Creating and Managing of Simulation Models for the Simulation of Sustainability in Business Processes

Final Team Anoa report for the Bachelor Practical Course

First Author, Second Author, Third Author, Fourth Author, and Fifth Author

TUM School of Computation, Information and Technology, Technical University of Munich

February 10, 2024

This report details the outcomes of the bachelor practical course for the Information Engineering B.Sc. program, focusing on creating and managing simulation models for business processes simulation using environmental data. The project's core involved adapting the existing SimuBridge project[BKN+23][PWW18], employing Scylla as the simulation engine, and utilizing the OpenLCA database for environmental costs. We aimed to simulate business processes with an integrated environmental impact assessment, promoting sustainable business practices. A separate team undertook the implementation of the actual plugin for processing environmental data in Scylla.

Extensive research and learning were crucial to this course, including Web Development with React, responsive design using Chakra UI, and integration with Scylla through Scylla API, which uses the Python-based framework Flask. We also explored Scylla for process simulation and OpenLCA for environmental data.

This report comprehensively covers the problem statement, tasks undertaken, our solution approach, and a description of the final product. Additionally, it outlines the project timeline and organizational aspects.

1 Introduction

1.1 Sustainability in Business Processes

Simulating business processes is becoming increasingly important for companies. It allows them to test out different ways of working without risking real-world consequences. This means they can find the most efficient ways to operate. Adding environmental data into these simulations is now essential. As the world focuses more on being environmentally friendly, companies need to understand how their actions affect the environment. By using environmental data in their simulations, businesses can see the impact they have on nature. This helps them make better choices that are good for both the company and the environment. In short, combining business simulations with environmental data is an important move for any company that wants to be efficient and responsible.

1.2 SimuBridge

SimuBridge is a web-based application, the core application of this project; it functions as a bridge between business process mining and parameter-based simulation. The current version includes SimodConverter, the connector for **Simod**, a process miner that can output scenarios and process BPMN based on the process log. As for simulation, Simubridge offers ScyllaConverter, a connector for **Scylla**, an extensible business process simulator. This web application uses **React** and **Chakra UI** as its technological stack. Currently, SimuBridge does not implement a modular plugin-based architecture; however, this platform has a huge potential to be a bridge between different systems for handling tasks related to business processes. In particular, as a part of this project, we are targeting adding an integration with OpenLCA, described below.

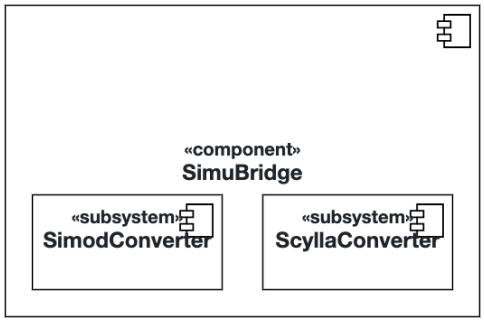

Figure 1 SimuBrige, a bridge between process mining and simulation

//TODO: Add info about internal files and structure ???

1.3 Scylla

Scylla is an open-source, MIT-License, BPMN process simulator. The application is written in Java and thus requires JRE/JDK to run. It can be run both with GUI, based on Java AWT (Abstract Window Toolkit), or in headless mode. In our setup, Scylla is run headless by ScyllaApi.py python3 Flask file exposing REST API to external applications, including SimuBridge. For simulation, Scylla requires the following data:

- BPMN model of the business process
- XML-based Global Configuration file containing general information needed for simulation, including resources, schedules, etc.
- XML-based Simulation Configuration file containing process-specific information
- When calling ScyllaApi, the API consumer also must provide *requestId* as a header

As the result of the simulation, Scylla provides the following files:

- XES-based execution log
- XML-based resource utilization file
- Additional .txt file *requestId* as a header

Additionally, Scylla supports plugins and provides the corresponding interface and comprehensive documentation.

1.4 OpenLCA

1.4.1 OpenLCA - A Comprehensive Life Cycle Assessment Tool

OpenLCA is a leading software for life cycle assessment (LCA) and sustainability analysis. It is designed to help businesses and researchers evaluate the environmental impacts of their products and services throughout their entire life cycle, from raw material extraction to disposal.

1.4.2 Data Storage and Types

OpenLCA stores a wide range of environmental data that is critical for comprehensive LCA studies. This includes, but is not limited to, information on raw materials, energy usage, emissions, waste generation, and other environmental impacts associated with different stages of a product's life cycle. The software supports various types of data, such as ecoinvent databases, which are known for their extensive and detailed life cycle inventory datasets.

1.4.3 API and IPC Server

One of the key features of OpenLCA is its Application Programming Interface (API) and Inter-Process Communication (IPC) server. The API allows for the integration of OpenLCA with other software tools and systems, enabling automated workflows and data exchange.

The IPC server in OpenLCA facilitates communication between the OpenLCA software and other applications. This means that users can remotely control OpenLCA and access its functionalities programmatically, allowing for more advanced and customized LCA analyses.

1.4.4 Methods and Functionality

OpenLCA offers a range of methods for environmental impact assessment, including carbon footprinting, water footprinting, and more comprehensive methods like ReCiPe and CML. These methods enable users to quantify and compare the environmental impacts of products and processes, supporting informed decision-making towards sustainability.

1.5 Terminology

From now on, we will name all imported Product Systems from the OpenLCA database as **concrete cost drivers**. The category of the Product System will be **abstract cost driver** as it is an impact cost driver without concretization. Fig. 2 shows the example of abstract and concrete cost drivers.

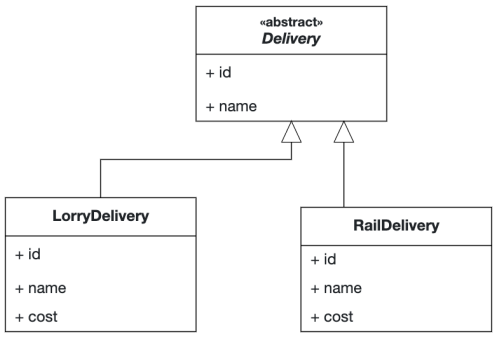

Figure 2 Abstract and concrete cost drivers

Variant is a set of mappings between abstract and concrete drivers to save different combinations of concrete cost drivers for the same set of impact categories.

1.6 End Task

Taking into account all the systems and statements described above, our end goals were:

- Extract all Product Systems from the OpenLCA Database using its API as abstract and concrete cost drivers
- Calculate unit-less weighted and normalized concrete cost drivers
- Provide the mapping between abstract cost drivers (i.e., impact categories) and business process activities
- Allow to create, save, edit, and delete variants
- Modify the existing module that is responsible for the integration with Scylla to ensure the data mentioned above is passed to the simulation engine
- Develop the UI for the problems above in the existing SimuBridge web application

2 Approach

2.1 Organisational approach

When organizing our work, we worked using **Agile** and **Scrum** approaches. The length of each iteration was 1 week. At the end of each iteration, we had a meeting with our project supervisors to report the progress (them acting as *customers*), define the tasks for the next iteration, and get support (for the last two activities, they acted as *tutors* and *mentors*).

In order to track the issues, their status, iterations, assignees, workload estimations, we chose **GitHub Projects** which is a part of GitHub. We created and used the Kanban board to be more efficient in working with the tasks and iterations.

For communication and video calls, we used enterprise TUM **Matrix** server and **Element**, an open-source software instant messaging client implementing the Matrix protocol. Additionally, we used other instant messaging applications for internal communication.

2.2 Technical approach

We used **GitHub** git repository, which was added to **INSM-TUM-Teaching** GitHub organization for source control. As this is not a fullscale project in the industry, we have not defined a formalized git flow, however, used user-based branching. For Web development, we used **React** to build components and **Chakra UI** as a component library. The choice of this stack was set by the technologies used in SimuBridge. All applications except for OpenLCA were assembled in Docker containers. Additionally, we had **python** environment installed in order to run Scylla API, which is written on Flask.

Application	Role	Port	Docker
SimuBridge	Core application	3000	+
Scylla API	Simulation	8080	+
Simod	Process Miner	8880	+
OpenLCA	LCA Software	8081	-

Table 1 Applications

3 Project Results

3.1 Modeling the data flow between the systems

During our investigation phase, we identified the data flow between the systems described above.

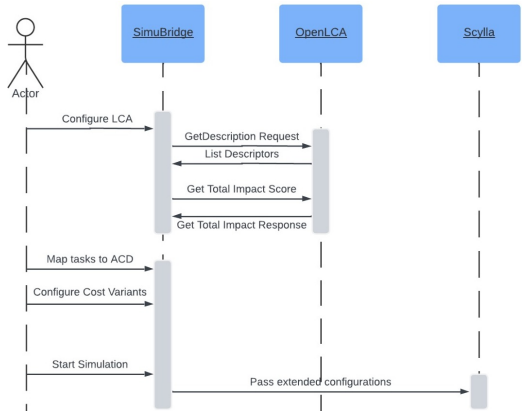

Figure 3 Sequence Diagram

In this diagram and description below, we omit process mining operations as they are not a part of our project. From now on, we assume that the user used SimuBridge to mine the process using Simod and has a configured project in SimuBridge.

1. The user uses the SimuBridge web application to configure LCA data
2. SimuBridge sends requests to OpenLCA to retrieve all cost drivers and calculate concrete cost drivers, discussed further in subsection 3.2

3. Having all abstract cost drivers, the user may execute the mapping between business activities and abstract cost drivers and configure variants

4. The user may start the simulation using the *Simulation* page

5. Three files are passed to the process simulator:
   a) BPMN model
   b) Global Configuration File
   c) Simulation Configuration File

6. The process simulator returns simulation results

As our target goal, taking that we pass correct environmental data as a part of global and simulation configuration files, we expect the process simulator to return the file containing the sustainability simulation results in addition to the files described in subsection 1.3. As mentioned before, the Scylla plugin to provide this output is developed by a separate team.

To work with data and follow its flow, we primarily used the data provided by our tutors:

- Environmental data-set for OpenLCA
- Example event logs available in the SimuBridge GitHub repository
- Our own manually created sample business process and SimuBridge project

3.2 Retrieving data from OpenLCA

We conducted intensive research on OpenLCA. The interface of the application allows to enable a REST HTTP Server to handle API requests (*Tools -> Developers yools -> IPC Server*). By default, the app suggests using port **8080**, however, for our test environment, we chose **8081** as Scylla API already uses the first port in the default SimuBridge setup. Additionally, we specified a requirement to configure the OpenLCA host and port using the web application to make this flexible. Our solution has two steps of OpenLCA usage to generate abstract and concrete drivers in SimuBridge; these steps are described below. For executing API queries, we used **olca-ipc** npm package, a TypeScript client for the openLCA IPC API.

3.2.1 Get all Product Systems

The first step is receiving all product systems, which will be later transformed into abstract and concrete cost drivers.

```
const client = new o.IpcClient.on(
   apiUrl);
const systems = await client.
   getDescriptors(o.RefType.
   ProductSystem);
```

Listing 1 Getting all product systems

This request would invoke *data/get/all* API request for the type of **ProductSystem**. Each Product System represents an object that includes all impacts associated with this descriptor (e.g., fresh water, CO_2, etc,...)

3.2.2 Calculate concrete cost drivers

After we have the list of all Product Systems, we get normalized and weighted values using the normalization method of EF3.0 method.

```
const impactMethod = await client.
   get(
o.RefType.ImpactMethod,
{ id: impactMethodId, refType: o.
   RefType.ImpactMethod })
// retrieving impact method by its
   id
let e = impactMethod.nwSets //
   normalization sets

let calcSetup = await o.
   CalculationSetup.of({
  target: el,
  impactMethod: impactMethod,
  nwSet: e[0], //default
     normalization set
  allocation: o.AllocationType.
     USE_DEFAULT_ALLOCATION,
  withCosts: false,
  withRegionalization: false
});

const result = await client?.
   calculate(calcSetup);
const impactSum = weights.map(i =>
   i.amount || 0).reduce((sum,
   current) => sum + current, 0);
```

Listing 2 Calculating concrete cost drivers

The variable *impactSum* is calculated as the sum of the *amount* values (i.e., amount of normalized impact) of each impact from each object in the *weights* array of impacts for every product system.

impactSum = $\sum_{i=1}^{n} a_i$

For each element in the array, if the *amount* is not specified or is equal to zero, it is treated as zero in the summation. This sum represents the cumulative impact, weighted and normalized. Later, these results are transformed into an array of objects according to the following data schema.

```
{
"name" : "AbstractCostDriver",
"properties": [
 { "name": "id", "type": "String"},
 { "name": "name", "type": "String
    "},
 { "name": "concreteCostDrivers", "
    type": "ConcreteCostDriver",
    isMany : true },]
},
{
"name" : "ConcreteCostDriver",
"properties": [
 { "name": "id", "type": "String"},
 { "name": "name", "type": "String
    "},
 { "name": "cost", "type": "Real"},
]},
```

Each abstract cost driver contains an array of all its concrete cost drivers.

3.3 Generating models for Scylla

3.4 Modifying web components of SimuBridge

4 Discussion and Prospects

4.1 Limitations

4.2 Further Extensions

One of the steps is to ship OpenLCA as a docker container to enable the hosting of all applications of the process in the Docker Engine.

4.3 Team Reflection

5 Summary

References

[BKN+23] Leon Bein, Finn Klessascheck, Sviatlana Nepeina, Christian Warmuth, Timotheus Kampik, and Luise Pufahl. Simubridge: Discovery and management of process simulation scenarios. 09 2023.

[PWW18] Luise Pufahl, Tsun Wong, and Mathias Weske. *Design of an Extensible BPMN Process Simulator*, pages 782–795. 01 2018.