Process Mining Can Be Applied to Software Too!

Vladimir A. Rubin[1,2]
vrubin@hse.ru

Alexey A. Mitsyuk[2]
amitsyuk@hse.ru

Irina A. Lomazova[2]
ilomazova@hse.ru

Wil M.P. van der Aalst[2,3]
w.m.p.v.d.aalst@tue.nl

[1]Dr. Rubin IT Consulting, 60599, Frankfurt am Main, Germany
(In collaboration with msg systems AG, Germany),

[2]International Laboratory of Process-Aware Information Systems,
National Research University Higher School of Economics (HSE),
33 Kirpichnaya Str., Moscow, Russia.

[3]Architecture of Information Systems, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.

ABSTRACT

Modern information systems produce tremendous amounts of event data. The area of process mining deals with extracting knowledge from this data. Real-life processes can be effectively discovered, analyzed and optimized with the help of mature process mining techniques. There is a variety of process mining case studies and experience reports from such business areas as healthcare, public, transportation and education. Although nowadays, these techniques are mostly used for discovering business processes.

However, process mining can be applied to software too. In the area of software design and development, process models and user interface workflows underlie the functional specification of almost every substantial software system. When the system is utilized, user interaction with the system can be recorded in event logs. After applying process mining methods to logs, we can derive process and user interface flow models. These models provide insights regarding the real usage of the software and can enable usability improvements and software redesign.

In this industrial paper we present several process mining examples of different productive software systems used in the touristic domain. With the help of these examples we demonstrate that process mining enables new forms of software analysis. The user interaction with almost every software system can be mined in order to improve the software and to monitor and measure its real usage.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ESEM '14, September 18–19, 2014, Torino, Italy.
Copyright 2014 ACM 978-1-4503-2774-9/14/09 ...$15.00.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: General; D.2.8 [**Software Engineering**]: Metrics—*process metrics*

General Terms

Experimentation, Design

Keywords

Process Mining, Software Process Mining, User Interface Design, Client Technology

1. INTRODUCTION

Information systems from different business areas, such as healthcare, public, transport and education, produce tremendous amounts of *event data*. Process mining provides mature theoretical and practical foundations for discovering processes from different kinds of event data [27, 28]. The core idea of process mining is to start from real event data and to use these to *derive process models*. There are numerous examples of successful applications of process mining in *Business Process Management* (BPM), but the adoption in other domains is lagging behind.

Big *software projects* and programs influence significantly the way companies work. IT architectural changes and improvements can be introduced only when the business processes are properly analyzed and structured. From a *software engineer's* point of view, business process design is a fundamental part of almost every enterprise architectural concept and of the functional specification of almost every information system [8, 4]. Even for building the *graphical user interface* (GUI), architects and requirement engineers often start with designing the user workflows (screenflows) – *GUI storyboards* [24, 11, 26]. These storyboards specify how the system should be used from the end-user perspective.

In this paper we focus on the *user workflows*. The idea of *user behavior analysis* at runtime is not new, e.g., it is especially widely adopted in the web domain. Well-known instruments as *Google Analytics* and *Piwik* are effectively used in many companies working in the e-commerce area. However, "smart" analysis of the user behavior and of the user workflows continues to be a challenging problem. Since usability and user acceptance are complicated issues in almost every innovative software project, foundational research and robust tools dealing with user behavior analysis are definitely needed.

In this paper we present the area of *software process mining*, which deals with applying process mining to the software analysis and especially to the *user behavior analysis.* We outline two *experience reports* based on the software systems used in the touristic domain: one big European customer reservation system and one ticket reservation system widely used in Russia[1]. There are a lot of similarities between these projects: stakeholders in both projects wanted to discover and monitor the real usage of the systems for improving the design and development. Moreover, similar use cases are available for a wide range of running software systems from different business areas. Therefore, our hypothesis is that process mining can be successfully applied to the domain of *software user interface design*, usability and runtime analysis of software systems.

This is an *industrial paper* and the usage of process mining was initiated due to the needs of the two projects. Thus, we start directly with experience reports in the next section.

2. EXPERIENCE REPORT

In this Section, we present two process mining case studies, which deal with the analysis of complex touristic *ticket reservation systems.* The architecture, the size and the market segments of these systems are different, but similar methods were applied and similar analysis results were obtained.

2.1 Mining Computer Reservation System

In the first case study, we deal with a big *European touristic system.* This system belongs to a wide class of *Computer Reservation Systems.* The system is rolled out in Europe in more than 10000 travel agencies. The *backend* of the system contains the business logic and provides booking services to different types of client applications. Most of the *travel agencies* use text-based clients for expert users to access the backend.

The communication with the backend is carried out through a special *touristic protocol* called *TOMA*, which is widely accepted in the industry. TOMA is standardized and perfectly documented [1]. The initial versions of the protocol were built in the 80's and used ever since. The main advantage for our research is that every message of the protocol exactly specifies the user behavior and the user input data. The start and the end of the interaction with the system are also defined. Thus, we have a perfect precondition for identifying the business cases (process instances) and the user activities.

One important component of the backend is a special "inbound adapter", which intercepts all the calls to the backend and logs the data. These logs are normally used for software debugging and testing – producing such logs is common practice in the industry. So, our initial idea was to extend these logs with the details of the TOMA messages. It is useful both for developing and testing the system components, which deal with TOMA, and for enabling process mining and behavior analysis.

2.1.1 *Event log*

A fragment of an event log is shown in Table 1. Every log entry contains an id of the case, an activity name, a timestamp, a user id, a booking code and a notification. An activity name encodes the type of message, the user activity and the screen, e.g. "T1-HF-H:TES" means that T1 message (initial process start message) was sent to the system, the user activity was "HF-H" – search for available hotels and the screen "TES" (entry screen) was shown. The booking code identifies the location, e.g. "BER" means Berlin, Germany. And the notification depicts the system message, which was sent to the user, e.g. "998" is a string message of a variable length. Request messages start with "T" and response messages start with "M". Every case starts with "T1" message and finishes with the "M52" message. Thus, in the example in Table 1 we show two cases: "Case 16" is search for available hotels in Berlin and "Case 17" – reserve double room in a concrete hotel in Berlin.

With the help of the inbound adapter, we collected the logs for each day, when the system was tested (acceptance tests) or productively used. We used the tool *Disco (Fluxicon)* for process mining and model analysis. Disco was chosen because of its usability, fidelity, reliability and performance when dealing with big logs. The core of Disco is based on the fuzzy mining algorithm [10]. The strength of the fuzzy miner is the seamless process simplification and highlighting of frequent activities. A Disco screenshot with the model for one working day is shown in Fig. 1. The size of the log in CVS format is 300 KB. The log contains 1122 cases, 4526 events and 79 activities. Each case represents a complete process containing one or several system requests like search, book, payment, etc. The nodes of the graph represent the activities and the arcs – the order of their execution. The numbers at the nodes show activity frequencies and the numbers at the arcs – frequencies of corresponding connections (execution paths).

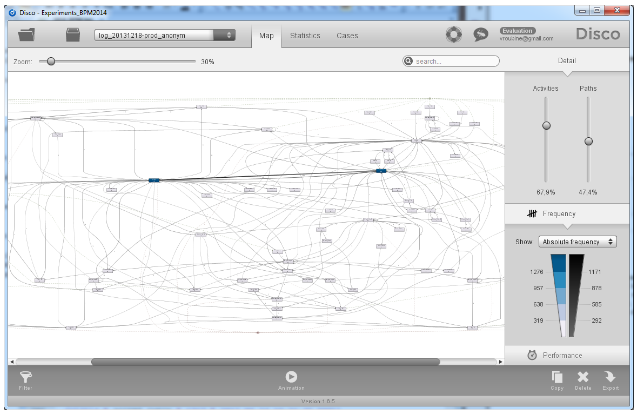


Figure 1: Disco Screenshot with a Fuzzy model

Even for one day (normally covering a subset of possible behavior) a model looks rather complicated. Thus, we used the benefits of Fuzzy mining for adjusting the level of details

[1]The European project was done by Dr. Rubin IT Consulting in collaboration with msg systems AG (Frankfurt am Main, Germany). The Russian project was done by the members of PAIS Lab (Moscow, Russia).

Case ID	**Activity**	**Timestamp**	**User**	**Booking Code**	**Notifi-cation**
Case 16	T1-HF-H:TES	2013-12-18 08:36:00:570	C05	BER	
Case 16	M55Type010Rsp-034	2013-12-18 08:36:04:717	C05		998
Case 16	T3-HF-H:HH004	2013-12-18 08:36:09:337	C05	BER	
Case 16	M52Rsp	2013-12-18 08:36:09:337	C05		998
Case 17	T1-BA-H:TES	2013-12-18 08:36:12:155	C05	BER45010 DH	
Case 17	M52Rsp	2013-12-18 08:36:18:397	C05		712

Table 1: Touristic protocol log

and also applied filtering in order to better perceive the process model. We consider these mechanisms to be absolutely essential for practical process mining. Further, we focus on one day of acceptance tests (pre-production phase).

2.1.2 *Frequency and negative behavior*

With the help of frequency-based adjustment of detail level, we easily identified the most frequent activities, see Fig. 2. In the model, we see that activities such as "T1-AI" (get information about travel agency), "T1-D" (get reservation) and "T1-BA-H" (reserve hotel) were executed often.

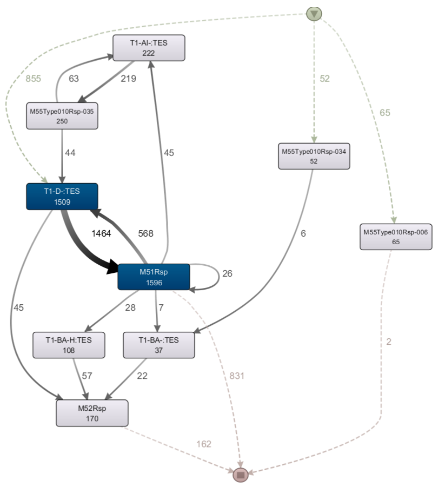


Figure 2: Frequency-based process model

Sometimes, it is useful to highlight the *negative behavior*; in our example the paths which cause message "M51Rsp"(message 51 is a failure message) as a response contain failures. With the help of filtering, we can focus on the cases which include the message of this type. In Fig. 2, the transition between "T1-D-:TES" and "M51Rsp" is frequent, it shows the situations when the user wanted to get a reservation and received failures. From this part of the model, we understand that the implementation of "get reservation" function is *error-prone*. "T1-D" (get reservation) caused "'M51Rsp' 1464 times. In 77% of all cases, the user had problems with representing existing reservations.

Next, we switch to the cases and variants view, see Fig. 3. Variants represent the types of cases with the same behavior. In 65.86% of all negative cases, the user just wanted to get a reservation, received an error and stopped his work.

The process models and particular cases were shown to the team of *acceptance testers* and also to the team of *developers*. Both teams could easily understand the models

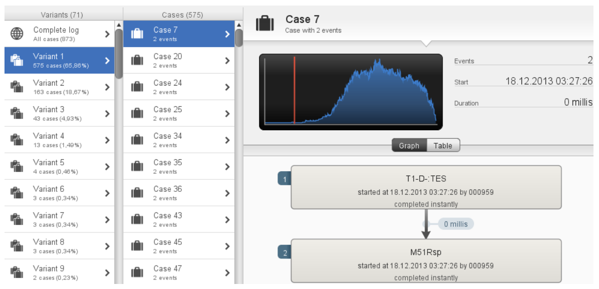


Figure 3: Cases having message "M51Rsp" as a response contain failures

and emphasized that the models really reflected the tested behavior. During the whole day, the testers had significant problems with "get reservation" function, many of them were disappointed and stopped their tests of this function. Developers confirmed that that particular release introduced new changes to "get reservation" and some basic functionality was broken. Since we had particular timestamps for the negative cases, we *aligned* these timestamps with the software logs (debugging and exceptions logs) and found out the *exceptions* thrown by the system. Then, we created *bug issues* for the developer team. The process models were included to the description of the bugs, which helped the developers to reproduce and fix them. Moreover, such type of concrete formal communication, when the behavior is visualized and the bugs are formulated, helped the teams of testers and the developers to understand each other better.

2.1.3 *Positive behavior*

For the same day of acceptance tests, we also excluded all the cases with failures (message "M51") and focused only on the positive behavior. Such functions as "T1-G-F" (flight search), "T1-BI" (booking information), "T1-BA-H" (reserve hotel) and "T1-AI" (get agency information) worked properly during the whole day.

This information was extremely useful for *management*, since it depicted the *stable functionality* of the system. We showed two things: the positive behavior and also the functionality which was *tested* by the test team. The managers understood the widely used features and the testers received an overview of their test cases. The missing *test cases* were identified and for some types of behavior it was decided (based on frequency) to make more thorough tests.

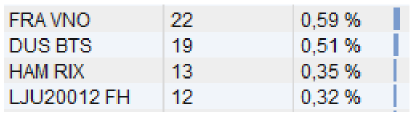
2.1.4 *Typical Worfklow*

For the designers and developers it is relevant to know how the user works. It is helpful for reproducing the bugs, writing tests and for learning the real system. Thus, we filtered out all the response messages and just looked at the requests typically sent by the user. In Fig. 4a we show a performance view and in Fig. 4b – a frequency view on the process. The arcs in Fig. 4a are labeled with median duration between activities. Users use to start with "T1-BA" (make pre-reservation) or "T1-D" (get reservation), then they proceed with such activities as "T1-H-H" (search hotel) or "T1-G-F" (search flight). It is interesting to see, that sometimes users do not close the clients properly, for example transition "T1-BA-H" – "T1-AI" took 3.4 hours because the user left the application open for a long time.

These models were especially useful for the *developers* and for the *test automation* team. Developers could learn how the users actually work. For example, make a reservation, then open it, search for further service (hotel or flight) in the context of reservation, book it and repeat the whole process again. Thus, saving the reservation object in the execution context and caching was a logical *technical decision* made by the developers based on the process mining results. The test automation experts also understood the typical flows and could improve their *test scripts* in order to check the real behavior.

2.1.5 *Statistics*

Also, the statistical information derived from the logs was extremely useful. Since our log (see Table 1) contained information about booking codes, users and notifications, we found out the most frequent travel directions, most active users and most typical failure messages. For example, in Fig. 5, we see that "FRA VNO" (Flight from Frankfurt to Vilnius) was one of the most frequent routes.



FRA VNO	22	0,59 %
DUS BTS	19	0,51 %
HAM RIX	13	0,35 %
LJU20012 FH	12	0,32 %

Figure 5: Statistics about booking codes

It is important to note that not only the booking code and the notification code can be included into the log, there are plenty of different data attributes which could be relevant for the analysis. For example, we could concentrate on the *organization aspect* and deal with roles of the users and information about travel agencies, or on the *informational aspect* and booking and travel dates or customer data. The process model and the statistics perfectly complement each other: the model with frequencies and statistics enables focusing on relevant things, statistics based on the model better reveals the subject.

Thus, in this case study, with the help of process mining we did the following:

- *Visualized* user behavior and discussed it with the users and developers. Process models helped *designers, developers and testers* to learn the typical way of work. Moreover, the models could be regularly shown to the *management*, which helped them to estimate the real stability and readiness of the system.

- *Monitored* the acceptance tests. After focusing on negative behavior, we regularly identified the problems of the users even before they created *bug issues* for them. Moreover, the user problems were aligned with *exceptions* in the logs and the developers had a complete view on the faulty behavior – from user to the business logic.

- Identified the most *typical* and frequently used functions of the system. The developers and test teams focused on implementing, testing and fixing the most common and critical functionality.

At the end of this case study, it is worth mentioning that not only described touristic system, but almost *200 tour operators* in Europe use the same protocol. Similar mining techniques can be easily transferred to these systems. Such *standard touristic processes* as analysis of the preceding touristic season, future demand calculation and yield management can benefit from process mining.

2.2 Mining Traveling Portal

In the second case study, we examine a popular *Russian traveling portal*. In comparison to the previous case study, this system is completely web-based and focused on *e-trading*. Through the portal, users search and book their tickets. Traveling directions are searched using different criteria like origin, destination, date, class of service, etc. During the *sales process*, the tickets are booked and than, in the *after-sales process*, payment is done. Like in the previous system, the backend contains the business logic. But in this case, the customers use the *web interface* for accessing the system. The backend server (web and application servers) produces the event logs, which serve as an input for process mining.

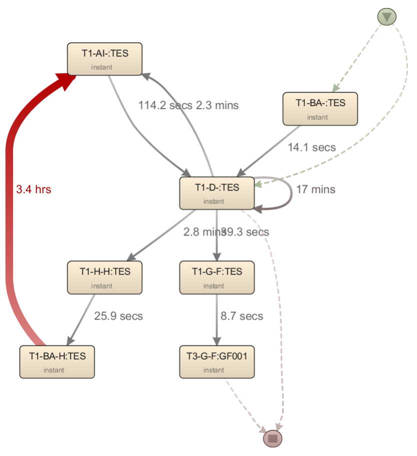
The touristic company constantly measures the *effectiveness* of the web portal usage with the help of "amount of purchases per visitor" metric. At some point of time it was noticed that the value began to decrease. It was obvious that people were leaving the website and the company was *loosing the clients*. Thus, it was decided to *analyze the user behavior* in order to improve the functionality of the portal. It motivated the company to use the process mining techniques for this analysis.

2.2.1 *Event log*

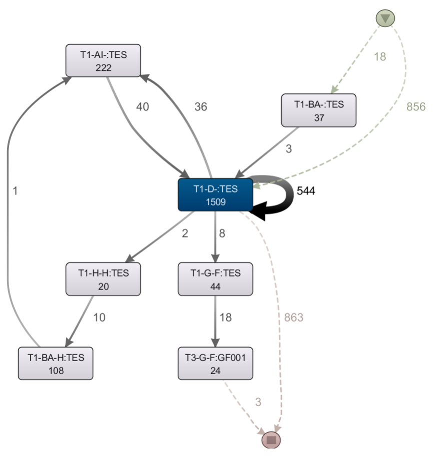
A fragment of the event log is shown in Table 2. A log entry contains a session id, an activity, a user id and a timestamp. The session id is used as a case id for process mining. An activity consists of an object and an action, e.g. "WINDOW/LOAD" means that the object "WINDOW" was loaded. Generally the log contains also additional attributes like url, browser version, order status, etc. All these attributes are relevant for statistical analysis and filtering, but for process mining we initially used only the attributes shown in the table.

This time we used the process mining toolkit ProM [29]. ProM contains more than 600 plugins and facilitates selection of appropriate mining and analysis techniques. For this case study, we used both fuzzy and heuristic miners. *Heuristic miner* [30] is a mining approach, which produces Heuristic nets and deals finely with noise and less structured models. A ProM screenshot with the results of process mining for a one month period is shown in Fig. 6. The log contained 11087 cases, 81872 events and 50 activities.

2.2.2 *Frequency*



(a) Performance view



(b) Frequency view

Figure 4: Models of typical user behavior

Session ID	Activity	User	Timestamp
4654514	WINDOW-LOAD	6767	2013.09.24 19:36:48.000
4654514	PROFILE-CHECK	6767	2013.09.24 19:36:49.000
4654514	ACCEPT-CHECK	6767	2013.09.24 19:36:49.000
4654514	CONFIRM_SUBMIT-CLICK	6767	2013.09.24 19:36:49.000
4654514	WINDOW-UNLOAD	6767	2013.09.24 19:36:49.000

Table 2: Web portal log

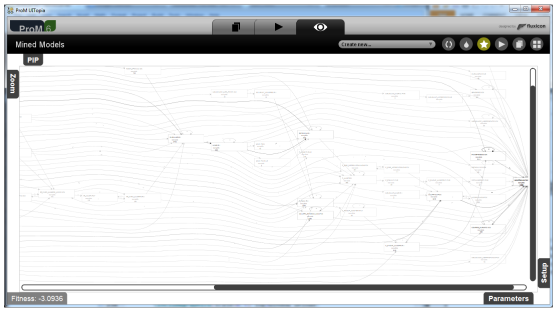


Figure 6: Screenshot showing a heuristic model discovered by ProM

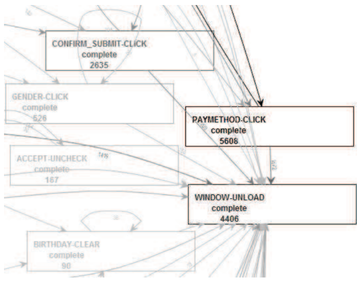


Figure 7: Model fragment of a filtered log

Our goal was to identify the *frequent scenarios* characterizing the cases where users left the website without finishing the booking process. So, we filtered out the cases with order status "finalized" (booking was completed) and also the cases containing only opening and closing events from the log. Then we focused on the events preceding the "WINDOW/UNLOAD" event. A fragment of this model is shown in Fig. 7.

With the help of statistical analysis, we found out that the most frequent events preceding leaving the portal were:

1. Payment method verification "PAYMETHOD-CLICK": 43.14%,
2. Booking confirmation "CONFIRM_SUBMIT-CLICK": 38.29%,
3. Acceptance of fare conditions "ACCEPT-CHECK": 33.88%,
4. Removal of an insurance policy "INSURED_PERSON-UNCHECK": 20.47%

We carefully analyzed all the paths listed above. For example for the case (1), we found out that the user normally pressed the selection of payment method multiple times, before leaving the page. It was a clear sign of a *misleading software behavior*.

2.2.3 *Typical Workflow*

Like in the previous case study, we were also interested in a users typical way of work. The log was filtered, so that only events existing in more than 44% of cases remained. The Heuristic Net model is shown in Fig. 8. The users normally started with entering the personal data and contact information, proceeded with selecting the payment method and submitting the data to the server.

It was interesting to notice that more than a half of all the cases did not follow the normal scheme. After analysis of different cases, we found that many exceptional cases contained multiple clicks of the same buttons and duplicate user actions. Developers were surprised about various system usage scenarios, which were neither designed nor expected.

Our investigations on typical workflows and on the frequent portal leaving were presented to the *management* and to the *development*. Both sides were impressed by the deepness of the analysis of their processes, which was achieved with the help of process mining. On the basis of our analysis, we also made research on improving the situation and formulated the *agenda*. The agenda contained points such as (1) improving the purchase scheme and reassessing the payment methods, (2) changing the design of the offers page, (3) revising the fare policy, etc.

So, in our case studies we focused on the *touristic systems*, their functionality is transparent and easily understandable to the readers. Both systems produce *similar event logs* as input for process mining; in both systems managers and developers were interested in *visualization and analysis* of *frequent positive/negative* and typical behavior. However, similar logs can be produced by different software systems from various application domains. Independently of the technology, whether it is a rich client application (first case study) or a web portal (second case study), the GUI design and usage *principles are similar* and, thus, similar event data can be produced.

3. PROCESS MINING IN THE SOFTWARE ENVIRONMENT

In this Section, we *generalize* our experiences described and argue that our findings apply to many other software systems. Several pragmatic *software practices*, which are helpful for integrating process mining in a generic software environment are presented here.

The GUI of *desktop* and *mobile* software systems are built using a variety of different client technologies. People distinguish between *rich client* (e.g., .NET WPF, Eclipse RCP, Objective C applications) and *thin client* applications (e.g, Google Andoid, JSF, HTML5 + Javascript). The *communication* between client and server can be done using HTTP protocol directly like in case of Web or Web/REST Services or using different remoting frameworks like RMI, IIOP, JMS, which normally rely on the TCP/IP stack. A coarse view on different types of client/server architectures is given in Fig. 9. A rich client can communicate with the server directly or using a proprietary protocol (an example of TOMA protocol was given in Section 2.1), and a web client accesses the server over http (an example was given in Section 2.2).

In order to apply *software process mining*, it is necessary to produce the *event logs*. In case of a *rich client with proprietary protocol*, like in our first case study, it is possible to write the logs on the application server (see "Event Logs (1)" in Fig. 9), since request messages contain the data about the user behavior and the identity of the user. In case of *normal rich client*, the server request does not reflect the user behavior, furthermore, the client can contain its own business logic and workflows. Thus, it is necessary to make logging on the client side, see "Event Logs (2)". Event logs from different rich clients must be assigned a user identity and be merged together in a combined log. In case of *web client*, the user logic is distributed among browser, web and application server. Depending on the particular technology it is possible to produce logs on the server side or on the client side, see "Event Logs (3)". For example, in case of HTML5, the client side containing powerful Javascript libraries can intercept all the user events and call the logging service on the server side (similar approach is used by Google Analytics).

The *event log entries* reflecting the user behavior have *similar structure*:
$[caseid, action, object, userid, timestamp]$. The client session id is normally taken as a case id, like in our second case study in Section 2.2. The user actions are applied to some widgets or dialogs (objects). For example: load window, input address, click search button, etc. Therefore, action and object together normally represent a user activity. User id reflects the organizational aspect and timestamps are used to analyze the performance aspect. As presented in the case studies, *additional attributes* can be logged in order to reflect informational, security, transactional, infrastructural and other software aspects.

From the point of view of *software architecture* needed for process mining, *logging* is normally seen as a separate *aspect*, which should not be mixed with the business logic. Thus, for all the cases listed above, event logging should be implemented either as a separate aspect (in terms of aspect-oriented programming) or using *listeners* (observer pattern) for different GUI forms and widgets.

For broad industrial usage, it is useful to implement persistence and management of event logs as a *separate service* with a separate *storage*. Every activity of the user produces an event. Depending on particular client technology, listeners or aspects are notified, and separate service for storing event data is called. Event log is stored separately and is separated from the core of the software system. Therefore, process mining can be seen as an *independent service*, see Fig. 9. Thus, different user behavior events are sent to the *process mining service*. The service formats, filters and normalizes the data and stores it into a separate *process mining database*; relational, but also NOSQL databases can be used here. The *process mining core* comprises a set of process mining techniques and algorithms.

In *both companies* where we did the case studies, IT services departments used different application performance monitoring and failure tracking tools. Almost all of these tools were available as *separate services* deployed in the intranet cloud. *Cloud services* become more and more effective and applicable nowadays [6]. Since, depending on the project, different amounts of data and different computational performance is needed, it is worthwhile also to deploy process mining service in a cloud as a *SaaS (software as a service)* service. In this case, different projects get access to the service without deploying and installing the mining software. Moreover, cloud infrastructure, which provides *scalability*, *elasticity* and dynamic resource provisioning, enables solving process mining problems of an extremely high complexity.

4. RELATED WORK

Process mining is a developed research field providing a set of mature methods and techniques [27, 28]. For the *process mining introduction*, the interested reader is also referred to

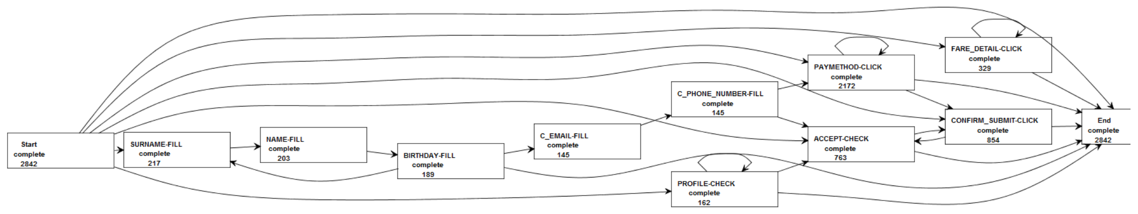


Figure 8: Typical workflow as a Heuristic net

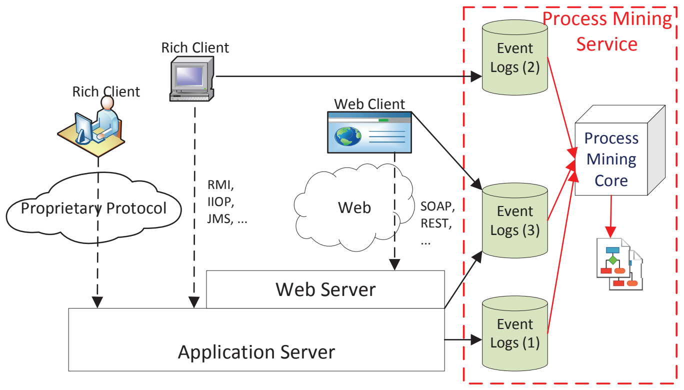


Figure 9: Overview of Software Architecture

the process mining website *www.processmining.org*. Process mining was successfully applied in a set of business areas, such as healthcare [16, 31, 21], insurance [25], auditing [12], hardware test processes [22] and even vessel behavior [15]. In this paper we present case studies from the other application area – tourist industry.

In our research we look at user behavior analysis from a software engineering perspective. Some preliminary research in this area already exists. The ideas about using process mining for detecting abnormal behavior in social networks were described by Sahlabadi et al. [23]. An approach to the analysis of farmers interaction with the decision support systems was presented by Maruster et al [17].

Generally, a lot of contributions have been made in the area of user behavior analysis and especially in web analytics [13] und user activity tracking [3]. Such tools as Piwik [18] and Google analytics [20] are widely used in this domain and accepted both by the research and industrial communities. Also a lot of work has been done in web usage mining [19, 7]. A separate domain here is mining search query logs [5]. The existing approaches normally use statistical and/or data mining methods for analysis of user behavior in Web. Process mining can be seen here as an important enhancement of web analytics, which provides a holistic process view on the behavior. Also the domain of *software runtime analysis* with the help of model generation [14] and mining [2, 9] deals with applying mining methods to software, however dealing not with the user but with the runtime perspective.

5. FUTURE WORK AND CONCLUSIONS

In this paper, we used two industrial projects to demonstrate how process mining can be used for deriving and analyzing the software *user behavior*. Our experiments were done with the help of a commercial tool *Disco* and the open-source platform *ProM*. We generalized our experiences and presented *software process mining* as a separate important *application domain*. Our goal was to motivate the need for *foundational research* in the area of software process mining and especially mining the software usage. Based on our experiences we arrived to a conclusion that process mining is an advanced technique which can be used for analyzing the user behavior for almost every software system, without focusing on a particular business domain.

In the future, from the point of view of industrial process mining tools, it makes sense to see process mining as a "clever" *analysis and monitoring service*. It is important to be able to embed *event tracking* to the existing software systems and to provide *process mining in the cloud* in order to achieve scalability and elasticity. We also see an urgent need to support the automated adjustment of the abstraction level of logs and models and better support of intelligent filtering. Practical event logs and models are huge and we have to be able to do process mining for "big data" effectively.

6. ACKNOWLEDGEMENTS

Support from the Basic Research Program of the National Research University Higher School of Economics is gratefully acknowledged.

7. REFERENCES

[1] Amadeus. Amadeus Toma Basic (Trainingshandbuch). `http://www.amadeus.com/at/documents/aco/at/de/Amadeus%20TOMA%20Basic.pdf`, 2014.

[2] G. Ammons, R. Bodík, and J. R. Larus. Mining specifications. *SIGPLAN Not.*, 37(1):4–16, Jan. 2002.

[3] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 203–212, New York, NY, USA, 2006. ACM.

[4] J. Barjis. The importance of business process modeling in software systems design. *Science of Computer Programming*, 71(1):73 – 87, 2008.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: Model and applications. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 609–618, New York, NY, USA, 2008. ACM.

[6] R. Buyya, J. Broberg, and A. M. Goscinski. *Cloud Computing Principles and Paradigms*. Wiley Publishing, 2011.

[7] F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos. Are web users really markovian? In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 609–618, New York, NY, USA, 2012. ACM.

[8] M. Dumas, W. van der Aalst, and A. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.

[9] R. Gombotz, K. Baïna, and S. Dustdar. Towards web services interaction mining architecture for e-commerce applications analysis. In *Proceedings of the Conference on E-Business and E-Learning*, 1999.

[10] C. Günther and W. Aalst. Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer-Verlag, Berlin, 2007.

[11] M. Haesen, K. Luyten, and K. Coninx. Get your requirements straight: Storyboarding revisited. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT '09, pages 546–549, Berlin, Heidelberg, 2009. Springer-Verlag.

[12] M. Jans, M. Alles, and M. Vasarhelyi. The case for process mining in auditing: Sources of value added and areas of application. *International Journal of Accounting Information Systems*, 14:1–20, 2012.

[13] A. Kaushik. *Web Analytics: An Hour a Day*. Sybex, June 2007.

[14] D. Lorenzoli, L. Mariani, and M. Pezzè. Automatic generation of software behavioral models. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, pages 501–510, New York, NY, USA, 2008. ACM.

[15] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst. Analyzing vessel behavior using process mining. In *Situation Awareness with Systems of Systems*, pages 133–148. 2013.

[16] R. Mans, M. Schonenberg, M. Song, W. Aalst, and P. Bakker. Process Mining in Healthcare: A Case Study. In L. Azevedo and A. Londral, editors, *Proceedings of the International Conference on Health Informatics (HEALTHINF'08)*, pages 118–125. Institute for Systems and Technologies of Information, Control and Communication (INSTICC), 2008.

[17] L. Maruster, N. R. Faber, R. J. Jorna, and R. J. F. van Haren. A process mining approach to analyse user behaviour. In J. Cordeiro, J. Filipe, and S. Hammoudi, editors, *WEBIST (2)*, pages 208–214. INSTICC Press, 2008.

[18] S. Miller. *Piwik Web Analytics Essentials*. Community experience distilled. Packt Publishing, Limited, 2012.

[19] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, Aug. 2000.

[20] B. Plaza. Google analytics for measuring website performance. *Tourism Management*, 32(3):477 – 481, 2011.

[21] A. Rebuge and D. R. Ferreira. Business process analysis in healthcare environments: A methodology based on process mining. *Inf. Syst.*, 37(2):99–116, Apr. 2012.

[22] A. Rozinat, I. de Jong, C. Günther, and W. Aalst. Process Mining of Test Processes: A Case Study. BETA Working Paper Series, WP 220, Eindhoven University of Technology, Eindhoven, 2007.

[23] M. Sahlabadi, R. C. Muniyandi, and Z. Shukur. Detecting abnormal behavior in social network websites by using a process mining technique. *Journal of Computer Science*, 10:393–402, March 2014.

[24] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.

[25] S. Suriadi, M. T. Wynn, C. Ouyang, A. H. ter Hofstede, and N. van Dijk. Understanding process behaviours in a large insurance company in Australia : a case study. In *25th International Conference on Advanced Information Systems Engineering, CAiSE 2013*, pages 449–464, Valencia, Spain, 2013. Springer.

[26] K. N. Truong, G. R. Hayes, and G. D. Abowd. Storyboarding: An empirical determination of best practices and effective guidelines. In *Proceedings of the 6th Conference on Designing Interactive Systems*, DIS '06, pages 12–21, New York, NY, USA, 2006. ACM.

[27] W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.

[28] W. van der Aalst. Process Mining. *Communications of the ACM*, 55(8):76–83, 2012.

[29] H. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst. ProM 6: The Process Mining Toolkit. In M. L. Rosa, editor, *Proc. of BPM Demonstration Track 2010*, volume 615 of *CEUR Workshop Proceedings*, pages 34–39, 2010.

[30] A. Weijters, W. Aalst, and A. Medeiros. Process Mining with the Heuristics Miner-algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.

[31] W.-S. Yang and S.-Y. Hwang. A process-mining framework for the detection of healthcare fraud and abuse. *Expert Systems with Applications*, 31(1):56 – 68, 2006.