation tool allows a user to define and use any set of potential correlators.

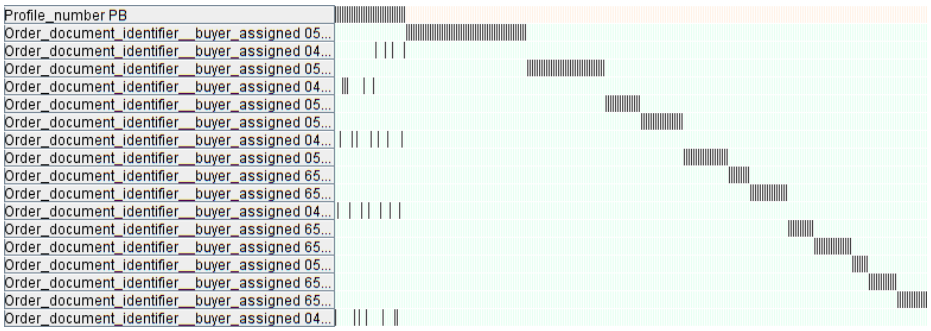

Fig. 4: Recognized traces in a condensed visualization (screen-shot). Columns represent individual EDI messages and rows represent recognized traces. The labels on the left hand side show the correlator that was used for matching the messages of a particular trace. Dark markings in the matrix on the right hand side indicate membership of a message in a trace. Note that some traces exhibit partial overlaps on the left hand border of the matrix.

length, is added to the correlator set. Also, with respect to the aforementioned three message types, the values contained in these two data elements match in 6 cases (of 12 resp. 28 cases total when comparing DELJIT with DELFOR and RECADV messages), giving a strong indication of the presence of a synonymy relationship between these two correlators. Hence, by putting these correlators in a synonymy group it is possible to connect traces consisting of DELFOR and RECADV messages with traces consisting of just DELJIT messages. Guided by this insight, the two correlators are added to a synonymy group resulting in a synonymy set $\mathcal{SS} = \{\{$`Order_document_identifier__buyer_assigned`, `Delivery_schedule_number`$\}\}$. Note that although some of the remaining potential correlators may also lead to interesting results, we leave them unconsidered in this case study for simplicity. Thus, the resulting correlator set is $\mathcal{C} =$ $\{$`Order_document_identifier__buyer_assigned`, `Delivery_schedule_number`$\}$.

The absence of partial overlaps in a new visualization of the preliminarily computed traces (as, for example, contrary to Fig. 4) indicates that the concatenation rule is not applicable to any of the previously defined correlators/synonymy groups. Hence, the concatenation set is left empty, i.e., $\mathcal{CS} = \emptyset$. Applying the correlation algorithm with this parameterization ($\mathcal{CS}$, $\mathcal{SS}$) on the data set $\mathcal{MS}$ yields 39 individual process instances $\mathcal{T}^{\mathcal{CS}}_{\mathcal{SS},\mathcal{MS}}$ comprising 382 messages in total, where each message is assigned to exactly one trace. This amounts exactly to the number of messages in the data set excluding messages of the GENRAL (*General purpose message*) message type. Hence, at this point the messages of the GENRAL message type are discarded and not included in subsequent process mining.

In order to derive the inter-organizational business process model, a process mining algorithm has to be applied. Hence, an event log suitable for the application of process mining algorithms is built from the messages in the data set

according to the discovered process instances $\mathcal{T}^{\mathcal{CS}}_{\mathcal{SS},\mathcal{MS}}$. In this event log each message represents an event that is assigned to a specific process instance. Furthermore, the events are ordered according to the timestamps of the corresponding messages. Then a suitable process mining algorithm has to be selected, as discussed in the following.

4.3 Mining the Inter-Organizational Business Process Model

Different process mining algorithms exhibit different strengths and weaknesses with regard to properties such as representational bias, robustness (i.e., ability to deal with noise) or assumptions about completeness [19, pp. 159]. *Heuristic mining algorithms* consider frequencies of events and sequences when deriving a process model. Together with their favorable properties regarding representational bias, these factors make heuristic mining algorithms much more robust than most other approaches [19, p. 163].

The data set under consideration shows apparent signs that it is incomplete. For example, the appearance of multiple traces in the generated event log which consist of repeated ORDCHG messages without corresponding ORDERS messages is an indication that the data set is missing the initial ORDERS messages. This is likely due to the limited time frame under which messages were collected. Furthermore, knowing that the EDI messages in the data set were observed in a limited time window allows for the assumption that process instances may be missing leading or trailing messages/activities. This assumed incompleteness of the data set suggests that a relatively robust mining algorithm should be used. Hence, we choose the *Heuristics Miner* plug-in of ProM 6 [25] for mining a business process model from the correlated EDI messages.

Providing the previously generated event log as input to the Heuristics Miner plug-in yields the inter-organizational process model shown in Fig. 5.

A possible interpretation of the figure is that the observed EDI messages are artifacts of two separate sub-processes. The first process appears to be a *purchase order* process; it consists of an activity to place orders (*ORDERS*), succeeded by multiple requests for changing these orders (*ORDCHG*). The second process appears to be a continuous (i.e., looping) supply chain management (SCM) process, where customers continuously send *delivery forecasts* (*DELFOR*) and *just-in-time delivery* requests (*DELJIT*). When goods are received, customers send *receiving advices* (*RECADV*). Note that the GENRAL messages from the data set do not appear in the mined process model because the business information they convey is not covered by any of the correlators we have used in this case study.

5 Conclusion and Reflection

According to the EDIFACT standards a *Delivery just in time* message (DELJIT) is intended to provide "firm deliver instructions" with fine-grained time granularity (i.e., days, hours) whereas *Delivery schedule* messages (DELFOR) are

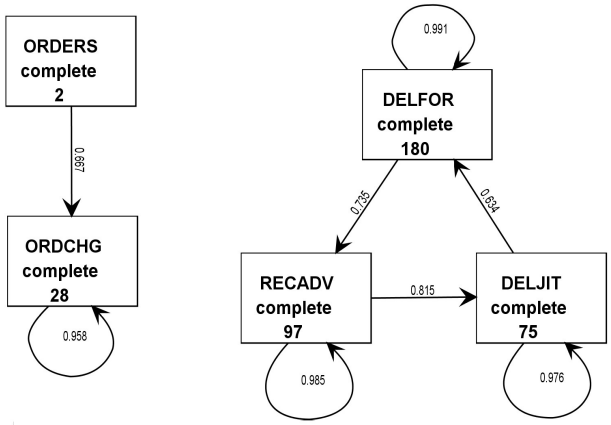

Fig. 5: Process model derived from the resulting traces with the *Heuristics Miner* plug-in of ProM 6. The arrows represent dependency relations between activities; the associated values are measures for the strength of the dependency relation, where values close to 1 indicate a strong positive dependency. The values next to the activity labels indicate the number of events in the event log (i.e., the number of messages) that led to the recognition of the corresponding activity (see [25] for a detailed description of the elements in the graph).

intended for forecasts of supply needs with a more coarse-grained time granularity (i.e., months, weeks). Also, common sense dictates that a *Receiving advice* message (RECADV) should be received after corresponding DELJIT and/or DELFOR messages. These considerations suggest that a corresponding SCM process should look similar to the (fictional) process model illustrated in Fig. 6. The mined SCM sub-process model visualized in Fig. 5 differs from the process model illustrated in Fig. 6 insofar that (i) the causality relation (i.e., the "control flow") represented by the arrows is in the opposite (and counter-intuitive) direction, (ii) that it contains repeated occurrences of individual messages/activities (length-one loops) and (iii) that it does not "end" with a receiving advice, but is a continuous (looping) process. Furthermore, one could ask why there is no link (i.e., no arrow in the graph) between the purchase order and SCM sub-processes. In the following, possible reasons for these differences are discussed.

A discussion of our results with representatives of the company that supplied the data set has led to some interesting insights. First, the company stated that it actually only uses the information from received DELFOR and DELJIT messages; received messages of other types are not processed. This means that the remaining message types are artifacts of processes that were originally designed and implemented unilaterally by customers of the company. Secondly, the company confirmed that a supply-chain-management process implemented by DELFOR, DELJIT and RECADV messages, including length-one loops as

Fig. 6: Model of a fictional supply chain management process consisting of delivery forecasts, just-in-time delivery requests and receiving advices.

featured in the mined model, is indeed employed. However, as already conjectured before, the order of the messages/activities (the direction of the arrows) does not correspond to the real-world process. The question of what causes the missing link between the order and SCM sub-processes still remained open after the discussion. We assume that this might be due to the limited time frame in which the messages for the data set were collected. If the delay between ORDERS and ORDCHG messages on the one hand and DELFOR, DELJIT and RECADV messages on the other hand is long enough as to span over a significant portion of the period in which the messages were collected, this would prevent us from finding references between these groups of messages in the data set (cf. Fig. 7).

Regarding the inaccurate order of activities in the SCM sub-process we consider the possibility that this is a result of the observed data set being incomplete *within* the observed time window. In other words, the data set may not contain *all* messages that were indeed received by the company in a specific timespan. This in turn could lead to distortions in the mined process model such as an incorrect order of activities in a looping process. However, one cannot draw such a conclusion with certainty solely by analyzing a supposedly incomplete data set itself when no other constraints are a priori known.

The reason that the *General purpose* messages (GENRAL) do not lead to a corresponding activity in the mined process model is rather trivial: the GENRAL messages in the data set contained only free-text fields with maintenance information for administrators of EDI systems. Therefore, they were not picked up by any of the potential correlators that we have used in the case study. This accurately reflects the fact that the GENRAL messages in the data set do not represent tangible business activities.

In this paper we presented an initial case study using our approach for mining inter-organizational business process models from EDI messages. Discussions with representatives of the company that provided the data set for this case study have shown that the mined process model allowed for a number of substantial insights. The above mentioned peculiarities and limitations of the mined process model are assumed to appear due to the relatively small size and assumed incompleteness of the used data set. We expect that more reliable statements about the generalizability of our approach can be made when using a larger data set. However, based on our results we assume that such larger sets with longer periods of observation and completeness of the data would increase the usability of our approach.

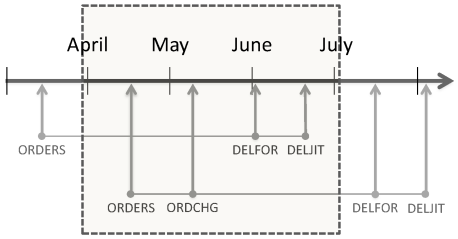

Fig. 7: Long delays between messages of the procurement and SCM sub-processes may exceed the time window in which messages were collected for the data set.

We expect that our approach will help companies to rediscover and document the relationships in their business network and enable the subsequent application of Business Process Management methods on inter-organizational business processes realized by means of EDI. Our future research plans include - besides working with larger sets - complexity analyses of the used correlation algorithm as well as the development of methods for assessing the quality of recognized process instances. Furthermore, we intend to determine how inter-organizational process models can be most effectively visualized to account for the inter-organizational character of the mined process models.

References

1. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *6th International Conference on Extending Database Technology (EDBT 1998)*, volume 1377 of *Lecture Notes in Computer Science*, pages 469–483. Springer, 1998.
2. J. Berge. *The EDIFACT Standards*. Blackwell Publishers, Inc., 1994.
3. J. E. Cook and A. L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, July 1998.
4. A. Datta. Automating the Discovery of AS-IS Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275, 1998.
5. P. C. Diniz and D. R. Ferreira. Automatic Extraction of Process Control Flow from I/O Operations. In *6th International Conference on Business Process Management (BPM 2008)*, volume 5240 of *Lecture Notes in Computer Science*, pages 342–357. Springer, 2008.
6. 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.
7. M. A. Emmelhainz. *EDI: A Total Management Guide*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1992.
8. R. Engel, W. Krathu, M. Zapletal, C. Pichler, W. van der Aalst, and H. Werthner. Process Mining for Electronic Data Interchange. In *12th International Conference on Electronic Commerce and Web Technologies (EC-Web 2011)*, volume 85 of *Lecture Notes in Business Information Processing*, pages 77–88. Springer, 2011.

9. D. Ferreira and D. Gillblad. Discovering Process Models from Unlabelled Event Logs. In *7th International Conference on Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2009.
10. G. Greco, A. Guzzo, L. Pontieri, and D. Sacca. Discovering Expressive Process Models by Clustering Log Traces. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1010–1027, Aug. 2006.
11. D. J. Hand, P. Smyth, and H. Mannila. *Principles of Data Mining*. MIT Press, Cambridge, MA, USA, 2001.
12. N. Hill and D. Ferguson. Electronic Data Interchange: A Definition and Perspective. *EDI Forum: The Journal of Electronic Data Interchange*, 1:5–12, 1989.
13. H. R. M. 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.
14. H. R. M. Nezhad, R. Saint-paul, B. Benatallah, F. Casati, and P. Andritsos. Message Correlation for Conversation Reconstruction in Service Interaction Logs. Technical Report DIT-07-010, University of Trento, 2007.
15. H. R. M. Nezhad, R. Saint-paul, F. Casati, and B. Benatallah. Event Correlation for Process Discovery from Web Service Interaction Logs. *VLDB Journal*, 20(3):417–444, 2011.
16. J.-M. Nurmilaakso. EDI, XML and e-business frameworks: A survey. *Computers in Industry*, 59(4):370–379, 2008.
17. W. D. Pauw, M. Lei, E. Pring, L. Villard, M. Arnold, and J. Morar. Web Services Navigator: Visualizing the Execution of Web Services. *IBM Systems Journal*, 44(4):821–845, 2005.
18. R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *5th International Conference on Extending Database Technology (EDBT 1996)*, volume 1057, pages 3–17. Springer, 1996.
19. W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
20. W. van der Aalst, M. Dumas, A. R. C. Ouyang, and H. Verbeek. Conformance Checking of Service Behavior. *ACM Transactions on Internet Technology*, 8(3):29–59, 2008.
21. W. van der Aalst, H. A. Reijers, A. M. Weijters, B. van Dongen, A. Alves de Medeiros, M. Song, and H. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32:713–732, 2007.
22. W. van der Aalst and H. Verbeek. Process Mining in Web Services: The WebSphere Case. *IEEE Bulletin of the Technical Committee on Data Engineering*, 31(3):45–48, 2008.
23. H. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst. XES, XESame, and ProM 6. In *Information Systems Evolution*, volume 72 of *Lecture Notes in Business Information Processing*, pages 60–75. Springer, 2011.
24. K. Vollmer, M. Gilpin, and J. Stone. *B2B Integration Trends: Message Formats*. Forrester Research, 2007.
25. A. Weijters and W. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.