PM4Py Web Services: Easy Development, Integration and Deployment of Process Mining Features in any Application Stack

Alessandro Berti[1], Sebastiaan J. van Zelst[1,2], and Wil van der Aalst[1,2]

[1] Process and Data Science Chair, Lehrstuhl für Informatik 9 52074 Aachen, RWTH Aachen University, Germany
[2] Fraunhofer Gesellschaft, Institute for Applied Information Technology (FIT), Sankt Augustin, Germany

Abstract. In recent years, process mining emerged as a set of techniques to analyze process data, supported by different open-source and commercial solutions. Process mining tools aim to discover process models from the data, perform conformance checking, predict the future behavior of the process and/or provide other analyses that enhance the overall process knowledge. Additionally, commercial vendors provide integration with external software solutions, facilitating the external use of their process mining algorithms. This integration is usually established by means of a set of web services that are accessible from an external software stack. In open-source process mining stacks, only a few solutions provide a corresponding web service. However, extensive documentation is often missing and/or tight integration with the front-end of the tool hampers the integration of the services with other software. Therefore, in this paper, a new open-source Python process mining service stack, PM4Py-WS, is presented. The proposed software supports easy integration with any software stack, provides an extensive documentation of the API and a clear separation between the business logic, (graphical) interface and the services. The aim is to increase the integration of process mining techniques in business intelligence tools.

Keywords: Process Mining · PM4Py · Web Services · Process Discovery · Case Management · Seamless Integration.

1 Introduction

Process mining [1] is a growing branch of data science which, starting from data stored and processed in information systems, aims to infer information about the underlying processes, i.e. as captured in the data. Several techniques, e.g., *process discovery* (automated discovery of a process model from the event data), *conformance checking* (comparison between an event log and a process model), *prediction* (given the current state of a process, predict the remaining time or the value of some attribute in a future event), etc., have been developed.

Process mining is supported by several open-source (ProM [5], RapidProM [9,2], Apromore [7], bupaR [6], PM4Py [3], PMLAB [4]) and commercial (Disco,

Celonis, ProcessGold, QPR ProcessAnalyzer, etc.) software. Apart from RapidProM (an extension of the data science framework RapidMiner) and Apromore, the majority of the open-source projects provide a standalone tool that only allows to import an event log and to perform process mining analyses on it. PM4Py and bupaR provide a set of process mining features as a library, and this provides integration with the corresponding Python and R ecosystem. At the same time, some commercial tools, e.g., Celonis and ProcessGold, as well as the Apromore open-source tool, offer a web-based interface supported by web services. This leads to some advantages:

- The possibility to access the information related to the process everywhere, from different devices.
- Identity and access management (typically unsupported by standalone tools).
- The possibility for multiple users to collaborate in the same workspace.

A systematic introduction of web services in the process mining field is offered by [8]. Process mining analyses offered through web services ideally permit an easy integration with other software solutions. In the case of Apromore, the business logic is offered to the web application mainly through servlets-based web services. This offers the possibility for external tools to use the algorithms integrated in Apromore by querying its web services. However, due to the high customization on the client-side, required to provide a higher number of features as application, Apromore does not allow easy embedding of the visual elements in external applications.

In this paper, the PM4Py Web Services (PM4Py-WS), that are built on top of the recent process mining library PM4Py [3], are presented. The high-level goals of the web services are (1) to provide an easy entrypoint for the integration of process mining techniques in external (business intelligence) tools, and, (2) to provide an extensible platform built on top of modern software engineering principles and guidelines (testing, documentation, clear separation between stacks, separation from the front-end). A prototypal process mining web application, supported by the services, is provided along with the services, in order to demonstrate that the services work as intended, and to provide some evidence that PM4Py-WS is easily integrated in any external application stack.

While in Python other ways to build non-trivial data visualization web interfaces are available, for example Dash by Plot.ly[3] that could offer an even simpler prototyping of process mining visuals based on PM4Py, they do not offer the same possibility of integration with external applications as exposing a set of web services, since the interface is more tightly coupled with the backend part.

The remainder of this paper is structured as follows. In Section 2, the architecture of the web services is explained. Section 3 provides information regarding the repository hosting the tool, the maturity of the tool and a reference to the video demo. Section 4 concludes this paper.

[3] `https://dash.plot.ly/introduction`

2 Architecture of the Web Services

PM4Py-WS is written in Python, i.e., a programming language popular among data scientists and developers for its simplicity and the vast set of high-performing data science libraries. The web-services are exposed as asynchronous REST GET/POST services using the Flask framework[4], which supports:

- Possibility to deploy the services using HTTP/HTTPS.
- Possibility to use an enterprise-grade server (e.g. IIS, UWSGI).
- Possibility to manage the Cross-Origin Resource Sharing (CORS[5]).

PM4Py Web Services are supported by the algorithms available in the PM4Py process mining library. The exposed services accept a *session identifier* (used to identify the user, and verify the general permissions of the user), and a *process ID* (used to identify the process, and verify the permissions of the user on the given log). Moreover, each service has access to a singleton object hosting all the software components needed for authentication, session management, user-log visibility and permissions, log management, and exception handling. Each different component is provided as a factory method, that means several different implementations are possible. Currently, the following components are provided:

- *Session manager*: Responsible for user authentication and verification of the validity of a session. Two different session managers are available:
  - *Basic session manager*: Supported by a relational database with user/-password information and a separate logs database table.
  - *Keycloak IAM Session manager*[6]: Users and sessions are verified through the Keycloak identity and user access control management solution, that is the most popular enterprise solution in the field. Thanks to Keycloak, several applications can share the same credentials and sessions. The configuration of user/password data, and the settings related to the session duration, are done in Keycloak.
- *Log management*: responsible to manage individual event logs, the visibility/permissions of the users w.r.t. the logs, and the analysis/filtering operations on the log. Each process is managed by a handler that controls the following operations on the log:
  - *Loading*: depending on the log handler, the log is either persistently loaded in memory, or loaded in memory when required. In the current version, two in-memory handlers are provided: a `XES` handler (that loads an event log in the `XES` format, and uses the PM4Py EventLog structure to store events and cases), and a `CSV` handler (that loads an event log in the CSV/Parquet[7] format, and uses Pandas dataframes[8]). These handlers both load event logs stored as files, however when the PM4Py library will be more mature, the file dependency might be gradually dropped.

[4] `http://flask.pocoo.org/`
[5] `https://de.wikipedia.org/wiki/Cross-Origin_Resource_Sharing`
[6] `https://www.keycloak.org/`
[7] A popular columnar storage format, see `https://parquet.apache.org/`
[8] A popular data processing framework in Python, see `https://pandas.pydata.org/`

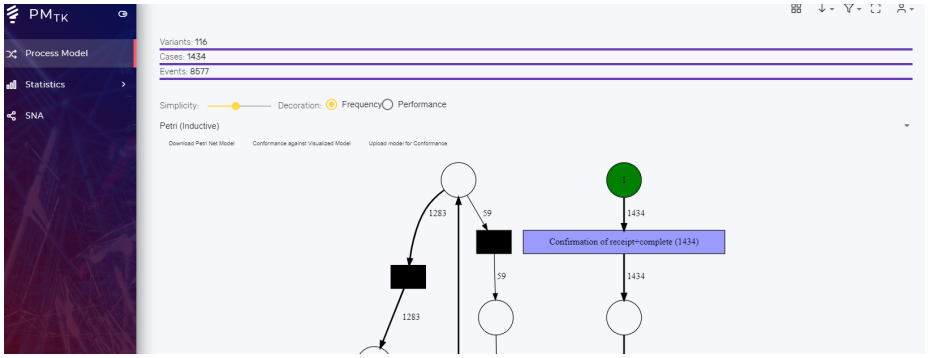

Fig. 1: PMTK prototypal web interface, using PM4Py-WS as an underlying service. The PMTK interface is available on Github at the address https://github.com/pm-tk/source.

  - *Filtering*: The filtering algorithms implemented in the PM4Py library are applied to the log when the filtering services (add filter, remove filter) are called.
  - *Analysis*: the algorithms implemented in the PM4Py library are applied to the log, in order to get the process schema, get the social network, perform conformance checking, etc.
- *Exception handler*: Triggered when debug warnings/error messages need to be logged. The default setting is using the logging utility in Python.

3 Repository of the Tool, Video Demo and Maturity

PM4Py-WS, along with the prototypal interface in AngularJS, is available at `https://github.com/pm4py/pm4py-ws`. The web services can be installed following the instructions contained in the INSTALL.txt file provided in the repository. A Docker image[9] is also made available with the name *javert899/pm4pyws* and could be run through the command *docker run -d -p 5000:5000 javert899 /pm4pyws*. The docker image uses port 5000, i.e., the web services are exposed at the URL `http://localhost:5000` and the prototypal Angular web interface is made available at the URL `http://localhost:5000/index.html`.

The documentation of the web services is available at the site `http://pm4py.pads.rwth-aachen.de/pm4py-web-services/`. A demo of the web services is made available at `http://212.237.8.106:5000/` (for example, the loginService and the getEndActivities GET services can be tested according to the documentation). A demo of the prototypal Angular web interface (represented in Figure 1) is made available at `http://212.237.8.106:5000/index.html`. A video demo, that shows how the web services can be easily queried through a browser, and shows the prototypal Angular web interface in action, is available at `http://pm4py.pads.rwth-aachen.de/pm4py-ws-demo-video/`.

[9] See `https://en.wikipedia.org/wiki/Docker_(software)` for an introduction to Docker

The web services have just been released and no real-life use case is available yet. The web interface is in a prototypal status.

4 Conclusion

In this paper, we presented PM4Py-WS, a stack of web services developed on top of the existing process mining python framework PM4Py. PM4Py-WS is open-source and extendible, and allows to integrate process mining technology in any external software stack.

When an algorithm is implemented in the PM4Py library, it is not immediately available in the PM4Py web services: a service exposing the algorithm should also be implemented. This does not hamper the scalability of the web services (in terms of number of available features), but requires an extra effort by the developer, that should work on both the PM4Py library and the PM4Py-WS projects.

In terms of scalability on big amounts of data, the current PM4Py-WS handlers are limited to a single core, and the performance suffers from the fact that not all the CPU cores are used. Some future work will consider to implement distributed handlers for logs, that would be able to manage bigger amount of data. We aim to actively develop PM4Py-WS, allowing faster integration and adoption of cutting-edge research into virtually any external tool.

References

1. van der Aalst, W.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. van der Aalst, W., Bolt, A., van Zelst, S.J.: RapidProM: Mine Your Processes and Not Just Your Data. CoRR **abs/1703.03740** (2017)
3. Berti, A., van Zelst, S.J., van der Aalst, W.: Process Mining for Python (PM4Py): Bridging the Gap Between Process-and Data Science pp. 13–16 (2019)
4. Carmona, J., Solé, M.: PMLAB: an scripting environment for process mining. In: Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014. p. 16 (2014)
5. van Dongen, B., de Medeiros, A.K., Verbeek, H., Weijters, A., van der Aalst, W.: The ProM Framework: A New Era in Process Mining Tool Support. In: International conference on application and theory of petri nets. pp. 444–454. Springer (2005)
6. Janssenswillen, G., Depaire, B., Swennen, M., Jans, M., Vanhoof, K.: bupaR: Enabling Reproducible Business Process Analysis. Knowl.-Based Syst. **163**, 927–930 (2019)
7. La Rosa, M., Reijers, H.A., van der Aalst, W., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: An advanced process model repository. Expert Systems with Applications **38**(6), 7029–7040 (2011)
8. Lambrechts, S., van der Aalst, W., Weijters, A.: Scenario-based process mining: Web servicing and automated scenario generation (2009)
9. Mans, R., van der Aalst, W., Verbeek, H.: Supporting Process Mining Workflows with RapidProM. In: BPM (Demos). p. 56 (2014)