Technical University of Vienna
Information Systems Institute
Distributed Systems Group

Mining of ad-hoc business processes with TeamLog

Schahram Dustdar, Thomas Hoffmann
and Wil van der Aalst
dustdar@infosys.tuwien.ac.at
thomas.hoffmann@onemail.at
w.m.p.v.d.aalst@trn.tue.nl

TUV-1841-2004-07 July 23, 2004

The design of workflows is a complicated task. In practice, the actual workflow processes will often differ from the processes as perceived by the management. Process mining supports the design and improvement of processes using transaction logs, which contain information about the actual process executions. This paper introduces basics about process mining, a common workflow log format and it gives an example: mining ad-hoc processes of Caramba – a process-aware collaboration system. It is discussed how Caramba-specific process information is converted to a common format using an application called TeamLog. Then EMiT – a process mining tool – mines the converted process information.

Keywords: Process Mining, Workflow Management, Business Process Management, Business Process Analysis, TeamLog, EMiT, MinSoN

Argentinierstr. 8/184-1
A-1040 Vienna, Austria
phone: +43 1 58801-18402
fax: +43 1 58801-18491
URL: http://www.infosys.tuwien.ac.at/

Mining of ad-hoc business processes with TeamLog

Schahram Dustdar, Thomas Hoffmann
Distributed Systems Group, Vienna University of Technology
{dustdar@infosys.tuwien.ac.at | thomas.hoffmann@onemail.at}
Wil van der Aalst
Information Systems, Eindhoven University of Technology
w.m.p.v.d.aalst@tm.tue.nl

Abstract. The design of workflows is a complicated task. In practice, the actual workflow processes will often differ from the processes as perceived by the management. Process mining supports the design and improvement of processes using transaction logs, which contain information about the actual process executions. This paper introduces basics about process mining, a common workflow log format and it gives an example: mining ad-hoc processes of Caramba – a process-aware collaboration system. It is discussed how Caramba-specific process information is converted to a common format using an application called TeamLog. Then EMiT – a process mining tool – mines the converted process information.

Keywords: Process Mining, Workflow Management, Business Process Management, Business Process Analysis, TeamLog, EMiT, MinSoN

1 Introduction

Process management in organizations becomes more and more important. To increase their competitiveness they have to introduce clearly defined processes, and these processes must be improved continuously. Actual work can deviate from process definitions due to many reasons. Therefore it is important for organizations to discover these differences in order to improve their processes. Process mining allows both (1) the identification of processes from transaction logs and (2) the detection of deviations between the (prescriptive or descriptive) process model and the real world process executions. Ad-hoc processes, a special category of processes, have no underlying process definition. Mining this kind of processes offers important information for the organization's management, which can be used to detect the actual processing behavior and therefore, to improve the organizations performance. For example, ad-hoc processes arise from loosely coupled collaboration between domain-specialists across geographical distances and organizational boundaries. Caramba [12] is one of the few collaboration systems actually supporting ad-hoc processes. This paper describes mining of ad-hoc processes by means of Caramba, TeamLog and EMiT [11]. Caramba offers a transaction log and TeamLog converts it into a general format. Then EMiT mines the process and creates a process model in forms of a Petri

net. Additionally other tools such as MinSoN can mine other aspects such as the organizational context.

Why is all of this relevant? Thus far, information technology has been focusing on two extremes: completely *unstructured processes* and highly *structured processes*. On one end of the spectrum we find groupware products that typically aim at supporting unstructured processes, i.e., these systems are completely unaware of the business process at hand. For example, e-mail programs support the exchange of information but are completely ignorant when it comes to the purpose of these information exchanges in context of business processes. If we ignore workflow components such as Domino Workflow, even advanced groupware products such as Lotus Notes, are unaware of processes and therefore, implicitly, assume unstructured processes. On the other end of the spectrum, we find traditional workflow offerings such as Staffware, MQSeries Workflow, etc. These systems definitely are aware of the business process at hand. In fact, they typically force the execution of these processes into structures defined at design time. Classical workflow technology has problems dealing with flexibility and therefore only supports highly structured processes. In reality, most business processes are in-between these two extremes, i.e., they are *semi-structured*. In semi-structure processes there may be typical patterns in the handling of cases, but for individual instances it is desirable, or even necessary, to deviate from these patterns. One way of dealing with these processes is the *case-handling paradigm* [1], i.e., resort to a more data-driven approach where in addition to the normal workflow, implicitly, alternative routes (e.g., "bypass" and "redo") are generated. Another way is to let end-user, at run-time, design or modify the process for one or more process instances. This is sometimes referred to as *ad-hoc workflow*, i.e., the process emerges or is modified at run-time by the end-users.

Dealing with ad-hoc processes in an adequate way is important to improve the support of business processes. Technologies aiming at completely *unstructured processes* (e.g., groupware) and highly *structured processes* (e.g., production workflow) have in a way been focusing on the "low hanging fruits", i.e., processes where it is easy to improve performance with relatively simple measures. Unfortunately, the majority of processes do not fall into one of these extreme categories. Therefore, concepts and tools aiming at ad-hoc business processes are of the utmost importance.

Ad-hoc business processes require more flexibility than traditional "production workflow" type of processes. However, at the same time, there is the need for management information and real insight into the actual processes as they unfold. Unfortunately, there is not a clearly defined process model that can serve as an anchor to gather and present process information. Therefore, we propose the use of *process mining* to analyze the logs of systems supporting ad-hoc processes. In this paper we demonstrate that this is actually possible using the existing ad-hoc collaboration system Caramba and the process mining tool EMiT. The result is a generic approach for mining ad-hoc business processes and a concrete tool linking Caramba, EMiT, and other process mining tools.

The remainder of this paper is organized as follows. First we discuss related work. Section 3 then gives an overview of process mining. The common workflow log format used by our approach is covered by Section 4. Caramba and its data model are discussed in Section 5. Furthermore, this section contains an example of an ad-hoc

process. The next section presents TeamLog – a format converter of Caramba process information. Section 7 contains an example of process mining with EMiT. Finally, in Section 8 we summarize the main conclusions and give an outlook on our future work.

2 Related work

Recently, the topic of process mining has been gaining more attention both in practice and research [5,6]. Gartner identifies *Business Process Analysis* (BPA) as an important aspect of the next generation of BPM products [14]. Note that BPA covers aspects neglected by traditional workflow products (e.g., diagnosis, simulation, etc.). *Business Activity Monitoring* (BAM), which can be considered as a synonym to process mining, is named by Gartner as one of the emerging areas in BPA [14]. The goal of BAM tools is to use data logged by the information system to diagnose the operational processes. An example is the ARIS Process Performance Manager (PPM) of IDS Scheer [17]. ARIS PPM extracts information from audit trails (i.e., information logged during the execution of cases) and displays this information in a graphical way (e.g., flow times, bottlenecks, utilization, etc.). Many other vendors offer similar products, e.g., Cognos (NoticeCast), FileNet (Process Analyzer), Hyperion (Business Performance Management Suite), Tibco (BusinessFactor), and webMethods (Optimize/Dashboard). These tools show the practical relevance of process mining. Unfortunately, these tools only focus on measuring performance indicators such as flow time and utilization and do not at all focus on discovering the process and its organizational context. For example, none of the tools mentioned actually discovers causal relations between various events or the underlying social network. Moreover, the focus of these systems is on well-defined processes and they are unable to handle ad-hoc business processes. Note that for ad-hoc business processes it is not sufficient to look into performance indicators such as flow time and utilization, i.e., it is vital to have insight in the actual processes as they unfold, emerge, and/or change.

Also in academia there is a growing interest in process mining as is illustrated by special issues of journals, workshops, and papers [2,4,5,6,7,8,10,11,15,16,23,24,27,29]. Different papers focus on different perspectives of process mining. The most challenging perspective remains the control-flow perspective. Especially for ad-hoc business processes, there are notorious, but interesting, problems. For example, “How to distinguish noise/exceptions from regular behavior?”, “How to discover duplicate/hidden tasks in a process?”, etc. [23].

This paper focuses on mining ad-hoc business processes. As indicated in the introduction, there are few tools supporting ad-hoc business processes, i.e., semi-structured processes. Most research efforts have been focusing at completely unstructured processes (e.g., groupware) and highly structured processes (e.g., production workflow). The literature on workflow is extensive [3,18,21,22] and within this domain several people have been working on “workflow change” [9,13,25]. However, most of the work is devoted to migrating an instance from one (structured) workflow model to another rather than focusing on truly ad-hoc

processes. Caramba is one of the few process-aware collaboration systems that truly supports ad-hoc business processes [12].

3 Process mining

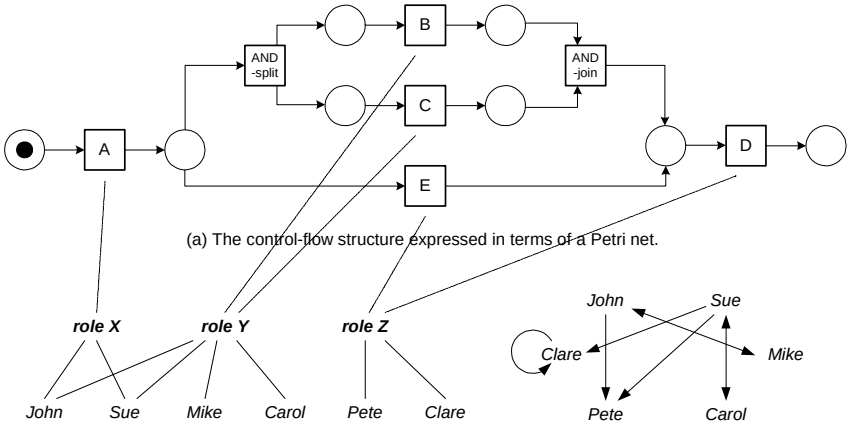
The goal of process mining is to extract information about processes from transaction logs [5,7]. We assume that it is possible to record events such that (i) each event refers to an *activity* (i.e., a well-defined step in the process), (ii) each event refers to a *(work)case* (i.e., a process instance), (iii) each event can have a performer also referred to as *originator* (the person executing or initiating the activity), and (iv) events have a *timestamp* and are totally ordered. Table 1 shows an example of a log involving 19 events, 5 activities, and 6 originators. In addition to the information shown in this table, some event logs contain more information on the case itself, i.e., data elements referring to properties of the case.

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

Table 1: An event log.

Event logs such as the one shown in Table 1 are used as the starting point for mining. We distinguish three different perspectives: (1) the process perspective, (2) the organizational perspective and (3) the case perspective. The *process perspective* focuses on the control-flow, i.e., the ordering of activities. The goal of mining this perspective is to find a good characterization of all possible paths, e.g., expressed in terms of a Petri net [26] or Event-driven Process Chain (EPC) [17,19]. The *organizational perspective* focuses on the originator field, i.e., which performers are involved and how are they related. The goal is to either structure the organization by

classifying people in terms of roles and organizational units or to show relation between individual performers (i.e., build a social network [28]). The *case perspective* focuses on properties of cases. Cases can be characterized by their path in the process or by the originators working on a case. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order it is interesting to know the supplier or the number of products ordered.



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

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

(c) A sociogram based on transfer of work.

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

The process perspective is concerned with the "How?" question, the organizational perspective is concerned with the "Who?" question, and the case perspective is concerned with the "What?" question. To illustrate the first two consider Figure 1. The log shown in Table 1 contains information about five cases (i.e., process instances). The log shows that for four cases (1, 2, 3, and 4) the activities A, B, C, and D have been executed. For the fifth case only three activities are executed: activities A, E, and D. Each case starts with the execution of A and ends with the execution of D. If activity B is executed, then also activity C is executed. However, for some cases activity C is executed before activity B. Based on the information shown in Table 1 and by making some assumptions about the completeness of the log (i.e., assuming that the cases are representative and a sufficient large subset of possible behaviors is observed), we can deduce the process model shown in Figure 1(a). The model is represented in terms of a Petri net [26]. The Petri net starts with activity A and finishes with activity D. These activities are represented by transitions. After executing A there is a choice between either executing B and C in parallel or just executing activity E. To execute B and C in parallel two non-observable activities (AND-split and AND-join) have been added. These activities have been added for routing purposes only and are not present in the event log. Note that for this example we assume that two activities are in parallel if they appear in any order. By

distinguishing between start events and complete events for activities it is possible to explicitly detect parallelism.

Figure 1(a) does not show any information about the organization, i.e., it does not use any information on the people executing activities. However, Table 1 shows information about the performers. For example, we can deduce that activity A is executed by either John or Sue, activity B is executed by John, Sue, Mike or Carol, C is executed by John, Sue, Mike or Carol, D is executed by Pete or Clare, and E is executed by Clare. We could indicate this information in Figure 1(a). The information could also be used to "guess" or "discover" organizational structures. For example, a guess could be that there are three roles: X, Y, and Z. For the execution of A role X is required and John and Sue have this role. For the execution of B and C role Y is required and John, Sue, Mike and Carol have this role. For the execution of D and E role Z is required and Pete and Clare have this role. For five cases these choices may seem arbitrary but for larger data sets such inferences capture the dominant roles in an organization. The resulting "activity-role-performer diagram" is shown in Figure 1(b). The three "discovered" roles link activities to performers.

Figure 1(c) shows another view on the organization based on the transfer of work from one individual to another, i.e., not focus on the relation between the process and individuals but on relations among individuals (or groups of individuals). Consider for example Table 1. Although Carol and Mike can execute the same activities (B and C), Mike is always working with John (cases 1 and 2) and Carol is always working with Sue (cases 3 and 4). Probably Carol and Mike have the same role but based on the small sample shown in Table 1 it seems that John is not working with Carol and Sue is not working with Carol. (Clearly the number of events in Table 1 is too small to establish these assumptions accurately. However, for the sake of argument we assume that the things that did not happen will never happen.) These examples show that the event log can be used to derive relations between performers of activities, thus resulting in a sociogram. For example, it is possible to generate a sociogram based on the transfers of work from one individual to another as is shown in Figure 1(c). Each node represents one of the six performers and each arc represents that there has been a transfer of work from one individual to another. The definition of "transfer of work from A to B" is based on whether for the same case an activity executed by A is directly followed by an activity executed by B. For example, both in case 1 and 2 there is a transfer from John to Mike. Figure 1(c) does not show frequencies. However, for analysis proposes these frequencies can be added. The arc from John to Mike would then have weight 2. Typically, we do not use absolute frequencies but weighted frequencies to get relative values between 0 and 1. Figure 1(c) shows that work is transferred to Pete but not vice versa. Mike only interacts with John and Carol only interacts with Sue. Clare is the only person transferring work to herself.

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

example be possible two show a positive correlation between the size of an order or its handling time and the involvement of specific people.

Orthogonal to the three perspectives (process, organization, and case), the result of a mining effort may refer to *logical* issues and/or *performance* issues. For example, process mining can focus on the logical structure of the process model (e.g., the Petri net shown in Figure 1(a)) or on performance issues such as flow time. For mining the organizational perspectives, the emphasis can be on the roles or the social network (cf. Figure 1 (b) and (c)) or on the utilization of performers or execution frequencies.

To address the three perspectives and the logical and performance issues we have developed a set of tools including EMiT [2,11], Thumb [29], and MinSoN [4]. These tools share a common XML format, as described in the next section. Through this format we will try to mine ad-hoc processes based on the logs in systems such as Caramba.

4 Workflow logs

Many process-aware collaboration systems store information in application-specific formats. Thus a general format would simplify process analysis with common tools like EMiT [2,11]. [11] gives a definition of a general XML workflow log format. This section contains a simple ad-hoc process and its general XML description to show how a workflow log looks like. The general formats DTD is discussed in Section 6.4. We assume that the example-ad-hoc-process (Figure 2) belongs to the real world process "plan IT-installation for offices (banks)".

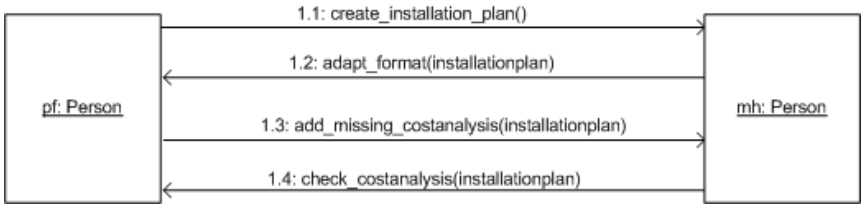


Figure 2: A simple ad-hoc process.

Two persons collaborate in this process (Table 1).

Abbreviation	Full name	Department
mh	Marta Huchen	Consulting
pf	Peter Fogosch	Sales department

Table 2: Involved persons in the ad-hoc process.

Customer Trust Bank sends a request for an IT-installation plan of a new office to *pf*. *pf* instructs *mh* to create the requested plan (fig. 2, collaboration 1.1). When *mh* completes the plan she sends this document back to *pf* and asks him to verify/change the document formatting (collaboration 1.2). After *pf* has received the installation plan, he notices that the cost analysis is missing. So he tells *mh* to add the missing details (collaboration 1.3). *mh* adds the cost analysis and returns the plan to *pf*, who has to check the updated plan (collaboration 1.4). The code below describes this simple ad-hoc-process using the general workflow log format.

```
<WorkFlow_log>
    <source program="other"/>
    <process description="none" id="process_1">
        <case description="New office (Trust Bank)" id="id_1">
            <log_line>
                <task_name>Create installation plan</task_name>
                <event kind="normal"/>
                <date>13-02-2004</date>
                <time>11:14:24</time>
                <originator>Fogosch Peter</originator>
            </log_line>
            <log_line>
                <task_name>Adapt the format of the installation
plan</task_name>
                <event kind="normal"/>
                <date>14-02-2004</date>
                <time>08:13:01</time>
                <originator>Huchen Marta</originator>
            </log_line>
            <log_line>
                <task_name>Add missing cost analysis</task_name>
                <event kind="normal"/>
                <date>14-02-2004</date>
                <time>14:50:21</time>
                <originator>Fogosch Peter</originator>
            </log_line>
            <log_line>
                <task_name>Check cost analysis</task_name>
                <event kind="normal"/>
                <date>16-02-2004</date>
                <time>12:05:30</time>
                <originator>Huchen Marta</originator>
            </log_line>
        </case>
    </process>
</WorkFlow_log>
```

This application-independent process description facilitates further process analysis and can be applied to ad-hoc business processes. The remainder of this paper demonstrates this by mining ad-hoc processes supported by a concrete collaboration system: Caramba [12].

5 Caramba

5.1 Overview and definitions

Caramba is a process-aware collaboration system which helps people to collaborate across geographical distances and organizational boundaries. It was implemented to improve the effectiveness of collaboration in virtual teams [12]. [12] lists special properties of virtual teams:

- team members require status information on all work activities (process-awareness)
- team members are frequently imbedded in different organizations and they have to collaborate across multiple business processes, time zones and locations
- virtual teams work on highly dynamic ad-hoc processes which require interaction of many domain experts
- members of virtual teams jointly work on artifacts (documents, databases,…)
- team members require knowledge about the multiple relationships between artifacts and the context in which they were created, shared or distributed (e.g. who, what, when, in which context).
- team leaders of virtual teams need on-demand access on the projects status and artifacts and on critical communication (team-internal or between team members and customers).

Caramba offers support for predefined and ad-hoc processes. Predefined processes are modeled at design time, whereas ad-hoc processes result from runtime-collaboration between the involved persons. Therefore, ad-hoc processes do not have an a-priori known process definition. Table 3 contains some important definitions.

Term	Explanation
business process	A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [20].
process instance	The representation of a single enactment of a process [20], also referred to as (work)case.
process definition	The representation of a business process in a form which supports automated manipulation, such as modeling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc. [20]
(Work)case	In this paper we use it as synonym for process instance.
Activity	A description of a piece of work that forms one logical step within a process. [20]
Coordination	Process of message creation. A coordination has a sender and an addressee (the term addressee will be discussed in Section 5.2). A coordination causes the generation of activity instances for the sender and for each recipient. [20]

Table 3: Definitions.

Data model

To be able to describe the functionality and further technical TeamLog issues in the next sections, it is necessary to explain some basics about Caramba's data model. Caramba stores its data in a relational database that manages the Caramba objects. Although the database contains many tables this section focuses only on CARAMBA_AI which holds the activity instance information required by TeamLog. Figure 3 lists the TeamLog-relevant attributes of CARAMBA_AI.

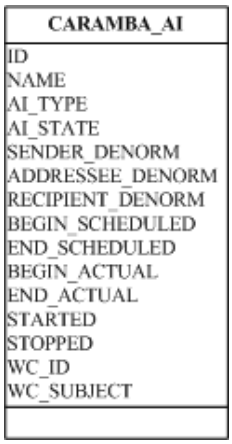


Figure 3: Table CARAMBA_AI: TeamLog-relevant attributes.

ID is a unique number which serves as primary key. The attribute *NAME* contains the task description (e.g. "Verify installation plan"). Caramba distinguishes between different record types (*AI_TYPE*). A "W"-record (workcase) indicates the creation of a new workcase. Caramba adds a "C"-record (coordination) for each coordination. Additionally to that, Caramba inserts an "S"-record (source) for each message sent to an addressee, i.e. for every "S"-record there are one ore more "C"-records. The example in Section 5.2 offers details about the notion of addressees and shows when Caramba creates the different kinds of records. *AI_STATE* holds the record-status: "N" marks a coordination as new, "R" indicates a coordination which was already read and "D" (done) means that a coordination is completed. Caramba supports further record states which are not relevant for the purpose of this paper. *SENDER_DENORM* is used to store the name of the creator (sender) of a coordination, *ADDRESSEE_DENORM* holds the name of the addressee and *RECIPIENT_DENORM* contains the name of the coordinations recipient. Table CARAMBA_AI stores different kinds of timestamp information. *BEGIN_SCHEDULED* and *END_SCHEDULED* denote a coordinations scheduled start and end time. On the other hand *BEGIN_ACTUAL* and *END_ACTUAL* hold the actual begin and end timestamps. *STARTED* is used to store the creation time of the record, *STOPPED* is set whenever the status (AI_STATE) changes to "D". In contrast to *STARTED* and *STOPPED* which are automatically created by Caramba, the schedule and actual time intervals are under the users control. If a record is a coordination (*AI_TYPE* is "C") then *WC_ID* contains the primary key of the corresponding "W"-record. *WC_SUBJECT* holds the appropriate workcase description.

5.2 Example: ad-hoc process in Caramba

To illustrate the principle of ad-hoc processes in Caramba we consider the collaboration diagram shown in Figure 4. This section extends the example from Section 4 with another workcase (a second process instance of the same real world process). Another bank (Simpson Bank) requests an installation plan for a new office. In addition to the control flow, Figure 4 contains information about the generated database records in table CARAMBA_AI.

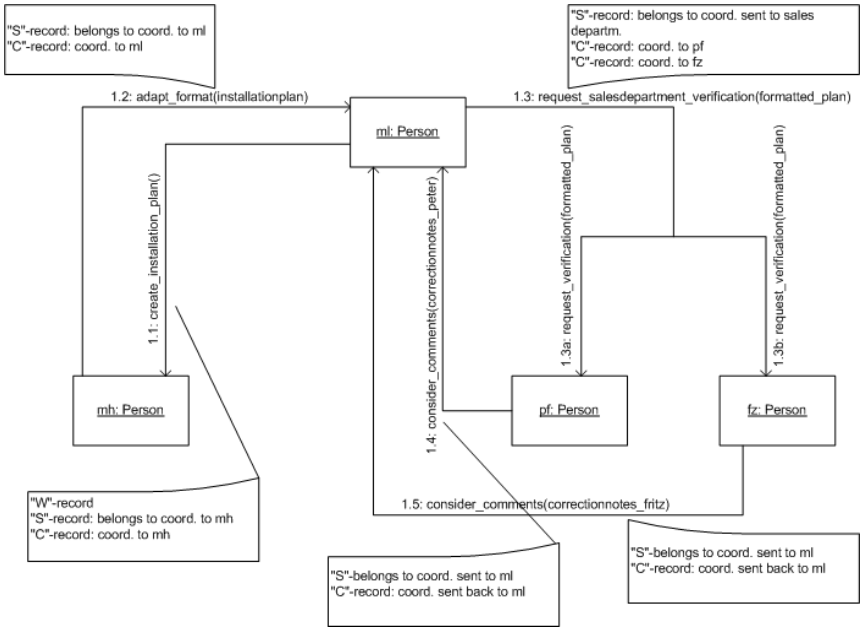


Figure 4: Ad-hoc process: collaboration diagram.

Several persons collaborate during process execution (Table 4).

Abbreviation	Full name	Department
ml	Monika Lachs	Sales department
mh	Marta Huchen	Consulting
pf	Peter Fogosch	Sales department
fz	Fritz Zander	Sales department

Table 4: Involved persons in ad-hoc process.

In this case, customer Simpson Bank sends the request for an IT-installation plan of a new office to *ml*. *ml* instantiates the process "New office (Simpson Bank)" by sending the coordination "Create installation plan" to *mh* (fig. 4, collaboration 1.1). The coordination appears in *mh*'s personal worklist. When *mh* completes the plan she

sends this document back to *ml* and asks her to verify/change the document formatting (collaboration 1.2). After *ml* has revised the formatting she wants the plan to be verified by the members of the sales department. In contrast to the ad-hoc process in Section 4, *ml* might think that verification by multiple person makes sense, because the project costs are much higher. Hence, *ml* instructs Caramba (collaboration 1.3) to send coordinations to the appropriate persons (1.3a and 1.3b). Because *ml* also belongs to the sales department normally she would receive a coordination too. To simplify the example, this coordination from *ml* to *ml* is ignored. Later on, when, *pf* and *fz* have verified the installation plan, they return their correction notes to *ml* (collaboration 1.4 and 1.5).

How does Caramba process a coordination to a group, such as collaboration 1.3 in Figure 4? Caramba's coordination model distinguishes between recipients and addressees. The sender (in our example *ml*) is not allowed to select the recipient of a coordination directly. Instead of that he has to choose an addressee. If the addressee is a person, Caramba does not need to do further conversions. It maps the coordination directly to this person, which means that the addressee is equal to the recipient. But if the user chooses a group as addressee (e.g. sales department), Caramba determines the group members and sends a coordination to each of them. Caramba offers further possibilities [12]: other objects than groups can serve as addressees and it is possible to control a coordinations distribution type (e.g. all members of a group, a random member of a group, etc.).

TeamLog reads the database records shown in Figure 4 to create an XML workflow log. The next section discusses more technical TeamLog issues to offer knowledge about how TeamLog works in principle and how it maps Caramba database information into the XML log.

6 TeamLog

6.1 Motivation and goals

Normally, in current enterprise systems process information is distributed over several "log databases" (e.g. files, relational databases etc.). To support easier processing of this data, it is convenient to collect the relevant process information from the different logs and summarize it in one XML workflow log. TeamLog is a tool which offers appropriate functionality for ad-hoc process information. Furthermore, TeamLog supports anonymous task names in the XML workflow log, because that's practical for general process analysis. It can be configured to read information from different systems. Currently, there are already some tools that support visualization of processes on the basis of workflow logs (e.g. EMiT [11], Thumb [29]). In this paper we will use TeamLog to read process information out of Caramba's database in order to generate XML input for EMiT (Enhanced Mining Tool). In contrast to visualization, at the moment there is only limited tool-support for analysis and interpretation of ad-hoc processes.

To give a principal overview Figure 5 shows the transition from relational ad-hoc process information in CARAMBA to EMiTs output: a Petri-net of the reconstructed workflow.

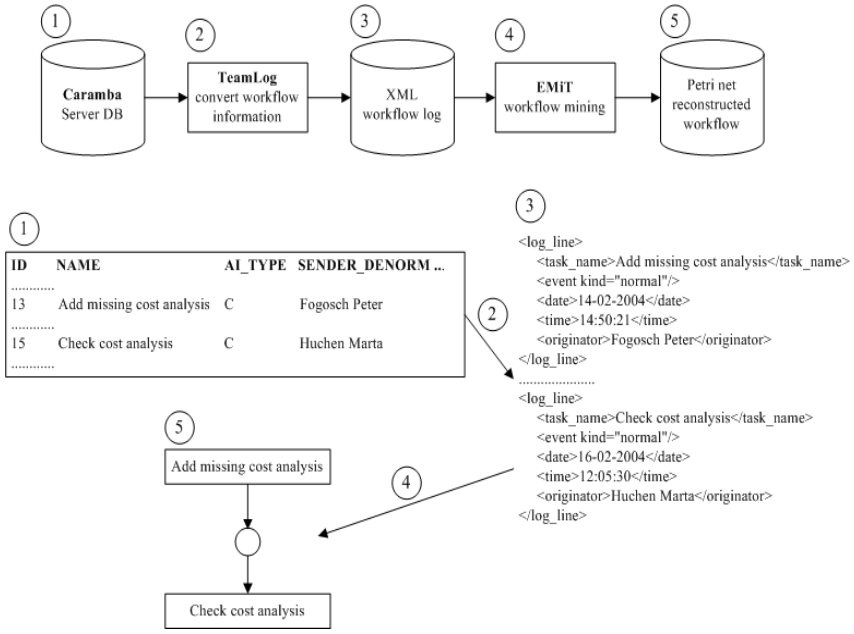


Figure 5: From Caramba database information to the reconstructed workflow.

6.2 Technical overview of TeamLog

TeamLog is a portable standalone application written in Java. To get the required data, TeamLog accesses Caramba's database directly via JDBC[1], loading the communication settings from its property file. The information read from table CARAMBA_AI is filtered, converted and written to an XML-file in the file system.

Three packages are used to organize the applications components: *TeamLog.general* contains classes/interfaces for general use, *TeamLog.logic* includes business logic (see Figure 6) and external interface components and *TeamLog.ui* holds all classes which implement the user interface. This section offers information about the components functionality and their purpose. If the type (class, interface) of a component is not explicitly stated, then it's a class, not an interface. Figure 6 provides a high-level overview about TeamLog's component organization.

[1] See JDBC Technology Overview, http://java.sun.com/products/jdbc/.

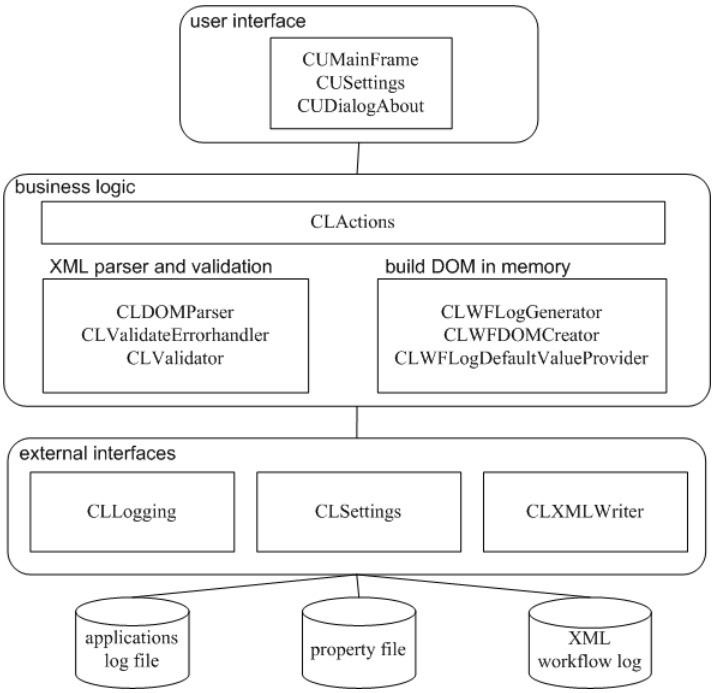


Figure 6: Overview of TeamLog's classes.

TeamLog.general
TeamLogStart is the main class which starts the application. The interface *IConstants* is used to enable common access to application-wide constants. *ETeamLog* implements a user-defined exception that is thrown whenever an error occurs that must be forwarded to a calling method/component. Another class for general use is *ContentDefinition*, which purpose is to describe the content of the requested workflow-log.

TeamLog.logic
This package comprises 10 classes that implement the business logic. *CLActions* serves as interface-component between user interface and business logic. Because TeamLog creates XML-output validation is necessary. The classes which implement the appropriate parser are *CLDOMParser* and *CLValidateErrorhandler*. *CLValidator* uses this parser to offer high-level methods for XML workflow log validation. In order to record TeamLogs behaviour the class *CLLogging* is used to add information to the applications log file. *CLSettings* provides access to TeamLogs property file, which contains application settings. To create XML output TeamLog has to build the XML tree in memory (DOM). Three classes provide the functionality for that: *CLWFDOMCreator*, *CLWFLogDefaultValueProvider* and *CLWFLogGenerator*.

CLWFLogGenerator gets a *ContentDefinition* (package TeamLog.general), reads information from table CARAMBA_AI and uses *CLWFDOMCreator* and *CLWFLogDefaultValueProvider* to build the workflow logs DOM-representation. *CLXMLWriter* offers functionality to write this in-memory-representation of the workflow log to a file in the file system.

TeamLog.ui
This is a container for TeamLogs user interface components. *CUStart* initiates the user interface creation during application-start-up. *CUMainFrame* implements TeamLogs main window. *CUSettings* implements a dialog which enables the user to change application settings (e.g. database connection string). *CUDialogAbout* displays general TeamLog information ("about"-box).

Structuring the applications components in this way leads to a clean separation between business logic and user interface implementation.

6.3 Security

TeamLog does not require authentication to start the application, because security issues are not considered as a major goal of TeamLogs implementation. The major goal of TeamLog is to convert Caramba process information to XML. But although TeamLog does not require authentication, accessing the Caramba database via JDBC does. Therefore, TeamLog provides functionality which enables the user to modify this access-information (connection string, database user and the corresponding password), which is stored as plain text in TeamLogs property file.

6.4 Output

A major goal of this section is to describe TeamLogs mapping of Caramba information to the XML workflow log. TeamLog distinguishes between different output types, that determine the source of <date> and <time>, the outputs timestamp-elements (Table 5).

Output type	**Type of logline**	**Attribute in table CARAMBA_AI**
STARTED	logline for a coordination dummy start logline dummy end logline	STARTED STARTED Latest STARTED information of all coordinations within the workcase
STOPPED	logline for a coordination dummy start logline	STOPPED STARTED

	dummy end logline	Latest STOPPED information of all coordinations within the workcase
BEGIN_ACTUAL	logline for a coordination dummy start logline dummy end logline	BEGIN_ACTUAL STARTED Latest BEGIN_ACTUAL information of all coordinations within the workcase
END_ACTUAL	logline for a coordination dummy start logline dummy end logline	END_ACTUAL STARTED Latest END_ACTUAL information of all coordinations within the workcase
BEGIN_SCHEDULED	logline for a coordination dummy start logline dummy end logline	BEGIN_SCHEDULED STARTED Latest BEGIN_SCHEDULED information of all coordinations within the workcase
END_SCHEDULED	logline for a coordination dummy start logline dummy end logline	END_SCHEDULED STARTED Latest END_SCHEDULED information of all coordinations within the workcase

Table 5: Workflow log timestamps depending on the selected output type.

The output type is under the users control (selection in user interface). The output types meanings are:

STARTED: The workflow log describes the real process flow. The timestamp of each coordination shows when it was initiated in Caramba. Caramba generates this timestamp automatically and therefore it is not under the users control.

STOPPED: The timestamp of a coordination indicates, when it's status was set to "done". If a coordination is marked as "done" Caramba automatically sets the STOPPED timestamp.

BEGIN_ACTUAL: Caramba provides functionality to enter the actual begin time which is used as timestamp in the generated workflow log. It indicates the coordinations start time from the users point of view. The main difference between

STARTED and *BEGIN_ACTUAL* is, that the timestamp *BEGIN_ACTUAL* is under the users control.

END_ACTUAL: Caramba enables the user to enter the actual end time of a coordination. Like *BEGIN_ACTUAL*, this timestamp is under the users control. It's the time when the coordinations are completed from the users point of view.

BEGIN_SCHEDULED: This time information is also under the users control. The timestamps indicate the planned start times.

END_SCHEDULED: Corresponds to *BEGIN_SCHEDULED*. The user enters the planned end time in Caramba. This information is then used as timestamp in the generated workflow log.

The comments in the DTD [11] listed below describe how TeamLog maps Caramba database information to workflow log information.

```
<!-- element WorkFlow_log:
     root-element, no Caramba database information required -->
<!ELEMENT WorkFlow_log (source?,process+)>

  <!-- element source: no contents -->
  <!ELEMENT source EMPTY>

    <!-- attribute source: constant value "other" -->
    <!ATTLIST source program (staffware|inconcert|pnet|IBM_MQ|other)
#REQUIRED>

  <!-- element process:
       There is exactly one process-element in TeamLogs output.
       Section "Limitations and assumptions" contains the reason for
       that. -->
  <!ELEMENT process (case*)>

    <!-- attribute id:
         Because the generated workflow log contains exactly
         one process-element, the value of the id-attribute is a
         constant ("caramba_process_1"). -->
    <!ATTLIST process id ID #REQUIRED>

    <!-- attribute description: constant value "none" -->
    <!ATTLIST process description CDATA "none">

    <!-- element case:
         Under the assumption that all workcases are instances of the
         same real life process TeamLog creates a case-element for
         each workcase. -->
    <!ELEMENT case (log_line*)>

       <!-- attribute id:
            "case_" + CARAMBA_AI.ID of the "W"-record e.g. "case_12"
            if the workcase-records ID is 12 -->
      <!ATTLIST case id ID #REQUIRED>

       <!-- attribute description:
            CARAMBA_AI.WC_SUBJECT + "_" + number
            Different workcases can have the same content in
            CARAMBA_AI.WC_SUBJECT. Therefore a serial number is
            appended to get a unique description (is advised in [5]).
```

```
            If CARAMBA_AI.WC_SUBJECT is null, the description-
            attribute gets the constant value "workcase_" + number
           (e.g. "workcase_1", "workcase_2"). -->
      <!ATTLIST case description CDATA "none">


      <!-- element log_line:
           TeamLog generates one log-line for each coordination.
           Additionally to that it adds a dummy-start-logline and a
           dummy-end-logline. -->
      <!ELEMENT log_line (task_name,task_instance?, event?, date?,
                          time?, originator?)>

        <!-- element task_name:
             For dummy start logline: "Case start"
             For dummy end logline: "Case termination"
             The remainder of this section gives additional
              information about task_names content. -->
        <!ELEMENT task_name (#PCDATA)>

        <!-- element task_instance:
             This field is designed to store the number of task-
             executions. The Caramba database does not yet contain
             this kind of information. Therefore it's not part of
             TeamLogs output. -->
        <!ELEMENT task_instance (#PCDATA)>

        <!-- element event: no contents. -->
        <!ELEMENT event EMPTY>

           <!-- attribute kind:
                Event information is information about the change of
                state. The Caramba database does not contain event
                information. Therefore the value of this attribute is
                constant ("normal"). -->
           <!ATTLIST event kind
(normal|schedule|start|withdraw|suspend|resume|abort|complete)
#REQUIRED>


        <!-- element date: Table 4 contains information about this
             elements content. -->
        <!ELEMENT date (#PCDATA)>

        <!-- element time: Table 4 contains information about this
             elements content. -->
        <!ELEMENT time (#PCDATA)>

        <!-- element originator: contains the sender of a
             coordination (value is CARAMBA_AI.SENDER_DENORM). -->
        <!ELEMENT originator (#PCDATA)>
```

The user can influence the <task_name>-elements contents. If he requests anonymous task names, TeamLog automatically replaces the real task names with “T1”, “T2”, “T3”, etc. Anonymous task names might be required for outsourced process analysis where the actual tasks are not relevant. Otherwise CARAMBA_AIs NAME-attribute is used. Unique task names are suggested in [5]. Therefore, if a task name appears more than once within a workcase (<case>-element), a number is appended

automatically. The number is reset to 1 at the beginning of each workcase.

Example:
Assume that the task “create design-document” appears three times within a case. The first task name does not change, the second will be “create design-document_1” and the third will be “create design-document_2”. If another task, e.g. “analyse design-document”, appears twice, the task name of the second appearance is changed to “analyse design-document_3”.

Limitations and assumptions
TeamLogs goal is to generate workflow logs describing Caramba ad-hoc processes. Per definition, ad-hoc processes do not have a-priori-known process definitions.
A workflow log usually contains information about multiple process instances (workcases), that means information about multiple executions of the same real world process. Because for ad-hoc processes it cannot be determined automatically to which real world process they belong, TeamLog makes the following assumption:
All workcases in table CARAMBA_AI within a Caramba workspace belong to the same real world process. Each Caramba workspace has its own database and therefore, it has its own table CARAMBA_AI. This assumption leads to exactly one <process>-element in the generated workflow log because all workcases belong to the same real world process. Caramba does not store event information. For each coordination there is only one “C”-record in CARAMBA_AI. If the coordinations status is changed, this record is updated and therefore information about the history of state-changes is not available. That’s the reason why the <event>-elements kind-attribute has the constant value “normal”.

The workflow-logs timestamp information (<date>- and <time>-elements) depends on the selected output type. Some output types require timestamp information which is under the users control. If the user does not enter the information in Caramba the corresponding output fields (<date>- and <time>-element) will have no value.

During process mining all cases (<case>-elements) are used to reconstruct a process-definition. Therefore the task names in different cases, which belong to the same task in real world, must be equal. Because the content of the element <task_name> is entered by the CARAMBA-user, that’s in his responsibility. If the user does not take care, useful process mining is not possible.

Social network analysis
In the workflow logs listed in the sections above the only possibility for social network analysis is the use of the <originator>-element, which contains the senders name. To enable more sophisticated SNA, TeamLog supports an alternative DTD, which contains two additional elements: <recipient> and <role>. The recipient-element contains the name of the recipient (attribute RECEIPIENT_DENORM in table CARAMBA_AI). That’s the name of the person that has to perform an activiy. <role> contains the role which serves as addressee (e.g. “sales department”, if a coordination is sent to the members of the group “sales department”). Its value comes

from attribute ADDRESSEE_DENORM in table CARAMBA_AI. If a coordination is sent to group, ADDRESSEE_DENORM contains the group name (e.g. “sales department”). But if a coordination is sent to a specific person, ADDRESSEE_DENORM contains the name of the person. That has to be taken into account during social network analysis. It’s up to the user if this additional elements are part of the workflow log (user interface).

Sorting and filtering

TeamLogs output contains data about workcases and their coordinations. It’s up to the user to choose those workcases that shall be part of the output. Furthermore it is possible to restrict the coordinations which shall be taken into account. Therefore, the user has the possibility to enter timestamp-ranges. Further information concerning this fields can be found in Section 6.5.

Additionally to this filters the user can choose the elements and attributes that shall be part of the workflow log (for each optional element or attribute there is a corresponding checkbox in the user interface). The workcase-order in the workflow log is always based on CARAMBA_AI.STARTED (ascending). Within a workcase, the coordinations are sorted by timestamp. Which timestamp information is used depends on the selected output type.

Validation

To avoid/detect application- or data-errors, the generated output has to be validated against a DTD. If this validation fails, TeamLog informs the user. TeamLog supports validation against 2 different DTDs: a standard DTD [11] and an alternative DTD, which contains additional elements (<recipient>,<role>) to allow more flexible social network analysis. If the user selects neither the recipient- nor the role-element, the generated workflow log is validated against the standard DTD, otherwise against the alternative DTD. But there is one exception: if the appropriate DTD file path is not set in TeamLogs property file, validation is skipped.

6.5 Using TeamLog

TeamLog is launched when *TeamLogStart*s main-method is called. During TeamLogs start procedure it reads the connection information used for JDBC access. Then it reads the distinct workcase descriptions from table CARAMBA_AI and copies them into the appropriate user interface components (screen descriptions in the remainder of this section) where the user can select the workcases. If the connection information is not available in the property file or the database connection can’t be established an error message appears.

TeamLogs property-file contains six persistent settings. The *database connection string*, the *database user* and the corresponding *password* are used to access Caramba’s data. The setting *Workflow-Log-Filepath* holds the path of the XML workflow log that TeamLog generates. The remaining settings are *Workflow-Log-DTD* and *Workflow-Log-DTD-alternative* which hold the locations of the document

type definitions. The remainder of this sections shows how TeamLogs user-interface looks like and which functions are available.

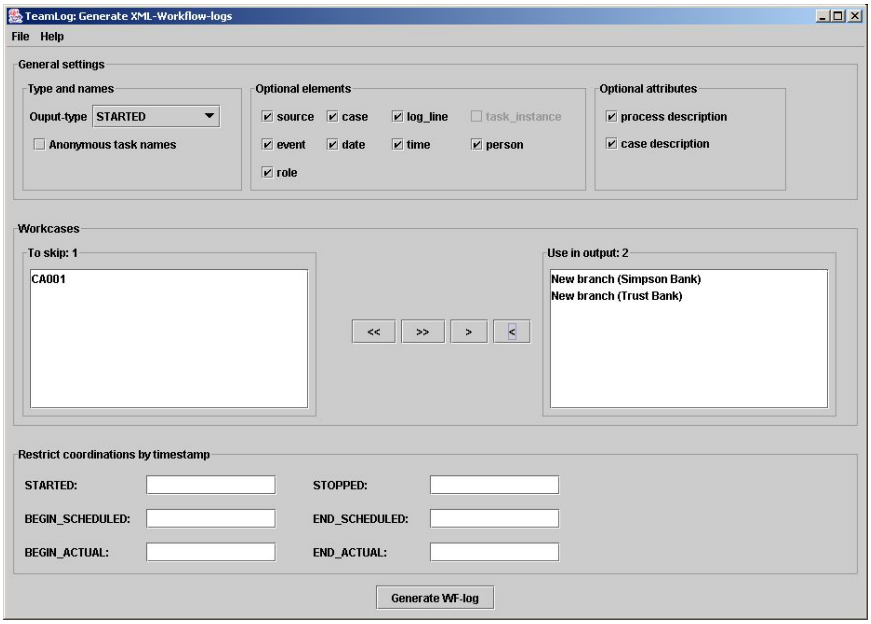


Figure 7: Main screen of TeamLog.

The user can choose the output type, request anonymous task names and select the optional elements/attributes that shall be part of the output. In addition to that he is able to select the workcases that shall be considered and he can restrict the coordinations within the workcase. The coordinations can be restricted by timestamp (input format “dd.mm.yyyy hh:MM:ss”) according to the following rules:

STARTED: the coordinations *STARTED* timestamp must be newer than the input (or equal).

STOPPED: the coordinations *STOPPED* timestamp must be older than the input (or equal).

BEGIN_SCHEDULED: the coordinations *BEGIN_SCHEDULED* timestamp must be newer than the input (or equal).

END_SCHEDULED: the coordinations *END_SCHEDULED* timestamp must be older than the input (or equal).

BEGIN_ACTUAL: the coordinations *BEGIN_ACTUAL* timestamp must be newer than the input (or equal).

END_ACTUAL: the coordinations *END_ACTUAL* timestamp must be older than the input (or equal).

These inputs can be combined freely. Figure 8 shows the dialog that offers possibilities to change the contents of TeamLogs property file.

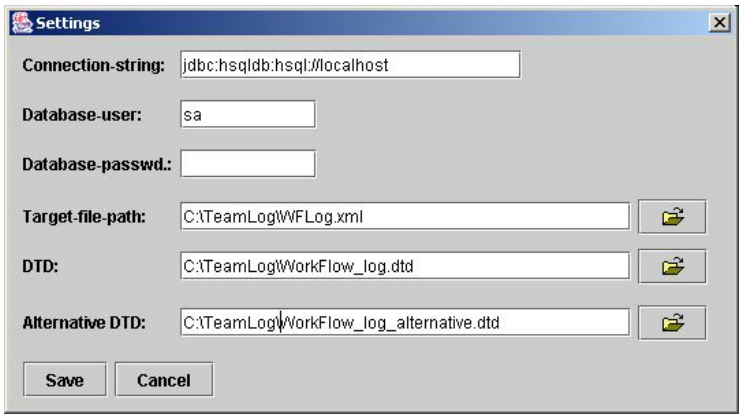


Figure 8: TeamLog – changing the contents of TeamLogs property file.

6.6 Workflow log generation

There is one method which controls the workflow log generation, *CLActions.generate_WFLog*. Figure 9 describes the most important steps of this method. To reduce complexity we omitted some details. After the user-interface component *CUMainFrame* has created an instance of *CLActions*, it sets up a *ContentDefinition*-object, which holds information about the request (e.g. which optional attributes shall be part of the output or which workcases have to be read from table CARAMBA_AI). Then this *ContentDefinition* is passed to *CLActions.generate_WFLog*. Furthermore, *generate_WFLog* uses the business logic classes *CLWFLogGenerator* and *CLWFDOMCreator* to get a DOM representation of the requested XML workflow log. Finally, the log is written to a file (*CLXMLWriter*) and it's validated against the appropriate DTD (*CLValidator*). If validation fails, an error message appears.

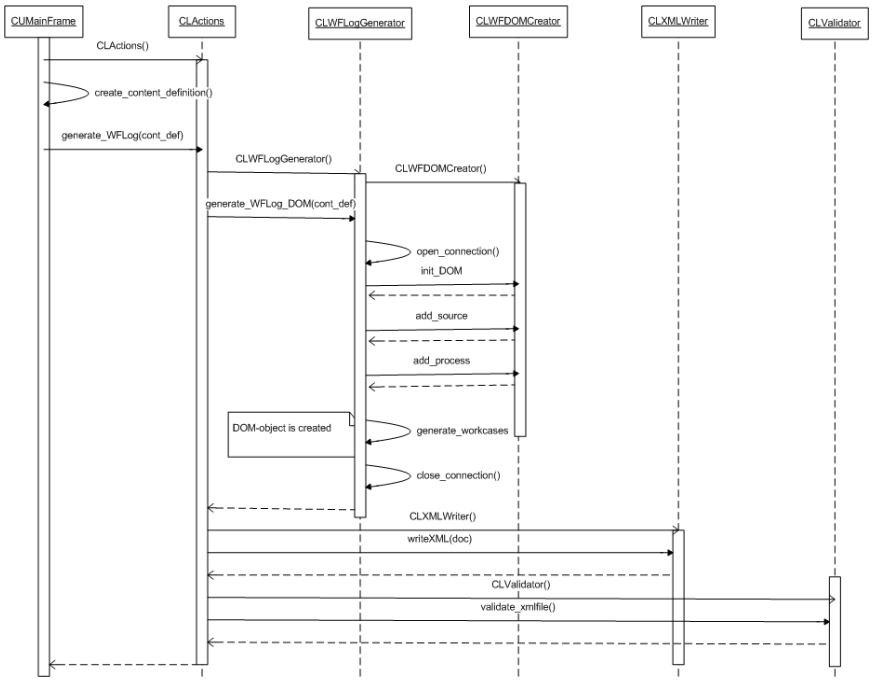


Figure 9: Sequence diagram - CLActions.generate_WFLog.

The remainder of this paper gives an application of what has been discussed – mining of ad-hoc processes.

7 Mining of ad-hoc processes

To illustrate the concept of process mining using real Caramba logs, we consider the process ("plan IT-installation for offices (banks)") described in the first part of the paper. In both Sections 4 and Section 5.2 we described a real process instance of this process. If we play these two scenarios using Caramba, then TeamLog is able to generate the following workflow log based on the information stored in Caramba.

```
<WorkFlow_log>
    <source program="other"/>
    <process description="none" id="caramba_process_1">
        <case description="New office (Simpson Bank)_1" id="case_12">
            <log_line>
                <task_name>Case start</task_name>
                <event kind="normal"/>
                <date>08-02-2004</date>
```

```
                <time>16:57:53</time>
                <originator/>
                <recipient/>
                <role/>
            </log_line>
            <log_line>
                <task_name>Create installation plan</task_name>
                <event kind="normal"/>
                <date>09-02-2004</date>
                <time>12:14:01</time>
                <originator>Lachs Monika</originator>
                <recipient>Huchen Marta </recipient>
                <role>Huchen Marta</role>
            </log_line>
            <log_line>
                <task_name>Adapt the format of the installation
plan</task_name>
                <event kind="normal"/>
                <date>09-02-2004</date>
                <time>16:02:29</time>
                <originator>Huchen Marta</originator>
                <recipient>Lachs Monika </recipient>
                <role>Lachs Monika</role>
            </log_line>
            <log_line>
                <task_name>Verify installation plan</task_name>
                <event kind="normal"/>
                <date>10-02-2004</date>
                <time>11:02:11</time>
                <originator>Lachs Monika</originator>
                <recipient>Lachs Monika </recipient>
                <role>Sales department</role>
            </log_line>
            <log_line>
                <task_name>Verify installation plan_1</task_name>
                <event kind="normal"/>
                <date>10-02-2004</date>
                <time>12:24:18</time>
                <originator>Lachs Monika</originator>
                <recipient>Fogosch Peter </recipient>
                <role>Sales department</role>
            </log_line>
            <log_line>
                <task_name>Verify installation plan_2</task_name>
                <event kind="normal"/>
                <date>10-02-2004</date>
                <time>14:05:22</time>
                <originator>Lachs Monika</originator>
                <recipient>Zander Fritz </recipient>
                <role>Sales department</role>
            </log_line>
            <log_line>
                <task_name>Consider comments from Peter</task_name>
                <event kind="normal"/>
                <date>11-02-2004</date>
                <time>09:07:17</time>
                <originator>Fogosch Peter</originator>
                <recipient>Lachs Monika </recipient>
                <role>Lachs Monika</role>
            </log_line>
```

```
            <log_line>
                <task_name>Consider comments from Fritz</task_name>
                <event kind="normal"/>
                <date>11-02-2004</date>
                <time>10:08:30</time>
                <originator>Zander Fritz</originator>
                <recipient>Lachs Monika </recipient>
                <role>Lachs Monika</role>
            </log_line>
            <log_line>
                <task_name>Case termination</task_name>
                <event kind="normal"/>
                <date>11-02-2004</date>
                <time>10:08:30</time>
                <originator/>
                <recipient/>
                <role/>
            </log_line>
        </case>
        <case description="New office (Trust Bank)_2" id="case_25">
            <log_line>
                <task_name>Case start</task_name>
                <event kind="normal"/>
                <date>13-02-2004</date>
                <time>11:14:24</time>
                <originator/>
                <recipient/>
                <role/>
            </log_line>
            <log_line>
                <task_name>Create installation plan</task_name>
                <event kind="normal"/>
                <date>13-02-2004</date>
                <time>11:14:24</time>
                <originator>Fogosch Peter</originator>
                <recipient>Huchen Marta </recipient>
                <role>Huchen Marta</role>
            </log_line>
            <log_line>
                <task_name>Adapt the format of the installation
plan</task_name>
                <event kind="normal"/>
                <date>14-02-2004</date>
                <time>08:13:01</time>
                <originator>Huchen Marta</originator>
                <recipient>Fogosch Peter </recipient>
                <role>Fogosch Peter</role>
            </log_line>
            <log_line>
                <task_name>Add missing cost analysis</task_name>
                <event kind="normal"/>
                <date>14-02-2004</date>
                <time>14:50:21</time>
                <originator>Fogosch Peter</originator>
                <recipient>Huchen Marta </recipient>
                <role>Huchen Marta</role>
            </log_line>
            <log_line>
                <task_name>Check cost analysis</task_name>
                <event kind="normal"/>
```

```
                <date>16-02-2004</date>
                <time>12:05:30</time>
                <originator>Huchen Marta</originator>
                <recipient>Fogosch Peter </recipient>
                <role>Fogosch Peter</role>
            </log_line>
            <log_line>
                <task_name>Case termination</task_name>
                <event kind="normal"/>
                <date>16-02-2004</date>
                <time>12:05:30</time>
                <originator/>
                <recipient/>
                <role/>
            </log_line>
        </case>
    </process>
</WorkFlow_log>
```

We can now use process mining tools such as EMiT [2,11], Thumb [29], and MinSoN [4]. For example, when EMiT processes this workflow log consisting of only two instances, it produces the Petri net shown in Figure 10.

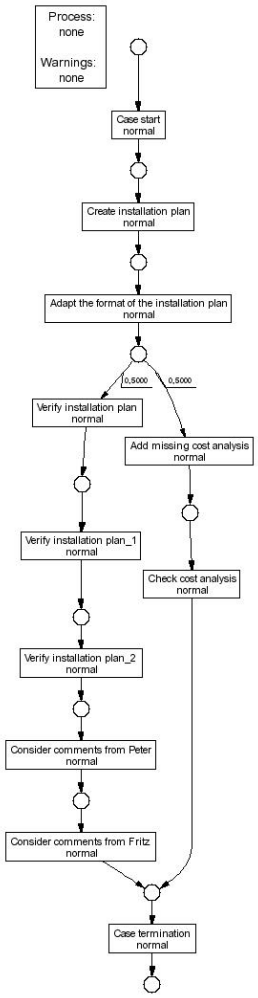


Figure 10: The mining result expressed in terms of a Petri net (diagram was automatically generated using EMiT).

During process mining both process instances (cases) are used. Note that the process model constructed by EMiT indeed captures both scenarios. Figure 10 shows that in some cases the installation plan is verified by multiple persons, and in other cases the plan is not verified. Suppose that the management has never advised the employees to do multiple-person-verification and that many installation-plan-orders are either waiting for processing or they are already delayed. If this is the case, then management can react to this situation and advise the employees to skip multiple-person-verification in order to create all installation-plans in-time. Clearly, these observations may seem trivial based on a simple process of only a few steps generated

on the basis of two cases. However, the same procedure can be applied to much more complicated processes that need to deal with hundreds or even thousands of cases.

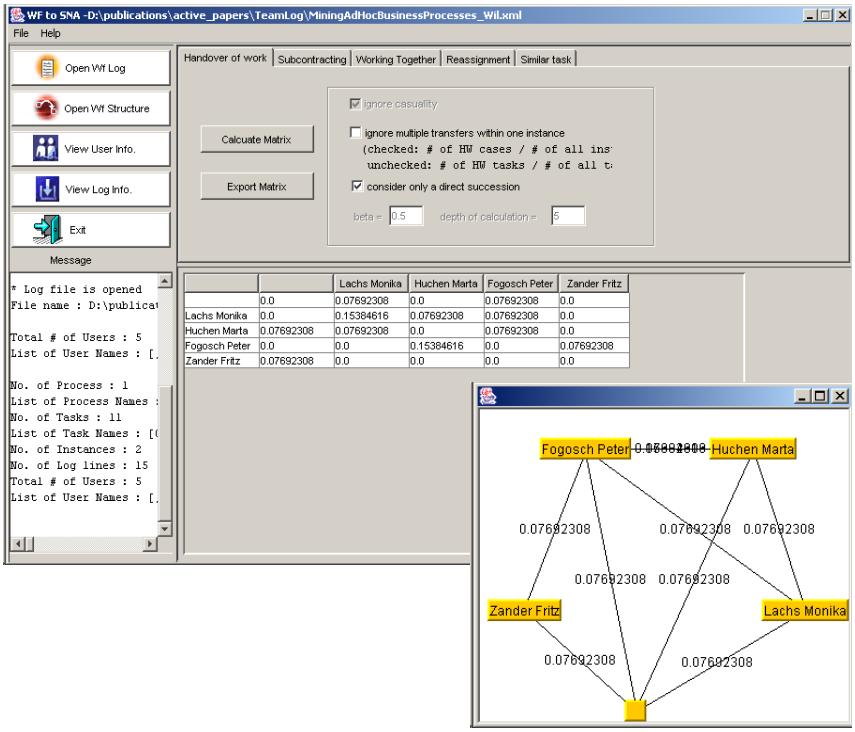


Figure 11: Social network derived by MinSoN based on a Caramba log.

The log can also be analyzed using MinSoN as is illustrated in Figure 11. The screenshot shows the social network constructed on the basis of the two cases. The social network shown in Figure 11 is based on the "hand-over of work" metric. This metric assumes that if there is a transfer of work from one person to another person, then there is a relation. The more work that is transferred, between two person the stronger the relation based on "hand-over of work" is. The resulting social network can be analyzed by all kinds of tools, e.g., MinSoN can export to Agna and NetMiner. These tools analyze both the organization as a whole and the role of each individual in the organization. Because Figure 11 is only based on two cases, it is not possible to obtain meaningful results. However, it illustrates the concept and the fact that through TeamLog and MinSoN it is possible to analyze the organizational context of ad-hoc business processes supported by Caramba.

We have applied the approach presented in this paper to some real-life Caramba logs but did not yet conduct a real case study in an organization using Caramba, i.e., the paper provides a proof of concept but no empirical validation. In Section 3 we mentioned existing BPA/BAM systems such as ARIS PPM. These systems typically

restrict their analysis to well-defined processes and performance indicators such as flow time. The two figures in this section, show that the results are quite different from existing systems, i.e., through TeamLog and EMiT/MinSoN we actually capture causal relations and the organizational context. Moreover, such information is particularly interesting for ad-hoc business processes.

8 Conclusion and future work

This paper investigated the application of process mining techniques to ad-hoc processes. The conclusion is that process mining is very promising in those situations where the steps in the process are logged in a systematic manner. Using process-aware collaboration systems such as Caramba, events are logged while allowing for the flexibility required for ad-hoc business processes. As a proof-of-concept, we developed TeamLog, a system that is able to extract the information required for process mining from the Caramba database. The resulting information is stored in an XML format that can be read by process mining tools such as EmiT and MinSoN. To illustrate the application of Caramba, TeamLog, EmiT, and MinSoN we used as small ad-hoc business process ("plan IT-installation for offices (banks)"). Based on two scenarios we automatically constructed a process model (in terms of a Petri net) and a social network.

Future work, will aim at the application of the entire toolset described in this paper in real-life situations. In addition, we are extending and improving the mining tools. In fact, the systems EmiT, Thumb, and MinSoN will be merged into a new open-source tool where it is easy to "plug-in" new analysis methods (cf. www.processmining.org).

Acknowledgements

This research was supported in part by a research award for *Development Methods for Dynamic Web services Workflows* of the Chamber of Commerce Vienna (Wirtschaftskammer Wien). The authors would like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song for their on-going work on process mining techniques and tools at Eindhoven University of Technology.

9 References

[1] W.M.P. van der Aalst and P.J.S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In S. Ellis, T. Rodden, and I. Zigurs, editors, International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001), pages 42–51. ACM Press, New York, 2001.

[2] W.M.P. van der Aalst and B.F. van Dongen. Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, International Conference on Engineering and Deployment of Cooperative

Information Systems (EDCIS 2002), volume 2480 of Lecture Notes in Computer Science, pages 45–63. Springer-Verlag, Berlin, 2002.
[3] W.M.P. van der Aalst and K.M. van Hee. Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, 2002.
[4] W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering interaction patterns in business processes. In J. Desel, B. Pernici, and M. Weske, editors, International Conference on Business Process Management (BPM 2004), volume 3080 of Lecture Notes in Computer Science, pages 244–260. Springer-Verlag, Berlin, 2004.
[5] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering, 47(2):237– 267, 2003.
[6] W.M.P. van der Aalst and A.J.M.M. Weijters, editors. Process Mining, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
[7] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. QUT Technical report, FIT-TR-2003-03, Queensland University of Technology, Brisbane, 2003. (Accepted for publication in IEEE Transactions on Knowledge and Data Engineering.).
[8] R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In Sixth International Conference on Extending Database Technology, pages 469–483, 1998.
[9] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. Data and Knowledge Engineering, 24(3):211–238, 1998.
[10] J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology, 7(3):215–249, 1998.
[11] B.F. van Dongen and W.M.P. van der Aalst. EMiT: A Process Mining Tool. In J. Cortadella and W. Reisig, editors, Application and Theory of Petri Nets 2004, volume 3099 of Lecture Notes in Computer Science, pages 454–463. Springer-Verlag, Berlin, 2004.
[12] S. Dustdar. Caramba - A Process-Aware Collaboration System Supporting Ad Hoc and Collaborative Processes in Virtual Teams. Distributed and Parallel Databases, 15(1):45–66, 2004.
[13] C. Ellis and K. Keddara. ML-DEWS: Modeling Language to Support Dynamic Evolution within Workflow Systems. Computer Supported Co-operative Work, 9(3/4):293–333, 2000.
[14] Gartner. Gartner's Application Development and Maintenance Research Note M-16-8153, The BPA Market Catches another Major Updraft. http://www.gartner.com, 2002.
[15] D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamo-hanarao, and R. Snodgrass, editors, Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01), pages 159–168. Morgan Kaufmann, 2001.
[16] J. Herbst. A Machine Learning Approach to Workflow Management. In Proceedings 11th European Conference on Machine Learning, volume 1810 of

Lecture Notes in Computer Science, pages 183–194. Springer-Verlag, Berlin, 2000.
[17] IDS Scheer. ARIS Process Performance Manager (ARIS PPM). http://www.ids-scheer.com, 2002.
[18] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, London, UK, 1996.
[19] G. Keller and T. Teufel. SAP R/3 Process Oriented Implementation. Addison-Wesley, Reading MA, 1998.
[20] P. Lawrence, editor. Workflow Handbook 1997, Workflow Management Coalition. John Wiley and Sons, New York, 1997.
[21] F. Leymann and D. Roller. Production Workflow: Concepts and Techniques. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
[22] D.C. Marinescu. Internet-Based Workflow Management: Towards a Semantic Web, volume 40 of Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, New York, 2002.
[23] A.K.A. de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Workflow Mining: Current Status and Future Directions. In R. Meersman, Z. Tari, and D.C. Schmidt, editors, On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, volume 2888 of Lecture Notes in Computer Science, pages 389–406. Springer-Verlag, Berlin, 2003.
[24] M. zur Muehlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33), pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
[25] M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. Journal of Intelligent Information Systems, 10(2):93–129, 1998.
[26] W. Reisig and G. Rozenberg, editors. Lectures on Petri Nets I: Basic Models, volume 1491 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1998.
[27] M. Sayal, F. Casati, and M.C. Shan U. Dayal. Business Process Cockpit. In Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02), pages 880–883. Morgan Kaufmann, 2002.
[28] J. Scott. Social Network Analysis. Sage, Newbury Park CA, 1992.
[29] A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. Integrated Computer-Aided Engineering, 10(2):151–162, 2003.