Towards Retrograde Process Analysis in Running Legacy Applications

Marius Breitmayer, Lisa Arnold and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{marius.breitmayer,lisa.arnold,manfred.reichert}@uni-ulm.de

Abstract. Process mining algorithms are highly dependent on the existence and quality of event logs. In many cases, however, software systems (e.g., legacy systems) do not leverage workflow engines capable of producing high-quality event logs for process mining algorithms. As a result, the application of process mining algorithms is drastically hampered for such legacy systems. The generation of suitable event data from running legacy software systems, therefore, would foster approaches such as process mining, data-based process documentation, and process-oriented software migration of legacy systems. This paper discusses the need for dedicated event log generation approaches in this context.

Keywords: legacy systems, process mining, code analysis, event log

1 Introduction

Software applications are implemented to address the needs of users, use cases, and business processes. However, the majority of common software systems (e.g., legacy systems or individual software solutions) have not been designed with the goal to provide high-quality process-related event logs that allow for comprehensive process analyses and visualizations with modern process mining tools. Relevant questions emerging in legacy software modernization projects include, for example, how the process implemented by the legacy software system is structured (*Process Discovery*) or to what extent its execution deviates from a predefined to-be process (*Conformance Checking*). Currently, there exist three basic approaches to obtain process models:

1. **Log analysis** uses existing logs (e.g., event logs) to reconstruct the implemented process based on audit or workflow data. Consequently, the quality of the resulting process model is directly correlated with both the existence and quality of corresponding event logs [2,3]. However, a vast majority of individual applications and legacy systems are often unable to provide appropriate event logs. Moreover, even database-centric applications typically do not provide transaction-level audit data. Consequently, there has been no effective entry point for process mining yet.
2. **Interviews** may be conducted to discover the desired process model as perceived by key users and process owners [9]. Additionally, data models may be parsed to identify effects of processes on corresponding data. Analyzing such data models enables assumptions on the underlying processes.

J. Manner, D. Lübke, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 14th ZEUS Workshop, ZEUS 2022, Bamberg, held virtually due to Covid-19 pandemic, Germany, 24–25 February 2022, published at http://ceur-ws.org

This approach, however, is very time consuming and paved with both misunderstandings and misconceptions. In addition, interviews do not ensure completeness of the relevant processes and their various aspects, as they often neglect exceptions or specific process perspectives (e.g., data, time).
3. **Pattern recognition** attempts to identify typical process patterns in various data pools using algorithms from the field of artificial intelligence [1]. The algorithms require a deep analysis and learning phase prior to their application to the raw data. This is a time-consuming, cost-intensive, and fuzzy approach, which is therefore hardly pursued.

In the context of legacy systems, however, none of the presented approaches is easily applicable. All three approaches have in common that the business processes (and event logs), implemented by the legacy software systems, need to be represented accurately. Since most individual software solutions do not necessarily use process engines capable of delivering suitable process data, alternative approaches are required. One approach to tackle this challenge is, to observe process participants during process execution and to record their interactions with the software system resulting in a fine-grained documentation.

Section 2 describes the proposed solution approach. Section 3 discusses related work. Finally, Section 4 provides a summary and outlook.

2 Solution Approach

A human-centered business process can be defined as a sequence of user interactions with a software application, where each interaction is subject-bound (i.e., part of the same transaction). In legacy systems, such processes can be initiated and terminated by suitable actions (e.g., pre-defined key combinations or menu items). Adding such actions to an event stream with the associated application object (e.g., an order identified by its unique order number), subsequently, process mining tools will have process related event logs as input. The collected event data may then constitute the basis for a plethora of use cases, such as process documentation, process mining, and process-oriented cost estimations for modernizing legacy software systems (i.e., software migration). We aim to create different logging variants for existing legacy production systems:

1. **Dedicated recording** documents existing processes by assigning related program components. Users may determine the start and end of the recording using predefined key combinations, thus precisely delimiting all activities that constitute the recorded process (or the considered process part).
2. **Silent recording** tracks the entire usage of the application from the first login until closing the application. A decision can be made as to whether this should be done for all sessions or only for selected user sessions (e.g., only sessions of users from a certain department). Furthermore, it may be configured, which information should be stored (e.g., to ensure compliance with data protection requirements).

To minimize the performance effects of these recording on running applications, we rely on existing logging mechanisms of the application infrastructure.

For Oracle applications using a WebLogic Server, for example, *Oracle Diagnostic Logging* (ODL) offers extensive possibilities to manage application information via the administration console. Among others, oracle logger classes (e.g., *Application Development Framework*) may use this information through ODL handlers [15]. In Single Page Applications (e.g., the Oracle JavaScript Extension Toolkit JET), the primary object is known, however, the context between multiple process steps may get lost due to the loose coupling of user sessions and services. Even applications based on Oracles Forms allow adding appropriate message calls for each PL/SQL unit.

Using existing system logging functionality, the recording quality is significantly increased compared to purely mining the data model, as user interactions can be unambiguously linked to the process, program code, and associated data.

Fig. 1 depicts the approach. In a first step we identify relevant objects using information from the database and the source code of the application. However, especially in databases of legacy systems, assumptions such as good normalization or even the existence of foreign key constraints are often not applicable. The reason for this is that in many cases the logic is represented in the source code of applications rather than the database. By combining knowledge from the database (e.g., create, read, update, and delete -operations) and corresponding source code (e.g., code fragments corresponding to such operations), we are able to tackle this issue. After having identified process-relevant objects in both source code and database, we correlate them and add code tracking capabilities to the legacy system using, for example, the possibilities mentioned previously. This does then enable the generation of event logs from either dedicated or silent recording. These event logs may then be used during analysis.

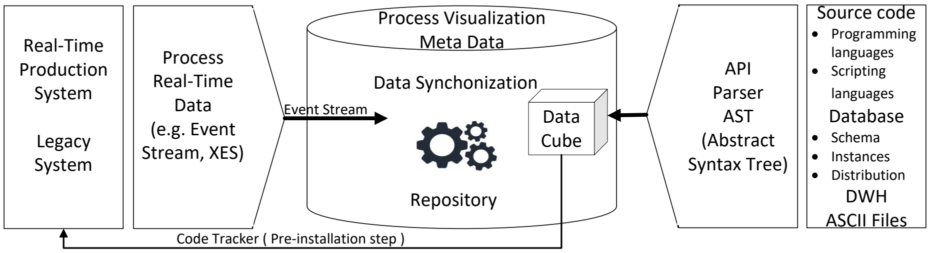
Real-Time Production System
Legacy System
Process Real-Time Data (e.g. Event Stream, XES)
Event Stream
Process Visualization Meta Data
Data Synchonization
Data Cube
Repository
API Parser AST (Abstract Syntax Tree)
Source code
Programming languages
Scripting languages
Database
Schema
Instances
Distribution
DWH
ASCII Files
Code Tracker (Pre-installation step)

Fig. 1. General approach

When analyzing event logs generated from such legacy systems, a valuable effect can be achieved that the three approaches described in Section 1 are unable to provide: If certain entries in the event stream are missing when comparing the event stream with the source code, this indicates that the process steps involved, although implemented and present, have never been used. This information is essential when removing technical debts and modernizing legacy systems [8].

3 Related Work

This work is related to the research areas process mining, event log generation, and code analysis. Process mining [2] provides techniques to discover business process models from event logs [16,12], to evaluate conformance between process event logs and models [6], and to enhance processes [3]. Existing process discovery approaches mainly focus on the control flow perspective while the data perspective is mostly neglected [13]. The latter is of particular interest for meaningful process analysis and improvements (e.g., legacy system migration to new software architectures).

Event log generation is concerned with the generation of event log based on various sources. In [11,4], approaches to record user activities based on desktop actions (e.g., for robotic process automation) are presented. Our approach is also able to correlate such desktop actions with the corresponding source code fragments and database operations, allowing for a more detailed event log generation. The case study presented in [14] discusses the generation of event logs from a real-world data warehouse of a large U.S. health system. While some challenges (e.g., correlating events) may also arise in the context of legacy systems, we plan to minimize required domain expert interviews by automatically extracting domain knowledge from the source code.

Code analysis comprises traditional analysis (e.g., style checking or data flow analysis [10]) and profiling (e.g., CEGAR [7] and BMC [5]) which, combined with process knowledge, yield great potential for software improvement and migration.

4 Conclusion and Outlook

This paper emphasizes the need for spending research efforts on the recording of high quality event data in legacy systems. This not only enables the application of existing process mining algorithms, but also additional use cases such as, for example, data-driven process documentation, facilitation software migration projects or cost reduction through process-driven development. Note that corresponding work is also relevant in the context of robotic process automation [17].

Acknowledgments This work is part of the SoftProc project, funded by the KMU Innovativ Program of the Federal Ministry of Education and Research, Germany (F.No. 01IS20027A)

References

1. van der Aalst, W.M.P.: Process discovery: Capturing the invisible. IEEE Computational Intelligence Magazine **5**(1), 28–41 (2010)
2. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer (2016)
3. van der Aalst, W.M.P., et al.: Process mining manifesto. In: Int'l Conf on BPM'11. pp. 169–194 (2011)

4. Agostinelli, S., Lupia, M., Marrella, A., Mecella, M.: Automated generation of executable rpa scripts from user interface logs. In: Business Process Management: Blockchain and Robotic Process Automation Forum. pp. 116–131. Springer International Publishing (2020)
5. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Carnegie Mellon University (2003)
6. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking. Springer (2018)
7. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) Computer Aided Verification. pp. 154–169. Springer (2000)
8. Cunningham, W.: The wycash portfolio management system. SIGPLAN OOPS Mess. **4**(2) (1992)
9. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, 2nd edn. (2018)
10. Khedker, U., Sanyal, A., Karkare, B.: Data Flow Analysis: Theory and Practice. CRC Press, Inc., USA, 1st edn. (2009)
11. Linn, C., Zimmermann, P., Werth, D.: Desktop activity mining - a new level of detail in mining business processes. In: Workshops der INFORMATIK 2018 - Architekturen, Prozesse, Sicherheit und Nachhaltigkeit. pp. 245–258. Köllen Druck+Verlag GmbH (2018)
12. Peña, M.R., Bayona-Oré, S.: Process mining and automatic process discovery. In: 2018 7th International Conference On Software Process Improvement (CIMPS). IEEE (2018)
13. Reichert, M.: Process and data: Two sides of the same coin? In: 20th Int'l Conf on Cooperative Information Systems (CoopIS'12). pp. 2–19. Springer (2012)
14. Remy, S., Pufahl, L., Sachs, J.P., Böttinger, E., Weske, M.: Event log generation in a health system: A case study. In: Business Process Management. pp. 505–522. Springer International Publishing (2020)
15. Vesterli, S.: Oracle ADF Survival Guide. Apress, Berkeley, CA, 1st edn. (2017)
16. Weerdt, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf Sys **37**(7), 654 – 676 (2012)
17. Wewerka, J., Reichert, M.: Robotic process automation - a systematic mapping study and classification framework. Enterprise Information Systems (2022)