Final report Scylla sustainability plugin

Kareem Ali Abou-sena, Mihail Atansov, Jing Yao Seow, Wen-Chun Chen, Hui-Yu Yu

08.02.2024

Abstract

This report features the documentation of both product and team as required for the following, ”Bachelor Practical Course (INHN0021) Building a BPM Research App - Campus Heilbronn”.

Contents

1 What is the customer trying to achieve? 1

2 Description of Product 2

2.1 Plugin functionality . . . 2

2.1.1 CostDriverGCParser Plugin . . . 3

2.1.2 CostDriverSCParser Plugin . . . 3

2.1.3 CostDriverLogging Plugin . . . 4

2.2 Visualisation . . . 5

2.3 Testing . . . 6

2.3.1 ExecutionLoggingPluginTest . . . 6

2.3.2 GCParserTest & SCParserTest . . . 7

2.3.3 Acknowledgment of ScyllaScripts.java . . . 7

3 Documentation of Process 7

3.1 Organization of project . . . 7

3.2 What insights we gained from working in and on such projects . . . 8

3.2.1 Retrospective Meetings . . . 9

3.2.2 Design Implementation . . . 9

4 Limitations and Future Work 10

1 What is the customer trying to achieve?

Whilst the concern for the world' s ecosystem seems to grow, businesses that have adapted unsustainable and harmful business processes must change their ways. Thus, the environmental impacts of business processes have become an important factor that needs to be taken into consideration. Business process model and notation (BPMN) have been introduced to organizations to allow them to construct models and store their business processes digitally. The paper [1] suggests to use their framework which is a framework that focuses on sustainability-oriented process analysis, which combines Life Cycle Assessment (LCA) and activity-based costing (ABC) within the BPM life-cycle [1]. The basic idea of the LCA score is that all environmental burdens attached to the product or service have to be assessed, meaning from the first step of collecting raw materials all the way to waste removal [2]. These LCA scores are stored inside the activities of the business processes.
With the context given, we can explain what the business requirements were. From the initial presentations given in October, it was understood that the customers wanted a product where it would provide a sustainability assessment, where it was clear which activities/events within the business process harmed their overall rating when it came to the sustainability evaluation. The customers would basically want a business process simulator which then was able to evaluate the business process sustainability score. The customers wanted team BearCrow to work on already existing business-process called Scylla. Scylla was chosen because it is extensible, which allowed a friendly-plugin environment, allowing us to integrate new data such as the LCA score. Thus, Scylla was used to help fulfill the customer's requirements. Following this, the plugin we had to develop should be able to parse the activities contained within the business processes, obtain the respective LCA scores, and calculate certain defined values. These defined values such as Average environmental activity cost would ease analysts in understanding which activities within the business process carry the worst LCA scores or which activities are causing the business process to have a poor sustainability 'performance'. Team BearCrow would then have to construct parsers which would read the configurations of the business processes and obtain the LCA scores. Then the plugin would calculate definition 6,7,8,9 from the SOPA paper [1]. With the calculated definitions and with the help of SimuBridge, these results would then be visualized to help ease analysis. Below are the definitions which would help understand what the customers in evaluating the sustainability of the business process.

Definition 1. (6) (Environmental Activity Instance Cost) Consider an activity instance, which is a set of concrete cost drivers. Cost drivers determine the cost of an activity. The environmental activity instance cost is a summation of all concrete cost drivers. Where a is an activity a ϵ A, and A is a set of activities and Q is a set of concrete environmental cost drivers Q $\subset$ P(C). Thus let (a,Q) ϵ I be an activity Instance

$$activity\_instance\_cost(a, Q) = \sum_{q \epsilon Q} cost(q) \tag{1}$$

Definition 2. (7) (Environmental Process Instance Cost) A process instance is a set of activity instances, so the environmental process instance cost is calculated by doing a summation of all the activity instances. Let T $\subseteq$ I* be a set of process instances and t ϵ T.

$$process\_instance\_cost(t) = \sum_{(a,Q)\epsilon t} activity\_instance\_cost(a, Q) \tag{2}$$

Definition 3. (8) (Average Environmental Activity Cost) The average environmental activity cost is calculated by counting the activity instances of an activity with all the combinations of possible concrete cost drivers, and multiplying it with the respective activity instance cost, and then dividing the multiplication of the two by the total occurrences of the activity. Let T be a multiset ϵ B(I*) of process instances.

$$average\_activity\_cost(a, T) = \frac{\sum_{Q\epsilon P(C)} \sum_{t\epsilon T} specific\_count(a, Q, t) \cdot activity\_instance\_cost(a, Q)}{\sum_{t\epsilon T} occurrence\_count(a, t)} \tag{3}$$

Definition 4.(9) (Average Environmental Process Instance Cost) The average environmental process instance cost is calculated by taking a summation of the process instance cost and dividing it by the total number of process instances.

$$average\_process\_instance\_cost(T) = \frac{\sum_{t\epsilon T} process\_instance\_cost(t)}{\mid T \mid} \tag{4}$$

To help further understand and motivate why we were assigned to create this plugin, an example should help. Consider the business process of food delivery for a pizza service. The delivery of the pizza has the choice of driving with a bicycle or a diesel-fueled car. Considering the choices relative to LCA scores, i.e. their impact on the environment, the bicycle is the better choice. The plugin would show this by displaying the defined calculations and proving that using the bicycle is the better choice regarding LCA scores. This is a simple example, but one can imagine how choices like these on a much larger scale can have an impact, and why the plugin we were assigned to create can be of great use for organizations trying to shift towards being green.

2 Description of Product

2.1 Plugin functionality

The plugin operates within Scylla, an open-source BPMN extensible simulation tool with a plugin system. Scylla facilitates discrete event simulation (DES) by processing input files, including general resource information, BPMN model files, and simulation configurations. These inputs generate simulated process instances, documented in XES event logs [3].

Developers interact with Scylla's plugin system through four distinct entry points: parsing, initialization, execution, and reporting. These entry points align with Scylla's internal structure, as depicted in [Figure 2].

[Figure 1] illustrates Scylla's overall structure, which requires three input files—global configuration, simulation configuration, and the BPMN model—for simulation. The simulation manager

oversees the process, initializing activities, simulating them, and documenting results in event logs (*.xes files).

We developed three plugins to handle sustainability data, extending two pluggable types. CostDriverGCParser and CostDriverSCParser Plugins, extending Parser Pluggable, parse global and simulation configurations separately. The CostDriverLogging Plugin rewrites sustainability information into event logs.

In accordance with SOPA guidelines, preprocessed data, such as average activity instance cost and process instance cost, are incorporated. Aggregated and rearranged data are stored separately in a new file (statistic.xml) for clarity.

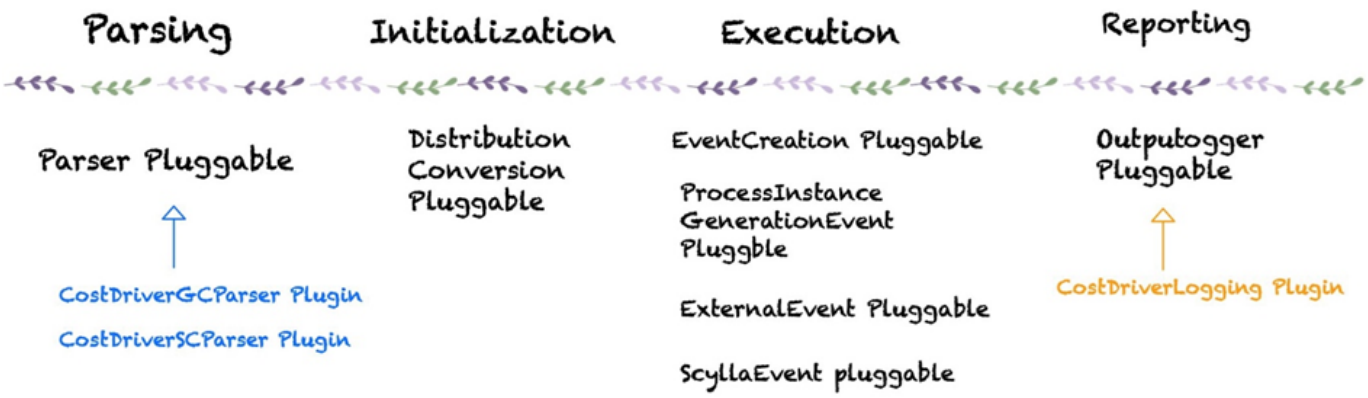

Figure 1: Entry points in Scylla [4]

2.1.1 CostDriverGCParser Plugin

CostDriverGCParser plugin extends from the entry point GlobalConfigurationParserPluggable. This parser plugin parses the sustainability information stated in the global configuration which contains details of each abstract and concrete cost variant in their class wrapper. This information will be put in the extensionAttributes in SimulationInput as CostDriverSCParser Plugin. The reason we build a class wrapper for each datatype is so that future extension is possible if there exists a need to include more information in the cost drivers.

2.1.2 CostDriverSCParser Plugin

CostDriverSCParser Plugin extends from the entry point SimulationConfigurationParserPluggable. Scylla has an existing data wrapper called SimulationConfiguration. This parser plugin has two aims:

1. Map the process activities with the cost driver's ID
2. Parse the cost variant information and store them in our new class wrapper CostVariantConfiguration and CostVariant.

After these steps, the plugin will put them into the extensionAttributes in SimulationInput which is the parent abstract class for SimulationConfiguration. The simulation Manager will then call the plugins before the simulation starts.

2.1.3 CostDriverLogging Plugin

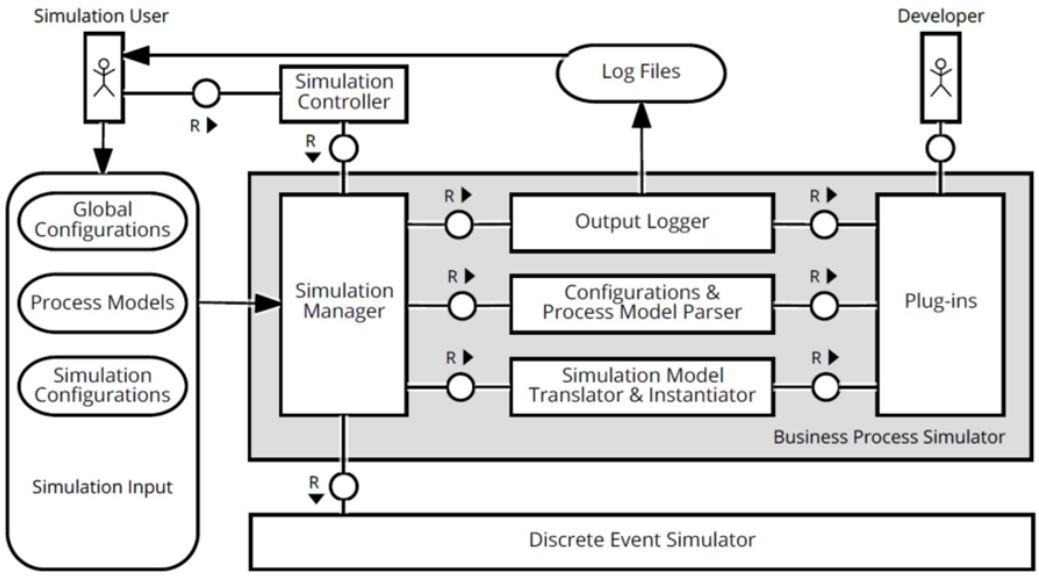

Figure 2: Structure of Scylla
[3]

In the CostDriverLogging Plugin, we map the process instance ID with the cost driver's ID, then link the cost driver's ID with the classes we created in the CostDriverGCParser Plugin. The essential step in this plugin is writing the environmental-related data into the event log and statistic file. Therefore, after the mapping is finished, the logger plugin's main task is to write those into both the event log and statistic file.

The output of our plugin would be four files with an additional one enclosed with statistic.xml. We separate the newly collected information into two separate files, one file containing individual data with different formats stored in the event log along with the newly created statistic output file, and the other file contains aggregated information mainly put in the statistic output file. For the individual activity cost, the result provides two formats, one encloses the concrete and abstract cost driver with its cost. It is called the count and it references the process instance id inside the event log [Figure 3] while the other format reorganizes those into a subjective function, mapping both activity and cost variant together to *activity_instance_cost* [Figure 4]. Definition 1 Environmental Activity Instance Cost is *cost:activity* in the event log [Figure 3]. Definition 2 Environmental Process Instance Cost is *cost:Process_Instance* in the event log [Figure 3]. Definition 3 Average Environmental Activity Cost and Definition 4 Average Environmental Process Instance Cost are in the *statistic.xml file which is

not shown here.
The sample of shipping logistics and the plugin package can be found in the public GitHub here.

```
<trace>
    <string key="concept:name" value="[Process_Instance_ID]"/>
    <string key="cost:Process_Instance" value="[Total Cost]"/>
    <string key="cost:variant" value=[Cost Variant A]/>
    ...
    <event>
        <string key="cost:driver" value=[Abstract Cost Driver(Concrete Cost Driver): [cost]]/>
        <string key="cost:driver" value=[Abstract Cost Driver(Concrete Cost Driver): [cost]]/>
        <string key="concept:name" value=[Activity]]/>
        <string key="lifecycle:transition" value="start"/>
        <date key="time:timestamp" value="2023-12-25T09:00:00+01:00"/>
        <string key="cost:activity" value=[activity cost]/>
    </event>
    ...
</trace>
```

Figure 3: Format of Individual Activity Cost in *event log*

```
<Sustainability_Info>
    <Average_Cost_Variant_Cost id="[id]">1.4019512857142856E-4</Average_Cost_Variant_Cost>
    ...
    <Average_Process_Instance_Cost>1.3816812000000002E-4</Average_Process_Instance_Cost>
    <Activity_Cost>
        <Activity id="[id]">
            <Activity_Average_Cost_Variant_Cost id="[id]">0.0</Activity_Average_Cost_Variant_Cost>
            ...
            <Activity_Average_Cost>0.0</Activity_Average_Cost>
        </Activity>
        ...
    </Activity_Cost>

    <Activity_Instance_Cost>
        <Activity id="[id]">
            <Cost_Variant id="[id]">
                <activity_instance_cost ProcessInstance_IDs="1, 2, 4, 6, 7, 8, 9" count="7">0.0</activity_instance_cost>
            </Cost_Variant>
            ...
        </Activity>
        ...
    </Activity_Instance_Cost>
</Sustainability_Info>
```

Figure 4: Format of Individual Activity Cost in the **statistic.xml*

2.2 Visualisation

For visualizing the simulation output, a visualizer tool developed with React is proposed. The visualization tool leverages the UI component library from MUI Charts and is inspired by a front-end prototype viewable at Visily. This tool is engineered to run on localhost:3001 to avoid conflicts

with SimuBridge, which operates on localhost:3000. Our proposed enhancement involves integrating a "View Dashboard" button within the "Run Simulation" interface of SimuBridge, specifically after the "Start Simulation" process concludes and the "Simulator Feedback" section appears. Currently, this section offers file download capabilities for simulation results. The addition of the "View Dashboard" feature would provide a seamless link to the localhost:3001 page, where users could explore visualized data [Figure 5].

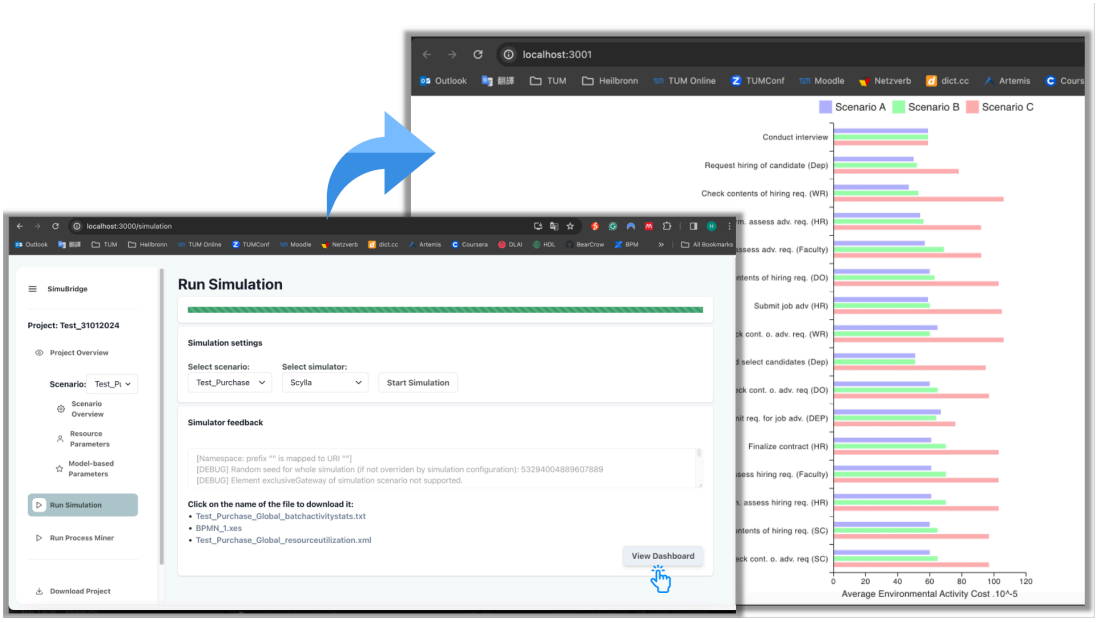

Figure 5: Proposal of Viewing Data Dashboard

2.3 Testing

In the development of the shipping logistics simulation with Scylla, an extensive and rigorous testing strategy was employed to ensure the reliability, performance, and accuracy of the system. This report outlines the testing methodologies implemented, focusing on the ExecutionLoggingPluginTest, GCParserTest, and SCParserTest classes. It also acknowledges the adaptation of ScyllaScripts.java, originally authored by Leon and modified to fit the project's specific requirements.

2.3.1 ExecutionLoggingPluginTest

The ExecutionLoggingPluginTest class extends SimulationTest, providing a robust framework for testing the execution logging plugin. Key test cases include:

- **testWriteToLog**: Ensures the cost_driver plugin accurately logs simulation details into XML and XES files, comparing generated logs with expected outputs to validate execution data accuracy. This test checks for logged events and cost metrics discrepancies, essential for reliable simulation analysis.
- **testCorrectFileExtensionXES**: Validates the correct file extension of the output log, ensuring compatibility with XES log standards.

- **testFindConcreteCaseByCost_Integration**: An integration test that verifies the system's ability to accurately identify and map concrete cost drivers to abstract cost drivers based on cost implications. These tests are crucial for verifying the integrity and accuracy of the logging mechanism, which plays a pivotal role in analyzing and optimizing simulation performance.

2.3.2 GCParserTest & SCParserTest

The GCParserTest focuses on testing the global configuration parser, critical for initializing the simulation environment. It includes an integration test: This test verifies the parser's ability to accurately extract and instantiate abstract cost drivers from the global configuration, ensuring the simulation's cost dynamics are correctly represented. This component's testing is essential for ensuring the simulation environment is correctly set up with the intended cost drivers and configurations.
The SCParserTest assesses the simulation configuration parser, ensuring the system accurately parses and applies simulation-specific settings. Key tests include:

- **TestParsing_CD**: Tests the parsing logic for cost drivers, validating the correct association between simulation elements and their respective cost drivers.
- **TestParsing_CV**: Validates the parsing and application of cost variant configurations, a critical aspect for simulating different cost scenarios within the logistics simulation. These tests ensure that the simulation configurations are correctly interpreted and applied, allowing for diverse simulation scenarios to be accurately modeled.

2.3.3 Acknowledgment of ScyllaScripts.java

The adaptation of ScyllaScripts.java from Leon's original work has been instrumental in extending the simulation's capabilities. The modifications were carefully designed to align with the project's specific needs, particularly in automating simulation runs and integrating custom configurations. This acknowledgment serves to recognize the foundational work by Leon and highlight the customization undertaken to tailor the script to our project's requirements.
The testing strategy implemented for the shipping logistics simulation with Scylla adheres to industry best practices, as evidenced by the structured and comprehensive testing classes. Through meticulous testing of execution logging, global configuration parsing, and simulation configuration parsing, the project ensures a high level of reliability and accuracy in simulating logistics operations. The customization of ScyllaScripts.java further demonstrates the project's commitment to leveraging existing resources while adapting them to meet specific simulation needs.

3 Documentation of Process

3.1 Organization of project

Team BearCrow would follow the agile development strategy Scrum. The Scrum team would have, one product manager, one Scrum master, and three developers. The roles would then be switched every week, allowing each person to at least take the role of the three mentioned. The product

manager would be responsible for defining the product backlog and deciding what the next steps are for the following week. The scrum master would then with the assistance of the product owner assign the tasks accordingly. The allocation of tasks was also assigned according to an individual's skill set. The scrum master would also be responsible for creating e a presentation each week for the customers to have a summary of what was achieved, what roadblocks were hit, and what the next possible steps are. The team would work on the tasks given, and communicate with the rest of the team regarding blockers or problems that occurred. If blockers or problems were hit, tasks would then be dynamically allocated to prevent loss of progress.
To help coordinate and keep everything organized we used two platforms to help keep track of the product log and who was responsible for what. Confluence was used to note the tasks with the corresponding assignee and with a brief description. Alongside this, Trello was also used to keep a nice visualization of the product backlog, with four different categories of tasks: Backlog, Committed tasks, Blockers, and Done.
The backlog would contain tasks which have been created but work on it has not begun, committed tasks are tasks which are currently being worked on, and blockers are tasks that have no current solution, and done are tasks which have been completed. For basic communication we used WhatsApp, and to hold our meetings and discussions, Zoom was used.
As a team, we would meet twice a week privately. Our week would rotate around the Thursday meeting. We would have a meeting on the following Friday, and the corresponding Product owner would have prepared the product backlog and would then assign the tasks accordingly. We would then have a second meeting on Wednesday, before the next customer meeting, to go over our progress and collect questions so the product owner could ask and present what has occurred throughout the development for that week. For each meeting we had with our customers (Leon and Finn), we would present our results of the previous week and discuss the goals for the following week. While presenting the results, Finn or Leon would provide insightful advice or comments to help guide our possible next steps. We would also ask questions that were collected from the prior week. Leon and Finn would act as both a customer and a technical supervisor. A large part of our source of information would come from the following documents provided by the team responsible for the Bachelor practical course. The documents are the Background papers of Scylla, Simubridge, LCA, and SOPA. Our main understanding of implementing the sustainability plugin stems from the SOPA paper.

3.2 What insights we gained from working in and on such projects

To emulate a real-life development project, it was important that the guidance of the mentors was controlled/minimized and that customers would provide vague requirements. After a brief adjustment period to navigating broadly defined requirements, we were able to self-organize and define concrete goals. We learned to ask the right questions to the customers to collect the necessary information to proceed. We realized that tasks should be assigned to those who can do the task the best.
Being able to absorb the answers given and to be able to create a product backlog which can then be worked on was another skill that was picked up for the whole team. With this, we would have

to then choose what platforms would be useful for our requirements, in our case, we ended up using Confluence and Trello to help us. Not only do we have to ask our customers questions, but we would also then have to learn how to present our results in such a manner, that any stakeholder with any background which might not have a background or understanding of computer science, would then be able to understand what is being presented. We would also have to learn how to communicate with another developer team, team Anoa, to provide us with information when needed.

3.2.1 Retrospective Meetings

The retrospective meeting in the sprint aimed to improve our progress by reflecting on our strengths and weaknesses. One of the main issues we identified was the task allocation process. We decided to assign the same person to the topics they had previous experience with, whenever possible. This would reduce the learning curve and increase the efficiency of task completion. However, if the person was unavailable or unfamiliar with the topic, we provided them with a knowledge hub on the confluence page. The knowledge hub contained the background research and the relevant information from the previous tasks. This would help them to understand the topic better and avoid duplication of work. Additionally, our WhatsApp group plays an essential role here to facilitate communication and collaboration among the team members. The group served as a forum where anyone could ask questions, share ideas, or seek feedback.

3.2.2 Design Implementation

Besides some issues with managing the progress, we also faced challenges in designing and constructing the service and structure of the cost drivers. For example, in the process of deciding how to incorporate the sustainability information into the three input files for Scylla, we had to choose where to put the cost variant configuration and the detailed information of each cost driver, as well as their formats. We had a lot of flexibility in this part, but we also needed to discuss it with Team A to finalize it. However, this discussion was very awkward, because Team A's confirmation of the format was at the end, after they had studied the LCA database and what data they could extract. Thus, it takes time and could only be confirmed near the second half of progress when finishing studying. But for us, this was something we needed from the beginning, and without confirming the format, we could not proceed with the rest of the work. Therefore, in the situation where Team A could not tell us what they could give us, and we did not know what we needed, what and the configuration of those data as well as where to put them, how to mock input files in a specific format in assumption was the biggest decision-making problem we faced in this project.
Another decision point concerns the output format of the global statistic file. We are not sure which output format would be better for the visualization platform that we developed using SimuBridge and JavaScript. However, since the native output file for the global event log data is written in XML, and our plugin's main programming language, Java, is more comfortable with XML, we opted for the XML file format.

4 Limitations and Future Work

References

[1] Finn Klessascheck, Ingo Weber, and Luise Pufahl. "SOPA: A Framework for Sustainability-Oriented Process Analysis and Re-design in Business Process Management". unpublished.

[2] John Glasson and Riki Therivel. *Introduction To Environmental Impact Assessment*. 5th ed. Routledge, 2019. URL: https://doi.org/10.4324/9780429470738.

[3] Tsun Yin Wong et al. *Scylla - An extensible BPMN process simulator*. 2024. URL: https://github.com/bptlab/scylla.

[4] Tsun Yin Wong et al. *Scylla - Plugin Concept*. 2019. URL: https://github.com/bptlab/scylla/wiki/Plugin-Concept.