PPINOT Computer and ppinot4py: Two Libraries to Compute Process Performance Indicators (Extended Abstract)

Manuel Resinas*, Adela del-Río-Ortega* and Antonio Ruiz-Cortés*
*SCORE Lab, I3US Institute, Universidad de Sevilla, Spain
{resinas,adeladelrio,aruiz}@us.es

***Abstract*—The ability to compute PPIs is of utmost importance for analyzing the performance of business processes. Most process mining tools support the computation of some types of PPI. However, in most cases they either just support a predefined set of metrics, which limits their usefulness in many scenarios, or the computation results are not designed to be used outside the tool platform and integrated with other tools or workflows. PPINOT Computer and `ppinot4py` are two libraries that were developed to overcome these limitations. Both libraries share the same approach to compute PPIs but serve two different purposes: one library has been designed to be integrated into custom solutions for process monitoring whereas the other has been designed to be part of data analysis and exploration workflows. The libraries have been successfully deployed in two different organizations and are used in several process management courses.**

I. Introduction

Process performance indicators (PPIs) are widely used as a mechanism to manage the performance of processes and provide essential information for decision making. PPIs are quantifiable metrics that allow the evaluation of the efficiency and effectiveness of business processes. They can be measured using data generated within the process and are aimed at process control and continuous optimisation [1].

Many process mining tools compute PPIs, especially those related to time. However, they present two limitations. First, in many cases, the PPIs are limited to a predefined set of metrics like the cycle time of the process and the cycle time of each activity. However, PPIs are domain specific and, in many occasions, they do not focus on processes or activities alone, but on process fragments [2]. For instance, in an incident management process, two typical PPIs are time-to-respond and time-to-fix, which include activities from the moment when the incident is registered to the moment when the user receives the first response and the moment when the incident is fixed, respectively. Second, commercial tools like Celonis do allow the definition of custom PPIs, but they are difficult to use outside their platform and to integrate them with other tools.

In this demo[1], we present two libraries that can be used to compute a wide variety of custom PPIs defined using the PPINOT metamodel and can be easily integrated with other tools and workflows. Next, we detail these features in Section II. Then, we detail their maturity in Section III.

II. Features

PPINOT Computer[2] and `ppinot4py`[3] are two alternative implementations to compute PPIs that follow a similar approach. First, the custom PPIs are specified following the PPINOT Metamodel (cf. Fig. 1) and then, the libraries use this definition to compute them for an event log that can be provided in different formats. Thus, the user can focus on what PPI she wants to define instead of how to compute it.

The PPINOT Metamodel allows the definition of PPIs with an emphasis on how the PPI is computed. It supports three types of measures. `Base Measures` are computed for each case and can be divided into time, count or data measures. `Time Measures` measure the time between two `TimeInstantConditions`. It supports work schedules and holidays, and mechanisms to handle situations where the `from` and `to` conditions occur several times in the same case by means of `Cyclic` time measures, which consider pairs of occurrences and use an `aggregationFunction` to combine them. `Count Measures` measure the number of times a `TimeInstantCondition` occurs in a case. `Data Measures` are used to obtain the value of a case or event attribute. If the value changes throughout the case, one can specify if we want the first value, last value or the value when a `precondition` is met. `Aggregated Measures` are used to aggregate the values of a `MeasureDefinition` using an `aggregationFunction` like sum, or average. One can also apply a `filter`, and the result can be `groupedBy` one or more values. Filters and groups are measures themselves. Finally, `Derived Measures` are used to compute a new measure by combining several measures using a `function`.

The metamodel depicted in Fig. 1 is not exactly the one described in [3], but some adaptations were made based on the experience of applying PPINOT Computer in several organizations. Furthermore, the metamodel was originally designed to model PPIs together with a process model. However, PPIs can be defined without any process model by directly referencing the elements of the event log from the `Conditions`.

This work has been partially supported by projects RTI2018-101204-B-C22 (MCI/AEI/FEDER, UE) and P18-FR-2895 (Junta de Andalucía/FEDER, UE).

[1]Video available at: https://www.youtube.com/watch?v=CK3KoKoeLHc

[2]https://github.com/isa-group/ppinot

[3]https://github.com/isa-group/ppinot4py

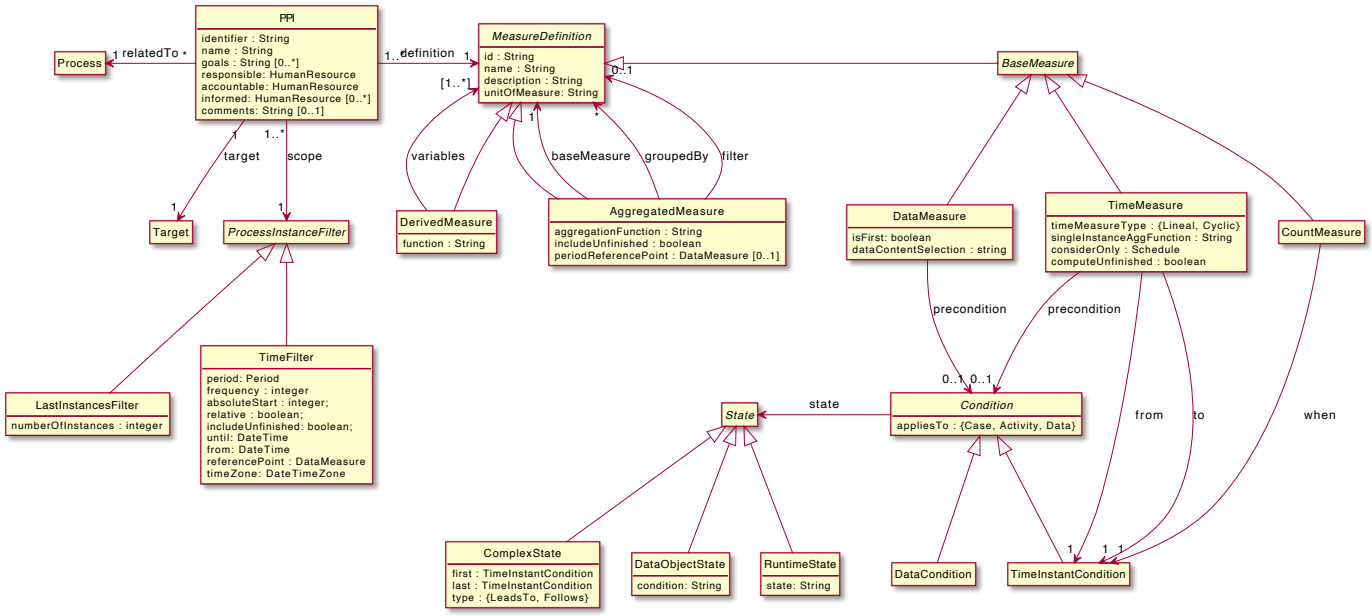

Fig. 1. Simplified version of the PPINOT metamodel implemented in the libraries

Although both libraries follow the same approach to compute PPIs, there are significant differences between them. The most obvious is that PPINOT Computer is developed in Java, whereas the `ppinot4py` is developed in Python. However, the difference goes beyond the programming language used. The Java library has been designed to be integrated into custom solutions for process analytics and dashboard design. To this end, it includes features like the ability to read the log from different data sources including databases like ElasticSearch, to load and save a collection of PPI definitions using a JSON format, and to wrap the library in a REST API that can be used in a microservices architecture. Furthermore, it supports the processing of large event logs that do not fit in memory, either by caching them in disk or by relying on external databases like ElasticSearch. Instead, `ppinot4py` has been designed to be part of more interactive data analysis and exploration workflows using tools like Jupyter notebooks, data analytics libraries like pandas, or process mining libraries like `pm4py`. It can also play a role in computing PPIs that can be used in predictive monitoring solutions developed in Python. For this reason, it leverages well-established Python libraries like `pm4py` to read the event log, and pandas to perform the computations. By doing so, it makes it possible to extend its functionalities with the capabilities provided by these libraries.

III. Maturity

PPINOT Computer is a mature tool that has been successfully deployed in two organizations. In both cases, its role is to compute the PPIs used to monitor the Service Level Agreements (SLAs) of external IT providers. Typical examples are the percentage of incidents solved in time, or the percentage of incidents solved without identifying the root cause. This monitoring was used to find inefficiencies and differences between IT providers, and to check the fulfillment of the SLAs and compute the penalties, if applicable. Therefore, the quality of the PPI measurement was critical to avoid conflicts between organizations. From a technical perspective, both deployments followed a microservices architecture. A REST API that wraps the library was implemented so that it can interact with the rest of the services. The event log was provided by a MongoDB server and the PPIs were defined using a JSON file. Besides these two projects, PPINOT Computer has been successfully used by students to define and compute PPIs in a process management course for more than six years. It has also been used by other researchers to add privacy to PPI values [4].

`ppinot4py` shares the same foundation as PPINOT Computer and its implementation learns from the experience acquired after developing PPINOT Computer during several years. However, its development is more recent and it does not support the same full set of features yet. Despite this, it has been successfully used in a performance management course and to compute PPIs for automatic dashboard generation [5].

References

[1] A. del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés, "Business process performance measurement," in *Encyclopedia of Big Data Technologies*. Springer, 2019.

[2] B. Estrada-Torres, A. del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés, "Modeling variability in the performance perspective of business processes," *IEEE Access*, vol. 9, pp. 111 683–111 703, 2021.

[3] A. del-Río-Ortega, M. Resinas, A. Durán, B. Bernárdez, A. Ruiz-Cortés, and M. Toro, "Visual ppinot: A graphical notation for process performance indicators," *Bus. Inf. Syst. Eng.*, vol. 61, no. 2, pp. 137–161, 2019.

[4] M. Kabierski, S. A. Fahrenkrog-Petersen, and M. Weidlich, "Privacy-aware process performance indicators: Framework and release mechanisms," in *CAiSE*, 2021, pp. 19–36.

[5] Ü. Aksu, A. del-Río-Ortega, M. Resinas, and H. A. Reijers, "An approach for the automated generation of engaging dashboards," in *OTM Conferences*, 2019, pp. 363–384.