EDIminer: A Toolset for Process Mining from EDI Messages

Robert Engel[1], R. P. Jagadeesh Chandra Bose[2], Christian Pichler[1], Marco Zapletal[1], and Hannes Werthner[1]

[1] Vienna University of Technology, Vienna, Austria
Institute of Software Technology and Interactive Systems
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
Department of Mathematics and Computer Science

Abstract. Organizations exchange data electronically to perform business transactions using Electronic Data Interchange (EDI). In order to gain insights on such transactions, approaches for *inter-organizational business process mining* based on the observation of exchanged EDI messages have been recently proposed. In recent approaches, however, only meta-information about the exchanged messages, such as message type, interchange time and sender/receiver information, has been used as data base for generating event logs. This neglects the opportunity of using business information from observed EDI messages to arrive at more detailed event logs, which in turn enable mining of detailed process models and fine-grained process performance analyses. In addressing this shortcoming, we present *EDIminer*, a toolset that allows for (i) enhanced visualization of contents of EDI messages, (ii) automatic and/or user-driven definition of mappings of EDI artifacts to events, (iii) generation of events from such mappings, (iv) semi-automatic correlation of events to process instances and (v) generation of industry-standard XES event logs for subsequent application of conventional process mining techniques. We demonstrate the utility of EDIminer by means of an exemplary EDI-based purchase order process based on real-world data.

Keywords: EDI, process mining, EDIminer, event logs

1 Motivation

Companies and organizations exchange data electronically to perform business transactions (e.g., requests for quotes, purchase orders, etc.). If the interchange of data is carried out in an automated and standardized manner, such processes may be referred to as Electronic Data Interchange (EDI) [1]. In order to understand choreographies, detect bottlenecks or identify scope for improvements, companies are interested in analyzing such transactions. Recently, methods for *process mining* of inter-organizational business processes from observed EDI message exchanges have been proposed [3]. Such methods deal with the generation of event logs from collections of EDI messages for enabling the subsequent application of conventional process mining techniques. Thereby, it is necessary to decide

what EDI artifacts constitute events and how to populate the events' attributes. In particular, the *activity*, *timestamp* and *resource* attributes are of importance for the subsequent application of process mining techniques. Moreover, events need to be correlated to process instances (cases) [4, p.113], which in the case of EDI may pose a significant challenge [3]. With regard to event generation from EDI messages, one can distinguish between *message flow mining* (MFM) and *physical activity mining* (PAM), as described in the following.

Message Flow Mining (MFM). In the approach presented in [3], each sent or received EDI message is considered to represent one event[3]. The event's *timestamp*, *resource* and *activity* properties are populated according to a message's interchange timestamp, the name of the interchange-initiating party and the message type, respectively. For example, an *order* message is interpreted as an activity "Send order" in the corresponding inter-organizational business process. The business data inside the EDI messages is generally ignored for the purpose of generating sets of events. Only for subsequent correlation of events to process instances, business data from the messages is processed. In MFM, the generation of events from EDI messages can generally be performed in a fully automated fashion because the mapping of EDI messages to events and their attributes follows the simple rules described above.

Physical Activity Mining (PAM). Business information conveyed in EDI messages may be used to infer additional events. For example, an *invoice* message may, in addition to general invoicing information, contain information about a shipping date of invoiced line items. Consequently, from such a particular shipping date one may infer that an activity "Ship goods" has occurred on that date. Events resulting from such information generally reflect activities that represent physical product flows, cash flows or other activities as opposed to message flows. Hence, we refer to such approaches as *physical activity mining* (PAM). Thereby, a major challenge is to identify and define appropriate *mappings of business information in EDI messages to events and their attributes in event sequences ("EDI/event mappings")*. Such mappings need to specify rules that define (i) what EDI artifacts constitute events and (ii) which EDI artifacts shall be used to populate event attributes. In this paper we illustrate PAM by means of manual mapping definitions. As in MFM, correlation of so-created events to process instances can pose a challenge in PAM as well.

We propose that MFM and PAM may be combined for process mining from EDI messages in order to arrive at detailed event logs. Such event logs may allow for the mining of detailed process models, performance analyses, data-aware conformance checking, and identification of bottlenecks. For example, a process model mined from fine-grained shipping dates documented in an *invoice* message may lead to insights about specific products that cause regular delays in the invoicing process. In this paper, we present a toolset named *EDIminer* that enables both MFM and PAM from EDI messages.

[3] More precisely, each sent or received EDI message is considered to represent at most one event per process instance, but can potentially map to multiple process instances.

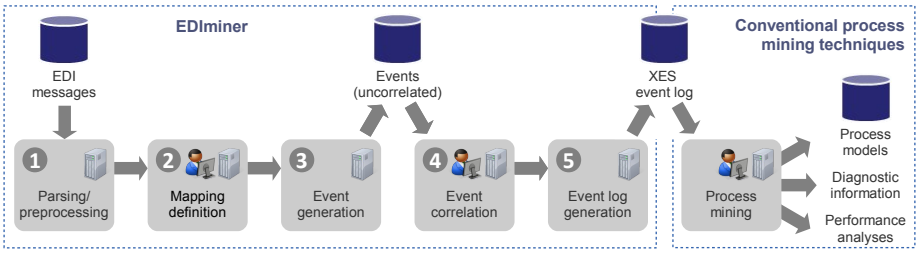

Fig. 1. Processing flow in EDIminer

2 EDIminer

EDIminer is a toolset implemented as a stand-alone Java Swing application that allows a user to perform the following tasks (cf. Fig. 1; cf. Sections 2.1-2.5):

1. Parse, preprocess and visualize a set of EDI messages using an enhanced visualization method based on semantic technologies (automatic, Mark ❶)
2. Automatically (for MFM) or manually (for PAM) define EDI/event mappings (Mark ❷)
3. Execute the defined mappings on the parsed EDI messages to generate events (automatic, Mark ❸)
4. Correlate generated events to process instances (semi-automatic, Mark ❹)
5. Export correlated events to an XES event log (automatic, Mark ❺)

After event log generation, conventional process mining techniques may be applied using existing tools such as ProM (`http://www.processmining.org`).

Running Example. In the remainder of this paper, we provide a proof of concept as well as illustrate the use of EDIminer by means of a running example that is based on real-world EDI data of an Austrian company from the consumer goods sector. For the sake of confidentiality, we will further on refer to this company as *SupplierCo*. The mentioned EDI data set is a snapshot of 466 inbound EDIFACT ORDERS (order) messages and 427 outbound EDIFACT INVOIC (invoice) messages collected between May 2012 and March 2013. It is related to a purchase order process which, according to the explanation of a company representative, consists of the following sequential activities (cf. Fig. 2):

1. The process is initiated when a customer sends an order message to SupplierCo, who receives and processes the order.
2. SupplierCo ships the ordered goods. If this can't be done all at once, the shipments are partitioned.
3. For each shipment, SupplierCo sends a corresponding invoice message to the customer. Usually, no invoices are sent before all ordered goods have been shipped. In addition to general invoicing data, the message contains information about the shipping date of the corresponding goods.

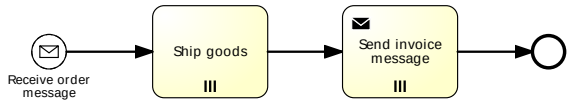

Fig. 2. BPMN model of the purchase order process from the viewpoint of SupplierCo

2.1 EDI Message Parsing, Semantic Preprocessing and Visualization

In order to enable user-driven mapping of EDI artifacts to events and process instances, the contents of EDI messages first have to be visualized to the user. A central question in this regard is what kind of visualization can be considered useful for this purpose. We argue that naive approaches of visualizing and mapping (only) plain data elements from traditional EDI standards will likely not lead to viable results for the purpose of creating event logs. The reason for this is that semantically accurate interpretation and visualization of EDI messages based on standards such as EDIFACT or X12 are non-trivial due to the existence of so-called qualified and non-qualified data elements in these standards. For example, in various EDIFACT standards, such as release D.01B, data element 3039 (*Party identifier*), found in NAD (*Name and address*) segment[4] instances, may have a plethora of different concrete semantics (e.g., buyer's party identifier, seller's party identifier, etc.) depending on the value of a corresponding qualifying code in data element 3035 (*Party function code qualifier*)[5]. We refer to the different concrete semantics of a data element as *data element variants* (DEVs). For example, an instance of data element 3039 qualified by the code "BY" (*Buyer*) in an instance of data element 3035 represents one DEV of data element 3039 - this DEV could be labeled, for instance, 'Buyer_PartyIdentifier'.

In EDIminer, we tackled the problem of handling qualification relationships by implementing the approach described in [2] for representing EDI standards, qualification relationships and concrete messages in ontologies and knowledge bases. EDIminer ships with predefined ontologies for various EDI standards[6]. When EDIminer is launched, the user is asked to point to a folder that contains a set of EDI messages. These messages are subsequently parsed into message knowledge bases. Thereby, DEVs are automatically identified through reasoning over qualification relationships as described in [2].

The employed ontological approach allows for the visualization of the contents of EDI messages in a tree structure of "plain" data elements *and* DEVs, as illustrated on the left hand side of Fig. 3. The tree entries in red italics represent DEVs. The hierarchichal structure of the tree entries resembles the corresponding message type specifications according to the underlying EDI standards. We consider this kind of visualization an extension to the state-of-the-art in EDI mapping tools, which are generally not capable of visualizing variants of data elements, except at most for limited subsets of specific EDI standards [2].

[4] `http://www.unece.org/trade/untdid/d01b/trsd/trsdnad.htm`

[5] `http://www.unece.org/trade/untdid/d01b/tred/tred3035.htm`

[6] These ontologies have been created based upon the official UN/EDIFACT directories using additional heuristics for the identification of qualification relationships.

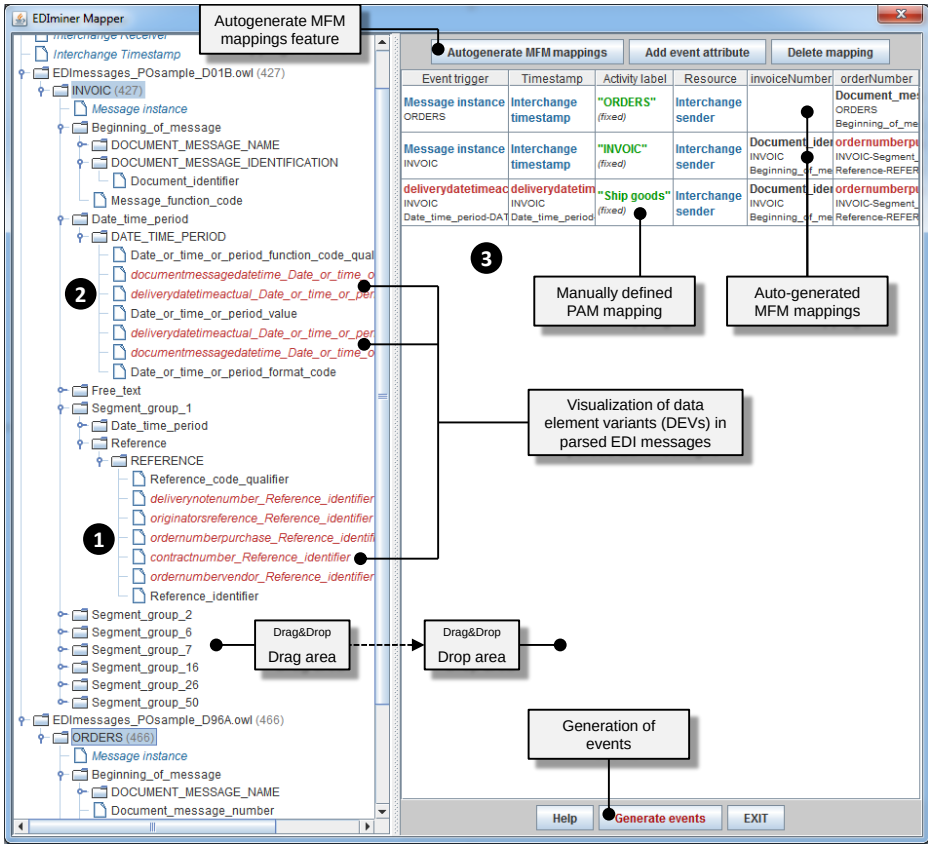

Fig. 3. Screenshot of EDIminer's mapping GUI

SupplierCo Example. As shown in the tree on the left hand side of Fig. 3, the contained data elements/DEVs of the parsed ORDERS and INVOIC messages are displayed to the user. Thereby, DEVs such as *order number reference identifiers* (Mark ❶) and *actual delivery dates* (Mark ❷) are visualized in addition to corresponding "plain" data elements, such as *reference identifier* and *date or time or period value*, respectively.

2.2 Definition of EDI/Event Mappings

Having this visualization of the parsed EDI messages at hand, the next step is to define EDI/event mappings specifying (i) what EDI artifacts constitute events and (ii) which EDI artifacts shall be used to populate event attributes. In the table shown on the right hand side of Fig. 3 (Mark ❸), each row represents a mapping rule. The first column ("Event trigger") specifies which values of data elements/DEVs in messages shall trigger the creation of events. The remaining columns specify the values of data elements/DEVs that shall be used to populate

the attributes of these events, such as *timestamp*, *activity* and organizational *resource*. Additional event attributes may be added by a user through adding columns to the table using the "Add event attribute" button. In addition to data elements/DEVs, also user-defined fixed strings and some special variables (e.g., interchange meta-data) may be used in mappings.

To automatically define mapping rules for the purpose of MFM, users can click the "Autogenerate MFM mappings" button. This leads to the creation of one mapping rule per parsed message type where the event trigger is set to the special variable *message instance* (i.e., every parsed EDI message triggers an event), the *timestamp* attribute is set to the timestamp of the message interchange, the *activity* attribute is set to the name of the message type (e.g., "INVOIC") and the *resource* attribute is set to the name of the party who initiated the message interchange. Such auto-generated mappings resemble the MFM approach described in [3]. To manually define individual mapping rules for the purpose of PAM, users can drag and drop data elements/DEVs or special variables from the tree on the left hand side to cells of the mapping table, or enter user-defined fixed strings directly into a cell.

SupplierCo Example. For SupplierCo's purchase order process we define combined mappings for MFM and PAM. By using the "Autogenerate MFM mappings" feature, two mappings (for ORDERS and for INVOIC messages) are created automatically (first two mappings in Fig. 3). A third PAM mapping is defined manually such that the creation of events is triggered by occurrences of *actual delivery dates* in INVOIC messages (third mapping in Fig. 3). These events' *timestamps* are set to these same delivery dates and their *activity* labels are set to the user-defined fixed string "Ship goods".

With a view to facilitating subsequent event correlation, we manually add an additional event attribute *invoiceNumber* that is applicable for the two INVOIC-based mappings and populate it with values of *document identifiers*. Additionally, we add an attribute *orderNumber* to all three mappings. In the ORDERS-based mapping, this attribute is populated by values of *document/message number*. For the other two mappings, it is populated by values of *order number reference identifier*.

2.3 Generation of Events

Once the user has completed the definition of EDI/event mappings, EDIminer can be instructed to automatically generate a set of events from the parsed EDI messages. However, since a single EDI message may contain multiple instantiations of identical data elements/DEVs at different positions, the question arises of how to decide which specific event-triggering values of data elements/DEVs are to be related to which specific attribute-populating values. In EDIminer, a heuristic method is used to arrive at reasonable matchings; due to space limitations, however, we refrain from describing this method in detail. Generally speaking, the closer an attribute-populating value is located to an event-triggering value in the hierarchical structure of segment group instances in a message, the more likely it is used for populating the corresponding event's attributes.

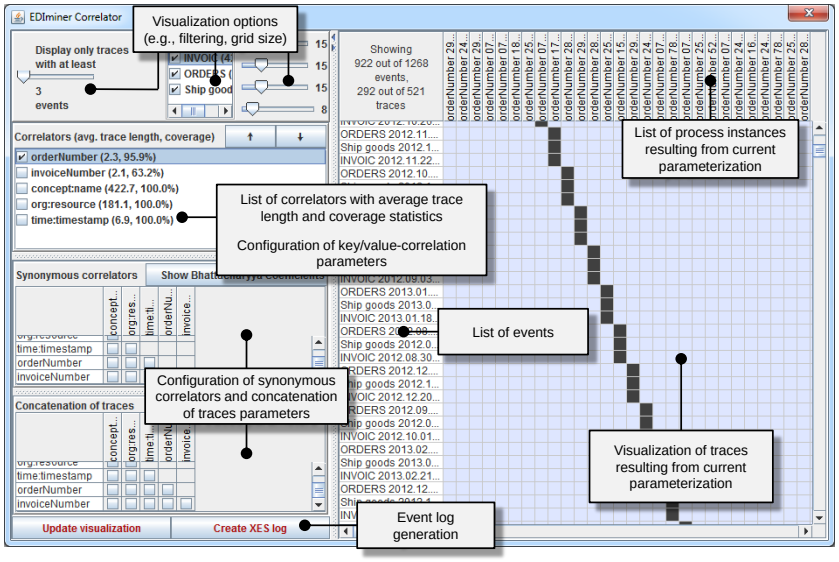

Fig. 4. Screenshot of EDIminer Correlator

2.4 Event Correlation

Correlation of events originating from EDI messages to process instances may pose a significant challenge [3]. EDIminer implements the event correlation algorithm described in [3] and allows a user to define the parameterization for the algorithm in a GUI as shown in Fig. 4. While the user is parameterizing the algorithm with correlation rules, he/she is provided with instantly updated visualizations of the resulting process instances. The user can iteratively refine the parameterization until he/she is satisfied with the results. Furthermore, the GUI shows various statistics for individual correlators, including average trace length, correlator coverage as percentage of events, and value overlap with other correlators. Based upon these statistics, EDIminer makes suggestions to aid the user in defining suitable correlation rules. For instance, data attributes with high average trace lengths and high coverages are suggested for key/value-correlation; pairs of correlators with a high value overlap are suggested as being synonymous.

SupplierCo Example. For the set of events generated from the SupplierCo example, events can be correlated by key/value-correlation using the previously defined *orderNumber* event attribute. A visualization of the resulting traces (filtered by a minimum trace length of three events) is shown in Fig. 4.

2.5 Generation of XES Event Logs

Once the events are correlated to process instances, EDIminer can be instructed to export them to an XES event log [5]. In doing so, EDIminer uses a version of the OpenXES libraries (`http://www.xes-standard.org/openxes`). Such event logs may in turn be fed into existing process mining tools for subsequent analyses.

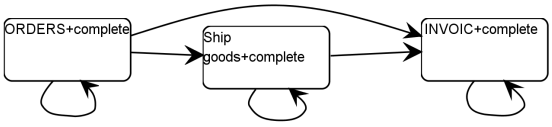

Fig. 5. Process model mined from SupplierCo's EDI data using MFM and PAM

SupplierCo Example. For the event log generated from the SupplierCo example, ProM's Heuristic Miner (default configuration) outputs the process model shown in Fig. 5. The original purchase order process is clearly visible in the mined process model. Note the "Ship goods" activity that was discovered using the manually defined PAM mapping rule. The exact reasons for the loop on the ORDERS activity and the optional omission of the "Ship goods" activity in the mined process model are yet unknown; however, a thorough analysis of this particular case is beyond the scope of this paper.

3 Conclusion

In this paper we presented EDIminer, a software toolset that enables process mining from EDI messages supporting two different methodologies: (i) *message flow mining* (MFM) and (ii) *physical activity mining* (PAM). EDIminer employs semantic technologies to provide an enhanced visualization of contents of EDI messages and allows users to automatically or manually define EDI/event mappings. These mappings are used to automatically generate events, which are subsequently correlated to process instances (cases) using a semi-automatic approach. The correlated events can be exported to XES event logs and used with existing process mining tools.

Our future research plans focus on conducting an extended evaluation of EDIminer, including an evaluation and potential extension of the MFM and PAM methodologies, by means of case studies using real-world EDI data from different industries.

Acknowledgment. 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.

References

1. M. A. Emmelhainz. *EDI: A Total Management Guide.* John Wiley & Sons, 1992.
2. R. Engel, C. Pichler, M. Zapletal, W. Krathu, and H. Werthner. From Encoded EDIFACT Messages to Business Concepts Using Semantic Annotations. In *14th IEEE Int.Conf. on Commerce and Enterprise Comp. 2012*, pp.17-25. IEEE, 2012.
3. R. Engel, W. van der Aalst, M. Zapletal, C. Pichler, and H. Werthner. Mining Inter-organizational Business Process Models from EDI Messages: A Case Study from the Automotive Sector. In *CAiSE'12*, LNCS 7328, pp.222-237. Springer, 2012.
4. W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.
5. XES. XES Extensible Event Stream. http://www.xes-standard.org/.