Data underlying the paper: An agent-based process mining architecture for emergent behavior analysis

Rob H. Bemthuis[1,*], Martijn Koot[2,*], Martijn R. K. Mes[2], Faiza A. Bukhsh[3], Maria-Eugenia Iacob[2], and Nirvana Meratnia[1]

[1]Pervasive Systems, University of Twente,Enschede, The Netherlands
[2]Industrial Engineering and Business Information Systems, University of Twente, Enschede, The Netherlands
[3]Datamanagement and Biometrics, University of Twente, Enschede, The Netherlands
[*]Corresponding authors: {r.h.bemthuis,m.koot}@utwente.nl

ABSTRACT

The dataset[1] contains a collection of experiment results and event logs generated. The experiment comprises a job-shop scheduling problem, implemented in a discrete-event simulation model. The raw experiment results are given from which event log files can be generated by following the steps as described in this data paper or the referred paper[2]. A collection of event log files is given, as well as the raw files. The logs include the filtered part of the case study as presented in the paper “An agent-based process mining architecture for emergent behavior analysis” by Rob Bemthuis, Martijn Koot, Martijn Mes, Faiza Bukhsh, Maria-Eugenia Iacob, and Nirvana Meratnia.

Date published: 27th of August 2019; Date revised: 8th of January 2020

Erratum

Update 8th of January 2020: the used plugin in step 4.1 of the example contained a mistake, it should be "Replay a Log on Petri Net for Conformance Analysis" and not "Replay a Log on Petri Net for Performance/Conformance Analysis"

Paper abstract

The abstract of the academic paper[2] where this dataset paper is based on is as follows:

Information systems leave a traceable digital footprint whenever an action is executed. Business process modelers capture these digital traces to understand the behavior of a system, and to extract actual run-time models of those business processes. Despite the omnipresence of such traces, most organizations face substantial differences between the process specifications and the actual run-time behavior. Analyzing and implementing the results of systems that model business processes tend, however, to be difficult due to the inherent complexity of the models. Moreover, the observed reality in the form of lower-level real-time events, as recorded in event logs, is seldom solely explainable by higher-level process models. In this paper, we propose an architecture to model system-wide behavior by combining process mining with a multi-agent system. Digital traces, in the form of event logs, are used to iteratively mine process models from which agents can learn. The approach is initially applied to a case study of a simplified job-shop factory in which automated guided vehicles (AGVs) carry out transportation tasks. Numerical experiments show that the workflow of a process mining model can be used to enhance the agent-based system, particularly, in analyzing bottlenecks and improving decision-making.

Paper keywords: Multi-agent System; Process Mining; Emergent Behavior, Enterprise Architecture; Supply Chain Logistics; Job-shop; Internet of Things.

Files

The raw results from the simulation model are presented in the repository [RawData/Events.txt] and the filtered event logs are stored in the repository [FilteredFiles/Experiment.xes].

RawData/Events.txt

Describes the event data generated as output of the simulation study conducted. Upon request, we can provide more experimental results, for that please contact the corresponding authors.

File naming convention
ExperimentXYZ.tzt, where:

- X = number of vehicles {4,5,6};
- Y = vehicle driving direction {1 = forward; 2 = backward; 3 = forward and backward};
- Z = vehicle dispatching rule {1 = random; 2 = longest waiting time; 3 = nearest vehicle}.

File format
.txt with tab-separated values.

File content
- ID = unique identifier of event;
- Timestamp = YYYY/MM/DD HH:MM:SS.MS;
- Product = type of product {Console; Helicopter; Robot} followed by a unique identifier;
- Type = type of product {Console; Helicopter; Robot};
- Event = activity {Arrival; Drain; Drilling; Painting; Sawing; Transport; Welding};
- Status = life cycle {Blocked; Complete; In progres; Start; Waiting};
- Resource = additional information about a utilized resource such as entering source (e.g., JobShop.Source; JobShop.Buffer), departing sink (e.g., JobShop.Drain), vehicle (e.g., AGV:1; AGV:2), machine entrance buffer (e.g., JobShop.Welding.Input), machine process (e.g., JobShop.Welding.Machine), and machine exit buffer (e.g., JobShop.Welding.Output).

Additional remarks
- All experiments are conducted with the same random seed values;
- Only one replication is conducted per experiment;
- The run length of one experiment is 24 hours;
- A warm-up period is not taken into account;
- .MUs = moveable units.

FilteredFiles/Experiment.xes
The *Experiment.xes* files are the input files for determining the quality metrics of the process models and the key performance indicators (e.g.,throughput times).

File format
.xes = Extensible Event Stream

File naming convention
XYZ.xes, similar to naming convention of [RawData/Events.txt]

Example
This part describes an example of going from the raw simulation output files to process model quality metrics and the throughput time KPI. This example considers a manual execution of all tasks.

0. We consider in this example the experiment 411. Used tool: ProM Lite 1.2; Experiments are conducted on the 31st of May 2019;
1. Convert the event logs from .csv to .xes format:

    1.1. Import the .txt file into ProM;

    1.2. Convert .txt to .xes using the ’Convert CSV to XES’ filter:

1.2.1. In the Configure Conversion from CSV to XES form, select as Case Column = Product; Event Column = Event; Start Time = Timestamp and erase the Completion Time declaration;

1.2.2. Open Show Expert Configuration and set XES Extension for Type to “concept:instance (Concept)”, for Status to “lifecycle:transition (Lifecycle)”, and for Resource to "org:resource (Organization)”;

1.2.3. Check: processes = 1; cases = 1450; events = 47575; event classes = 24; event types = 4; originators = 19.

2. Filter Log using Simple Heuristics plugin:

   2.1. Apply log filter to all event types;

   2.2. Start events = Arrival+Start;

   2.3. End events = Drain+Complete;

   2.4. Event filter = include all events;

   2.5. Check: processes = 1; cases = 1296; events = 44880; event classes = 24; event types = 4; originators = 19.

3. Construct the Petri-net by using the Alpha Miner plugin:

   3.1. Configure Alpha Miner: Event Classifier = (Event Name AND Lifecycle transition); Version = Alpha.

4. Replay the log on the Petri-net for performance/conformance analysis:

   4.1. Use the “Replay a Log on Petri Net for Conformance Analysis” plugin with the constructed Petri-net and Filtered event logs;

   4.2. Create a final marking by adding ([Drain+Start],[Drain+Complete]) to the Candidate Final Marking list;

   4.3. Select as classifier in the ’Map Transitions to Event Classes’ the ’Event Name’;

   4.4. Verify that Measuring fitness is checked. Do not penalize improper completion.

5. Open the visualizer:

   5.1. In the Inspector go to the tab Info and select Global Statistics (non-filtered traces);

   5.2. Verify that the Trace Fitness is 0.9997733410493826;

   5.3. Change the visualization to Time Between Transition Analysis (PNetAlignmentAnalysis);

   5.4. Check that the average time from Arrival+Complete to Drain+Complete is 73.34 minutes.

6. Go further by using as input (1) Replay result, (2) filtered event logs, and (3) constructed Petri-net:

   6.1. Select the “Measure Precision/Generalization” plugin;

   6.2. Verify that the precision equals 0.21834 and the generalization equals 0.77938.

References

1. Bemthuis, R. H. *et al.* Data underlying the publication: An agent-based process mining architecture for emergent behavior analysis. *4TU.Centre for Research Data* DOI: 10.4121/uuid:9e430177-1dd0-40e9-b48a-8eb39124ef4c (2019).

2. Bemthuis, R. H. *et al.* An agent-based process mining architecture for emergent behavior analysis. In *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*, 54–64, DOI: 10.1109/EDOCW.2019.00022 (2019).

Acknowledgements

This project is funded by the Netherlands Organization for Scientific Research (NWO) (under grant 628.009.015). The authors thank all partners of the DataRel project.