university of groningen

faculty of science and engineering

Implementing Sustainability of Workflows into a Conventional BPMN Workflow Engine

Bachelor's Project Computing Science

August 2025

Author: Josefine Eberhardt Sondergaard Rasmussen
Student Number: S5200725
First supervisor: Dimka Karastoyanova
Second supervisor: Michel Medema

Abstract

Workflow management systems (WfMSs) are one of the main ways businesses monitor and analyze their processes. The businesses use the WfMSs' key performance indicators (KPIs) in order to decide if the process can be improved in any of the four standard KPIs measured. Sustainability in business processes is a growing concern for many businesses. By implementing current Green BPM research surrounding modeling and calculating sustainability in workflows into a workflow engine, businesses would be able to monitor and react to key environmental indicators (KEIs) thus making informed decisions on the environmental sustainability of their processes easily and quickly compared to before. The selected BP language is BPMN4ES and the selected workflow engine is Camunda 7.

Contents

List of Figures

List of Tables

1 Introduction

An emerging priority for modern businesses is sustainability. Initiatives to be more sustainable like the UN Sustainable Development Goals [9] and pressure from the public lead businesses to have sustainability in the front of their mind when creating their processes. In the current system of Workflow Management Systems (WfMSs) there is no way to monitor a process' impact on sustainability, leading businesses to have to go through lengthy means to get an overview of their sustainability. This leads us to our main research question: How can a workflow engine be extended to allow businesses using it to view their sustainability metrics while adhering to the WfMC-reference model?
Workflow and process management is an important part of any business. It improves efficiency, allows for qualitative analysis of a business's workflow, and helps to locate and eliminate bottlenecks. The management is achieved through a workflow management system (WfMS). The workflows and processes are modeled using a business process (BP) modeling language, such as BPMN (business process modeling notation), and then deployed using a workflow engine like Camunda 7. A business is able to monitor key performance indicators (KPIs), such as quality, cost, time, and flexibility, through the workflow engine and change the workflow to prioritize specified KPIs decided by the company [11]. The WfMS reference model developed by the Workflow Management Coalition (WfMC), shown in Figure 1, is the industry standard for WfMS implementations.

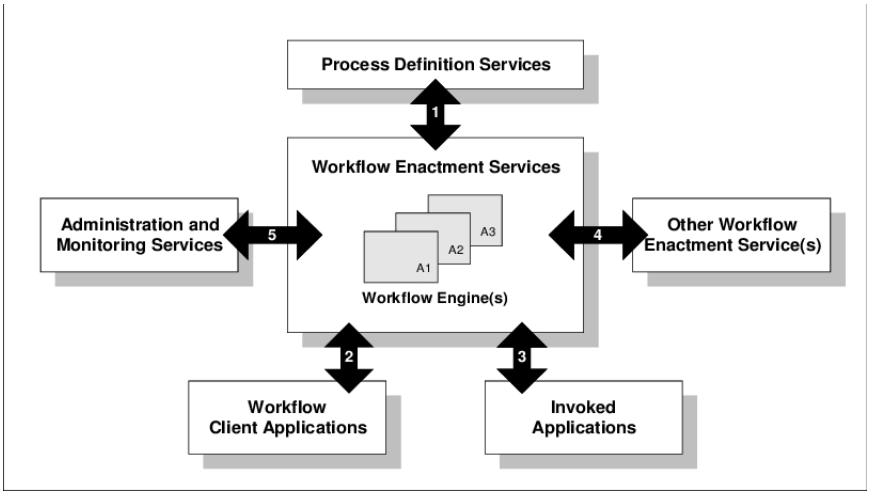

Figure 1: The WfMS-reference model developed by the WfMC [3].

Sustainability is a growing concern for businesses, both due to societal attention and legislative guidelines. There has been an increase in protests and boycotts against companies perceived as non-sustainable, like the Amazon Employees Climate Walkout. Guidelines like the UN Sustainable Development Goals [9] have led to legislation forcing companies to focus on their sustainability, such as the Corporate Sustainability Reporting Directive (CSRD), a law which states large companies must report on their environmental impacts. Green Business Process Management (Green BPM), a paradigm focusing on including environmental objectives into business process development and management, has been developed in order

to attempt to address these concerns [1]. Along with this paradigm, integrating sustainability into the BPM life cycle shown in Figure 2 would allow for organic sustainable BPM development [5].

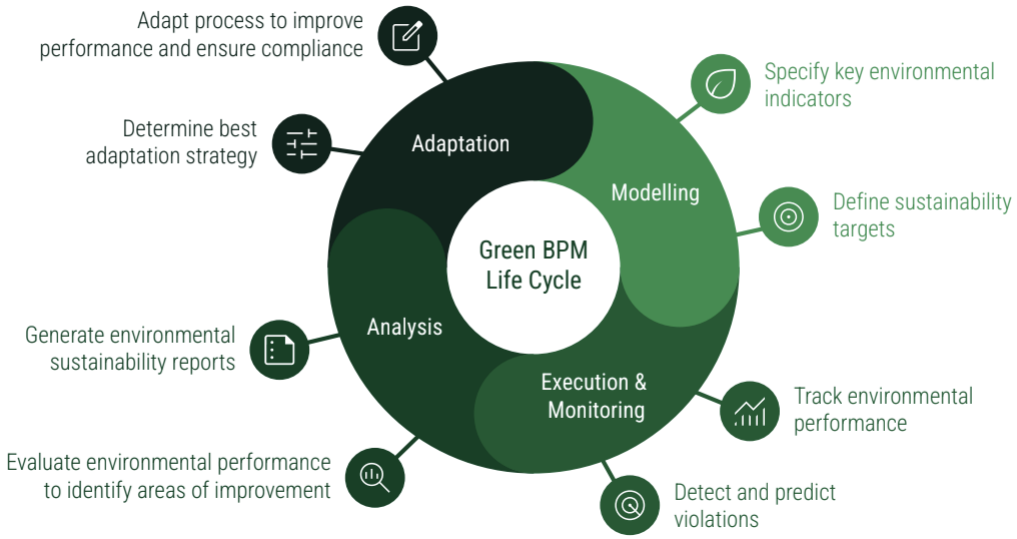

Figure 2: The phases of the Green BPM life cycle along with the activities related to environmental sustainability performed in each of these phases [5].

Developing and implementing your business process model while keeping certain sustainability metrics in check as a KPI is seen as a clear solution to help businesses better monitor their sustainability. The current selection of quality, cost, time, and flexibility as KPIs is not sufficient, and thus both the modeling language and the workflow engine should be capable of also tracking and reacting to key environmental indicators (KEIs), which can be seen in Figure 3. This has been an ongoing research topic for many years [4].
Notable green BPM research includes an extension to the BPMN modeling language that allows modeling of KEIs referred to as BPMN4ES, and a calculation service that calculates the co2 emissions of a process. The modeling of KEIs combined with the calculation of a KEI, such as co2 emissions allows an engine to track the KEIs of a process instance. These two technologies will be explained in further depth in the related works section. Currently, there is no workflow engine that integrates both of these technologies together to allow for tracking and adapting of BP's using KEIs as guidelines. Implementing this would allow for the BPM life cycle to be integrated with sustainability metrics. The modeling phase will be able to include modeling of the sustainable KEI metrics. Sustainability will be included in the monitoring, analysis, and adaptation phase by being able to monitor, analyze, and react to the individual tasks' KEI metrics when compared to their target values.

1.1 Proposal

This thesis aims to extend the Camunda 7 engine to be able to model processes using the

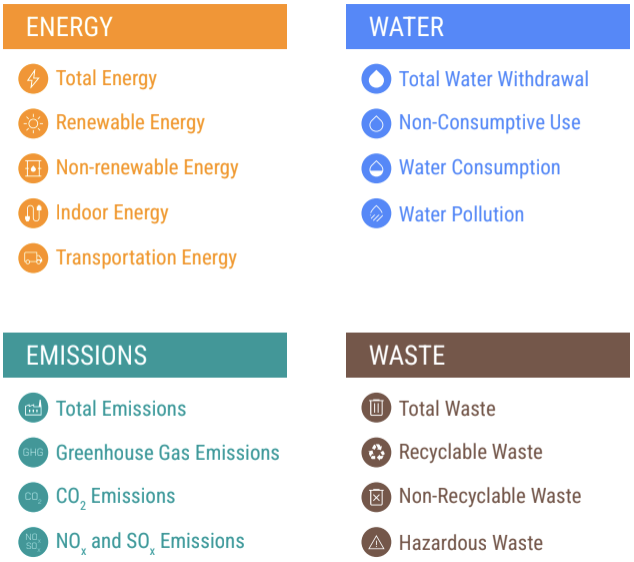

Figure 3: The KEIs obtained from RUG lecture slides based on Van den Broek [10].

BPMN4ES extension, integrate with the environmental sustainability calculator service to get real-time KEI calculations, and store the value of the KEIs in the engine to use for future decisions. This would allow for businesses to track their environmental impact easier and help automate environmental decision-making. The following is our main question as obtained by the introduction: How can a workflow engine be extended to allow businesses using it to view their sustainability metrics while adhering to the WfMC-reference model? This question can be further broken down into sub questions:

1. How can the Camunda 7 engine be extended to be compatible with BPMN4ES while adhering to the WfMC-reference model?
2. How can the environmental sustainability calculator service created by Popescu be connected to the Camunda 7 engine while adhering to the WfMC-reference model?

The answers to these research questions and the resulting code extension are expected to provide valuable contributions, since the engine extension does not yet exist. By implementing the engine extension, insight is provided into extending workflow engines to track and monitor future emerging metrics. Implementing the engine also allows businesses to keep their technology stack more concise, thus improving efficiency in on boarding of new employees and speeding up analysis of their system since all metrics (KPIs and KEIs) can be monitored in one place. In addition to a smaller technology stack, analysis is also expedited by the KEIs being available immediately after changes have been made to the business model since calculating them externally takes longer time.

1.2 Motivation

Sustainability in WfMSs has been an ongoing research topic in the information systems area of computing science for many years [4]. The paradigm that has been created with the goal of making business process management more sustainable is Green BPM [1]. The paradigm extends traditional BPM by including environmental objectives in business process management and design. Some of the current developments are in regards to sustainable BPM modeling, calculating sustainability metrics in business processes, and extending the BPM life cycle to integrate environmental sustainability. Many developments have been made; however, no fully functional workflow engine has been implemented. By extending an existing workflow engine to also be able to monitor sustainability metrics, businesses would be able to use their current models and deployments with a couple additions and monitor both their KPIs and sustainability in one place. This would allow for sustainability to be integrated into the BPM life cycle. The modeling phase would be able to include modeling of sustainability metrics, and the monitoring, analysis, and adaptation phases would all be guided by the sustainability metrics measured. Extending the workflow engine also allows for faster sustainable decision-making since by changing the business process model, the new sustainability metrics are automatically calculated and available immediately. The thesis is expected to not only provide an extended workflow engine capable of modeling and monitoring sustainability metrics, but also provide valuable insight of how the extended engine can be further extended in the future to monitor new emerging metrics.

1.3 Paper Structure

The structure of the paper aims to concisely explain the background, aims, implementation and results of the project. The following outlines the order and purpose of each section:

Introduction: The introduction introduces the terminology, basic theory, and the current state of the resource domain, as well as proposing and motivating the project as a whole. Concepts covered include WfMS, KPIs, KEIs, and the centers the explanations and concepts around the need for sustainability.

Related Work: The related works section functions to go into further deeper detail on the current state of the resource domain in regards to the proposal. It further explains the theory or resource results mentioned in the introduction and gives further context to missing areas of resource. Research explained includes BPMN4ES, previously extended modelers and KEI calculators.

Requirements: The requirements section sets up the main functional and non functional requirements of the engine extension.

Materials and Methods: The materials and methods section sets up and justifies the primary technology and tools the project will utilize. It helps to introduce the basic approach the project will take towards implementing the workflow engine extension. It puts emphasis on rationalizing the choices of the packages and libraries used throughout the project and why Camunda 7 was chosen as the engine to extend.

Implementation: The implementation section explains in depth how the extension was implemented and rationalizes any design decisions made along the way. The main sections explain the architecture, how BPMN4ES was connected to the engine, how the emissions of

the tasks were calculated and how the emissions were stored for the engine to later use.

Results: The results sections goes through the listed requirements of the project and states whether they were met or not. It also discusses limitations and assumptions the requirements have.

Conclusion: The conclusion wraps up the project by concisely summarizing what was achieved and any other results or comments about possible future work, such as adding additional KEI types, improving the accuracy of calculations, which additional modeling concepts are possible and reporting the KEI values after the process instance has completed.

2 Related Work

2.1 Modeling Languages

An extension to BPMN 2.0, BPMN4ES, has been developed in order to model KEIs by allowing visual and machine readable modeling. The extension allows users to specify both the type of KEI they want to model and a numerical target along with a unit [6][5]. This allows for easy modeling of KEIs and sets up the layers needed for adaptability by allowing a target value to be set. The following is an example of a diagram in BPMN4ES created using the modeler:

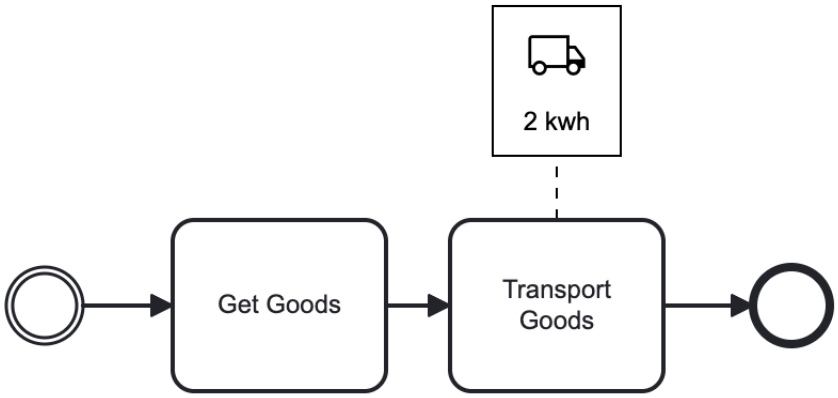

Figure 4: A BPMN4ES example diagram modeled in the modeler provided by [5].

The following is an exert of the associated .bpmn file the modeler creates:

```
<bpmn2:task id="Activity_12x0afq" name="Transport Goods">
  <bpmn2:extensionElements>
    <bpmn4es:environmentalIndicators>
      <bpmn4es:keyEnvironmentalIndicator id="transportation-energy"
         unit="kwh" targetValue="2" icon="local_shipping" />
    </bpmn4es:environmentalIndicators>
  </bpmn2:extensionElements>
  <bpmn2:incoming>Flow_0mu30l4</bpmn2:incoming>
  <bpmn2:outgoing>Flow_0lf0miq</bpmn2:outgoing>
</bpmn2:task>
```

Listing 1: Sample BPMN XML

The ”id” refers to the type of KEI which should be tracked, the targetValue refers to the maximum value the task can consume, the unit refers to the unit associated with the targetValue and the icon refers to the icon displayed in the BPMN4ES modeler.

Another modeling notation extension created with sustainability in mind is the one created by Recker [8]. This is an extended version of BPMN with notation elements specifically focusing on calculating a process's carbon footprint. Like the BPMN4ES notation, this notation also allows the ability to set up targets for specific activities. The notation is also designed so the carbon footprint of individual activities, sub-processes and the entire process to be viewed in the model at a glance. Figure 5 shows an example diagram using the extended notation elements.

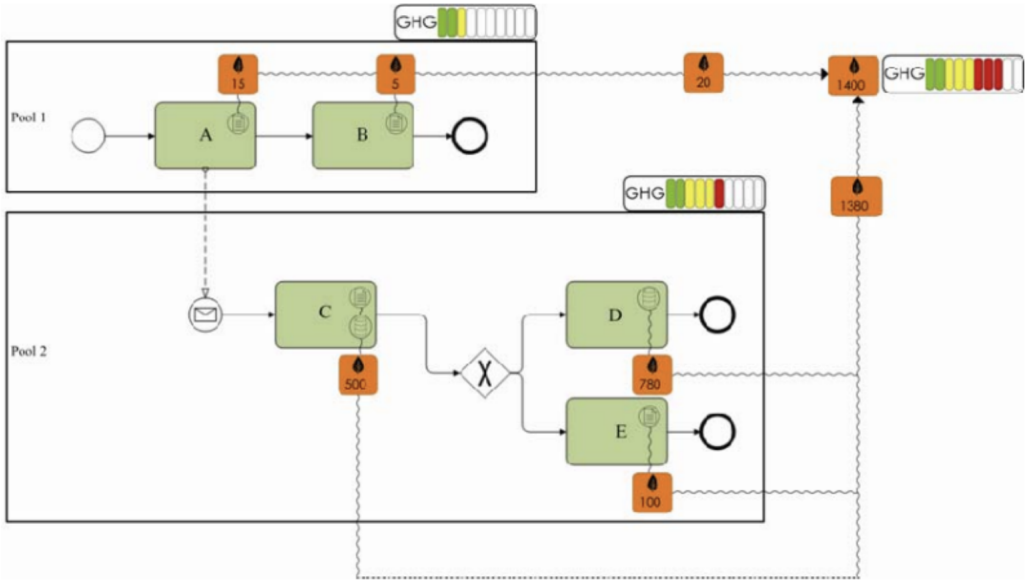

Figure 5: Sample extended BPMN Model including Carbon Footprint Notation Elements, provided by [8].

2.1.1 Gaps In Current Works

The BPMN4ES modeler is currently standalone and does not associate with any specific engine, so if one would want to use the BPMN code resulting from the modeler, one would first have to tweak it in order to make it compatible with the chosen engine.

While the notation created by Recker [8] is nicely set up for calculating emissions, it limits itself to only that KEI. The current version of the extension only has emission specific notation and would need to be extended in order to be useful in a larger sustainability context.

Overall, the current work in modeling languages provides a good base for sustainability of workflows however none of them have been implemented with any concrete BPMN engine.

2.2 KEI Calculators

KEI calculators have also been developed throughout the years to combat climate change. One such calculator is an environmental sustainability calculator service for business pro-

cesses which calculates the carbon dioxide and greenhouse gas emissions of a process [7]. While other sustainability calculators exist, this is one of the only ones that allows easy interaction between it and workflow engines. This allows compatible workflow engines (Camunda 7 and JBPM) to connect with the service through HTTP requests and the service is then able to calculate the emissions as the process instance completes [7]. After the process instance is finished and the end event has been triggered, the service is able to provide a CO2 report of the instance based on start times, end times, resources used, fuel types and emission factors. Most of these values have to be provided through annotations in the service's front end, which is also where the final report is located. Although this service is still in its infancy, it provides good infrastructure for extendability and is open for further development both vertically and horizontally.

Recker also developed a calculation to go along with the extended BPMN notation he created. This calculation is based on the activity based cost (ABC) approach which utilizes a driving cost for the calculation. Recker transforms this concept into activity based emission (ABE) analysis, using emission drivers instead of cost drivers. The basic steps of the analysis are as follows: 1. Identify the product or service process to be considered. 2. Determine all the resources and processes that are required to create the product or deliver the service, and their respective CO2 accumulation. 3. Determine the "emission drivers" for each resource. 4. Calculate CO2 emission for each activity by gathering Activity Data for each process and resource and define the emission type for each Activity Data. 5. Use the data to calculate the overall CO2 emission of the process. The calculation has not been implemented into a usable calculation service yet.

2.2.1 Gaps In Current Works

One clear gap in the current work for KEI calculators is the lack of different types of KEI calculators. The main focus on emission calculations results in other KEIs' calculators being inaccurate due to the lack of research or incomplete due to lack of availability.

While the calculator service created by Popescu [7] is able to connect to engines, it is a one-way relationship and the engine itself gets no information about its KEIs. This severely limits it from being able to handle future additions to BPMN4ES, since the engine is not able to access any information regarding its emissions and makes KEI based decision gates or boundary events practically impossible.

Overall, while the KEI calculators created serve as a good base for calculating accurate KEI metrics, they limit themselves by mainly focusing on emission calculations. The calculators have also not been fully integrated into an engine making them difficult to use in practice.

2.3 Literature Search

The literature search was completed using web of science to search for "sustainability", "workflow", "business process", "KPI", and "adapt*". It was also based on the literature provided by D. Karastoyanov and M. Medema, as well as snowballing.

3 Requirements

The main requirement of the project is to extend a BPMN workflow engine to be able to handle sustainability; however, within this, there are several other important requirements. The requirements will be split into two types, functional and non-functional. Functional requirements are requirements which specify the actions a product must perform and outlines its features and functionalities. Non-functional requirements, on the other hand, detail the system's overall attributes and qualities. Each requirement will have an associated ID for later reference which will follow the following naming scheme: An R notating requirement, an NF or F notating if the requirement is functional or non-functional, and a number associated with which number requirement it is. While all requirements are important, the requirements will be ordered from most to least important.

3.1 Functional Requirements

- **RF1**: The extension must use the BPMN4ES modeling language.
- **RF2**: The extension must extend a conventional and widely used BPMN workflow engine.
- **RF3**: The extension must be fully contained and usable within the chosen engine platform with no external tools or setup.
- **RF4**: The extension must calculate the emissions of a monitored task in real time as the process instance is running.
- **RF5**: The extension must return the final emission value of the task to the engine to be used for future decisions.
- **RF6**: The extension must accurately calculate GHG emissions per process instance, on the level of tasks.

3.2 Non-Functional Requirements

- **RNF1**: The extension must be designed to easily allow additional KEIs to be monitored and calculated.
- **RNF2**: The code must be well documented and easy to understand.
- **RNF3**: The extension should be able to run on multiple software systems.

4 Materials and Methods

The section highlights and rationalizes the choices in technologies as well as providing additional information of the methods used throughout the creation of the extension.

4.1 Technology Stack

Many technologies are used in order to realize the extension. The most important of these technologies are highlighted in this section.

4.1.1 Workflow Engine

The chosen workflow engine is the Camunda 7 engine. This engine was chosen due to its wide use in businesses as well as the abundance of resources related to it. While Camunda 7 will only be updated with maintenance updates from October 2025 and will instead be replaced with Camunda 8, Camunda 7 was chosen due to its easy set up and due to the internal working of its listeners. The code to the Camunda 7 engine is open source and available on GitHub [2].

4.1.2 Programming Language

In order to extend the Camunda 7 engine, the development will need to continue in Java since that is the original language the engine is written in.

4.1.3 Packages and Libraries

The Camunda library was utilized throughout the whole project in order to seemlessly extend the Camunda 7 engine. The uses include the org.camunda.bpm.engine, org.camunda.bpm.model, and org.camunda.bpm.spring.boot packages.
Spring Boot was used to run the Camunda 7 engine in an application. Spring Boot was chosen for its easy setup and straightforward development. Camunda also provides a Camunda 7 Spring Boot Application set up in its starter annotations, which further helped the initial setup of the project. h2 was also used as a database in order to run the Camunda 7 application. This database was used to store any information needed by the engine such as process instances, process definitions, and history information.
Lombok was used throughout the project for its ability to easily define standard setters and getters for a class without populating the file with unnecessary code, causing clean and understandable files. This allows future developers to more easily understand and further develop the project. Lombok was especially useful in the data storing classes of the extension.
Other than the above dependencies, the project only uses standard Java packages. During the development Maven was used in order to manage the dependencies.

4.2 Documentation and Version Control

4.2.1 Version Control

During the code implementation, git was used for source-code management. To avoid any serious loss of code due to material failure, the code was backed up to a private GitHub repository. The link to the repository is here: https://github.com/JJRas/Camunda7_Sustainability.

4.2.2 Documentation

The final code with be fully documented using Javadocs which will allow other developers to efficiently understand the code. The Javadocs will be at the class, method, and field level; however, the field level will only be for important variables in an attempt to keep clutter to a minimum. A very general explanation of the project and set up will also be available in the readme file of the GitHub repository.

4.2.3 Running the Application

In order to run the application simply do the following steps. Clone the GitHub repository, add your correct BPMN diagrams and run the main class (Camunda7Application). Once the application is running, it can be used through localhost:8080.

5 Implementation

This section focuses on the implementation of the extended workflow engine. Each decision will be rationalized and related back to the requirements in order to ensure all requirements are reached. The section is split into four main parts: the overall architecture of the project, connecting the engine to the BPMN4ES language, calculating the emissions, and storing the emissions in the engine.

5.1 Architecture

The overall architecture of the extension is very simple. The architecture will be explained through its conceptual design and its class diagrams as well as sequence diagrams.

5.1.1 Conceptual Design

The conceptual design of the engine extension is simple and consists of three parts, the engine, listeners and the polling system. The engine deals with executing the tasks as normal and triggers the listeners. The listeners then process the BPMN4ES attributes and determines the final emission calculation by gathering the final duration of the task. While any task is ongoing the polling system takes the tasks ongoing duration and calculates the ongoing emission value and stores it into the engine. Further detail into each part can be viewed in the later subsections.

5.1.2 Class Diagrams

Figure 6 shows the ULM class diagrams of the project. The class diagram shows how the different classes are related through dependencies and gives an overview of the internals of the classes. The following is a brief explanation of the classes. Functional classes are defined as the classes which contain the logic of the application whereas model classes are classes which are primary used for data storage.

Functional Classes: The following are the explanations of the functional classes of the project.

Camunda7Application: This class is the main class of the application. Once it runs it starts the camunda 7 platform up and runs it via a Spring Boot application running on localhost 8080.

UniversalTaskMonitor: This class extends the ExecutionListerner and TaskListener interfaces in order to listen for the custom BPMN4ES modeling attributes. The class both extracts the information stored in the attributes, retrieves the task duration and calls the EmissionCalculator in order the calculate and store the final task emission.

EmissionCalculator: This class calculates the emissions of a task based on the FinishedTask model. The class is a singleton and contains a method "calculateEmission" which is able to be called from other classes and uses private methods to calculate the final emission. The method then returns the final value.

EmissionsPollingJob: This class continuously polls the active tasks every 100ms in order to update the emission calculation values stored in the engine.

Model Classes: The following are the explanations for the model/information classes.

FinishedTask: The class is a model which contains information about a finished task. The information it contains is a list of associated monitored KEIs, a list of used resources and the duration of the task.

KEI: The class is a model which contains information about a key environmental indicator. It is used to store the information extracted from the KeyEnvironmentalIndicator attributes from the BPMN4ES model. The schema for the model is in figure 8.

Resource: The class is a model which contains information about a resource a task uses. It is used to store the information extracted from the Resource attributes from the BPMN4ES model. The schema for the model is in figure 9.

Factor: Factor is an enum containing the information about emission factors. The enums are used in the calculations for the final task emission. The enum contains the fuel type, unit and factor for the type of fuel.

5.1.3 Sequence Diagram

Figure 6 shows the sequence diagram for the entire project. A process instance is started by a BPM worker which is handled by the Camunda 7 engine. Once the Camunda 7 engine encounters a listener during the process instance, the UniversalTaskMonitor is triggered. The UniversalTaskMonitor then goes through the DOM Elements which the camunda engine set up in order to extract the information contained in the extended model attributes. Once the task has completed, the UniversalTaskMonitor then gets the duration of said task and sends the information, along with the model information, to the emission calculator to get the final emissions value. This value is sent back to the listener and sent back to the engine through setting a variable which the engine can then use in the future.

5.2 Connecting BPMN4ES to the Engine

5.2.1 Modifying the BPMN code

In order to connect and process the BPMN4ES model in Camunda 7, adjustments must first be made to the BPMN file produced by the BPMN4ES modeler created by M. Medema, B. Popescu, V. Andrikopolos, and D. Karastoyanova [5] in order to make it compatible. This includes specifying the Camunda schema, isExecutable and Camunda:historyTimeToLive. After this has been set up you can run the Camunda 7 spring application as normal. Currently the Camunda 7 engine will skip over any BPMN4ES notations since it is not sure how to process it so in order for the engine to use the information from the model, listeners need to be put in front of all the tasks that use BPMN4ES (ie. monitor any KEI). Listeners are used to indicate to the engine that once the task the listener is attached to either starts or ends, there is some custom code that needs to be run. So the listeners listen for the start or end of the task and then trigger whatever listener is specified which can then run the custom code. There are multiple types of listeners depending on what type of activity they

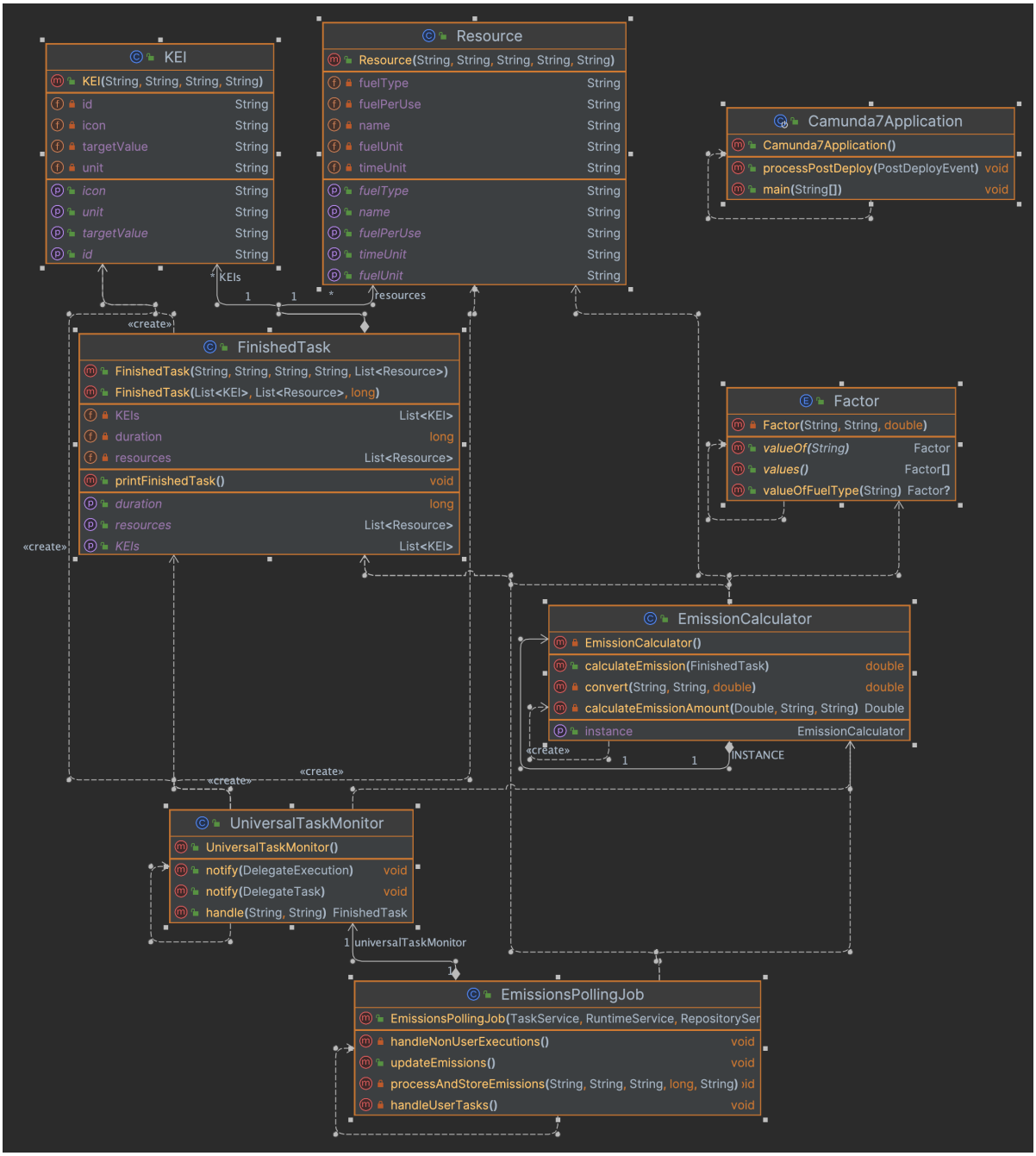

Figure 6: The class diagram for the project.

are listening on. For user tasks, the listener needs to be a task listener and for all other tasks an execution listener is more suited. Since we need to gather the information of the task after it has finished completing, the listeners both need to be end listeners. The following is the correct notation for both types of listeners: User tasks:

```
<bpmn2:userTask id="id" name="Name">
<camunda:taskListener event="complete" delegateExpression="${
    universalTaskMonitor}" />
```

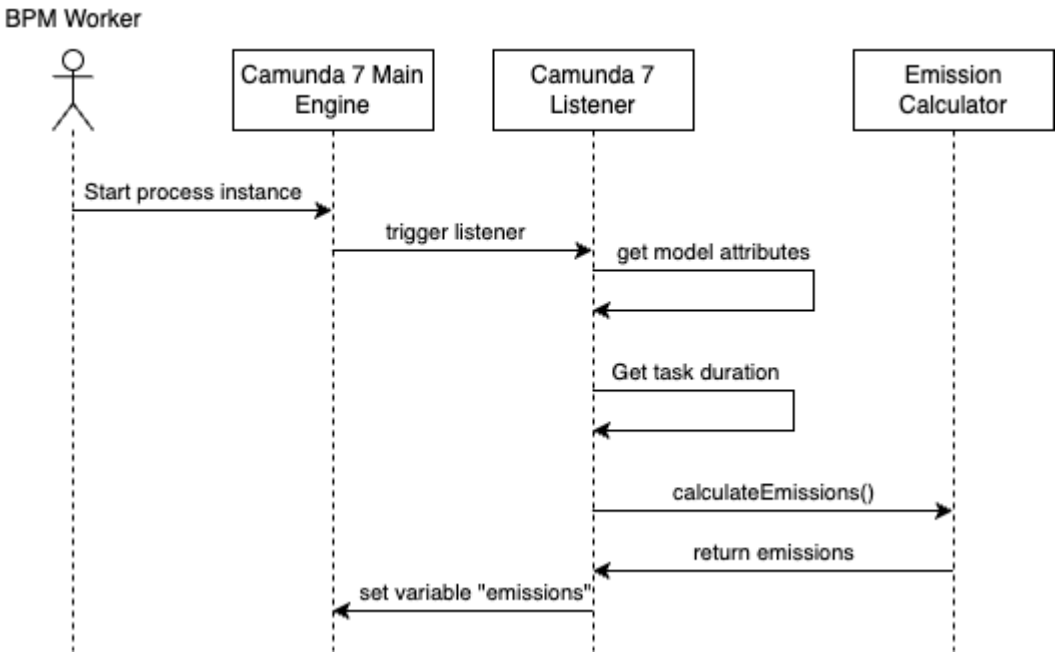

Figure 7: The sequence diagram which goes through the steps of the project.

Listing 2: User Task Listener BPMN code exert

All other tasks:

```
<bpmn2:serviceTask id="id" name="Name" camunda:asyncBefore="true"
   camunda:delegateExpression="${myDelegateExpression}">
<camunda:executionListener event="end" delegateExpression="${
   universalTaskMonitor}" />
```

Listing 3: All Tasks excluding User Task Listener BPMN code exert

5.2.2 Processing the BPMN4ES attributes

In order to process the adapted BPMN code, custom listeners needed to be created. Even though we use two types of listeners (taskListener and executionListener) since the function is the same, the decision was made to have one universal task monitor as the listener which implements both the ExecutionListener and TaskListener interfaces. This leads to both the notify(DelegateTask) and notify(DelegateExpression) having to be overridden, however, as stated before, since the functionality of both methods is the same regardless of it being a user task or another task, the methods both call another method (.handle()) which deals with extracting the information from the BPMN4ES code. This is also done in an attempt to reduce code redundancy and make the code simpler for future programmers to understand.

In the handle method, the code uses the DomElement class in order to get the information inside the attributes (see Figure 8 for schema). DomElements are low level wrappers for XLM

code which the engine uses for non standard BPMN. The engine does not know how to parse non standard BPMN so whenever it encounters any, it uses DomElements to store it. These DomElements can then be used in order to extract the information of the non standard BPMN. The code goes through all the DomElements in order to allow for a one to many relationship between a task and KEIs observed. This information is then stored inside of a FinishedTask model (Listing 4). The model stores all the BPMN4ES attribute information as well as a list of Resource models which will be explained later. The method then returns the list of FinishedTask elements to the main notify function to be used later when processing the emissions.

```
public class FinishedTask {
    private String id;
    private String unit;
    private String targetValue;
    private String icon;
    private List<Resource> resources;
}
```

Listing 4: An exerpt of the FinishedTask model.

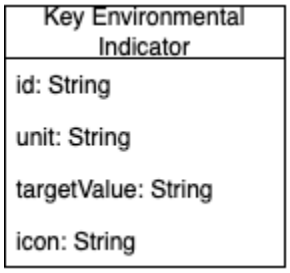

Figure 8: The schema for the Key Environmental Indicator BPMN model.

5.3 Emissions Calculation

The implementation of calculating the emissions is based on the work of [7] while modifying it to be suitable for this project. The basic calculation is based on emission factors, and fuel per use and time elapsed.

5.3.1 Setting up models

Since the calculation relies heavily on what resources are used in the task, this is the first model that needs to be set up. In order to contain the project mostly within the workflow engine, the decision was made to write the information relating to the resources used in the BPMN file along with the KEI information. The schema shown in Figure 7.

This information is processed the same way as the KEI attributes using DomElements, allowing for a one to many relationship between a task and resources. A single resource contains a name, fuel type, fuel per use measurement, fuelUnit, and a time unit (Listing 5).

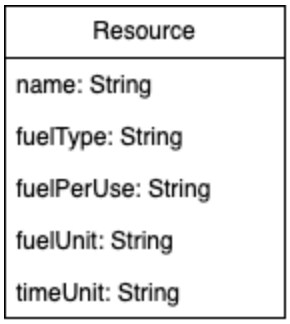

Figure 9: The schema for the Resource BPMN model.

```
public class Resource {
    private String name;
    private String fuelType;
    private String fuelPerUse;
    private String fuelUnit;
    private String timeUnit;
}
```

Listing 5: An exerpt of the Resource model.

In addition, the duration of the task also needs to be calculated. This is achieved through adding a custom transaction listener utilizing the history service to Camunda 7's command context. The history service is used in order to get the historicTaskInstance of the task which can then be queried for the tasks duration in milliseconds.

Emission factors also have to be set up in order for the calculations to work properly. Since this information is not resource specific and is not expected to change frequently, an enum is a good fit.

```
    DIESEL("diesel", "l", 2.70553),
    COAL("coal", "t", 2403.84),
    BIOETHANOL("bioethanol", "l", 0.00901);
```

Listing 6: An exerpt of the Factors enum.

An additional public static method was also included in order to find the value of a factor based on the fuel type.

5.3.2 Calculating the emissions

Calculating the emissions of a task is based on the work of Popescu [7]. The calculation combines the duration of the task with the fuel used per use of a resource and then gets the emission amount of the resource based on that value and the type of fuel that was consumed. This process is done with all the related resources of a task and added together to get the final total emission of a task. Algorithm 1 shows the algorithm for calculating the emission for a task whereas Algorithm 2 shows how the emission amount of a resource is calculated

Algorithm 1 Calculate Emission for a Finished Task

```
1: procedure CalculateEmission(finishedTask)
2:     taskEmission ← 0
3:     duration ← GetDuration(finishedTask)                ▷ In milliseconds
4:     duration ← duration/1000                            ▷ Convert to seconds
5:     resources ← GetResources(finishedTask)
6:     for all resource ∈ resources do
7:         fuelPerUse ← GetFuelPerUse(resource)
8:         consumptionValue ← ParseDouble(fuelPerUse)
9:         unit ← GetTimeUnit(resource)
10:        convertedDuration ← Convert(second, unit, duration)
11:        fuelConsumed ← convertedDuration × consumptionValue
12:        fuelUnit ← GetFuelUnit(resource)
13:        fuelType ← GetFuelType(resource)
14:        emission ← CalculateEmissionAmount(fuelConsumed, fuelUnit, fuelType)
15:        taskEmission ← taskEmission + emission
16:    end for
17:    scale ← 10^4
18:    taskEmission ← Round(taskEmission × scale) /scale
19:    return taskEmission
20: end procedure
```

Algorithm 2 Calculate Emission Amount

```
1: procedure CalculateEmissionAmount(amount, unit, fuelType)
2:     factorEnum ← GetFactorByFuelType(fuelType)
3:     if factorEnum.unit = unit then
4:         return factorEnum.factor × amount
5:     else if factorEnum.unit = TONNES ∧ unit = KILOGRAMS then
6:         return factorEnum.factor × amount/1000
7:     else if factorEnum.unit = KWH ∧ unit = WH then
8:         return factorEnum.factor × amount/1000
9:     else if unit = TONNES ∧ factorEnum.unit = KILOGRAMS then
10:        return factorEnum.factor × amount × 1000
11:    else if unit = KWH ∧ factorEnum.unit = WH then
12:        return factorEnum.factor × amount × 1000
13:    else
14:        ThrowException("Invalid unit")
15:    end if
16: end procedure
```

based on the amount of fuel consumed, fuel unit and fuel type. Algorithm 2 uses the Factor enum defined in the above section whose values have been hard coded.

5.4 Storing Emissions

Once the emission of a task has been calculated, it needs to be stored for the engine's future use. Storing the emission allows the engine to use the variable to determine the flow of the process instance via decision gates allowing for sustainable decision to be made on the fly. Throughout the development, two main approaches were experimented with.

5.4.1 Approach One: Utilizing Listeners

The initial approach for storing the variables in the engine was to use the method available by camunda's interfaces specifically designed for this and integrating it inside our listeners. The storing method is the .setVariable() method and works by providing a string the variable will be identifiable by and the value of the variable. This method can be called in two different ways, Execution.setVariable(name, value) and RuntimeService.setVariable(executionId, name, value), however, for our use case they can be used interchangeably. Setting the variable inside the listeners was necessary since the emission calculation needed the final duration of the task and the only way to retrieve this is once the task has been completed. Initially setting the variables in the listener worked, however when doing further testing, it was impossible to retrieve the set variables. This is due to the variable being set improperly. The commandContext needed to set them properly no longer exists once the task has completed its exection which leads to undefined behaviors and errors.

5.4.2 Approach Two: Polling System

Due to the problems found with approach one, the whole system of how to store the variables needed to be changed. Since we can not store the variable after the task has completed, but we need the task to be completed in order to get the duration for the emission calculation, we need to find a way to store the emission variable as the task is ongoing. This is the basis on which the polling system was created. The polling system is triggered once every 100ms and calculates each active tasks emission with the ongoing duration instead of the completed duration. After the emission has been calculated, the emission value is then stored into the engine with the name activityId follwoed by "_co2Emissions" for the process instance of the task. Once the task completes the polling system no longer updates the emissions value, meaning the final value stored for the emission of the task is the value calculated by the latest 100ms. While this means all the emission values will be slightly smaller than the actual value, because the frequency at which the emission value is polled is so high, the difference should be marginal. The decision to log the accurate emissions calculation with the correct final duration was also made in order for the user to be able to compare the reported value to the actual value. In the case where accuracy is very important, it is also possible to increase the polling frequency, however this goes come with greater computational overhead. Likewise, the frequency can also be decreased to decrease the computational power needed.

5.4.3 Error handling

Due to the frequency of which the polling system sets the process instance variable, the system is prone to OptimisticLockingException errors. This occurs when two threads or transactions try to update the same process instance variable at the same time and is dealt with by camunda, by abandoning the action and throwing the exception. This way of handling is not ideal in our case since it leads to the variable not being up to date. We navigate this by implementing an optimistic locking retry mechanism. By wrapping our variable setting inside a try and while loop, we are able to retry setting the variable if something goes wrong. We wait a small delay and decrease the number of retries until the retries reaches zero, in which case the engine will simple skip updating the variable for that specific duration and move on to the next. This retry system introduces a new problem however, since it is now possible that the process instance no longer exists when trying to update it after the initial try. This is handled with a null check for the process instance before updating the variable.

6 Results

Table 1 shows the requirements of the project in addition to whether they were met in the final project or not.

Table 1: The table of requirements of the project and whether it was achieved.

ID	Requirement	Achieved?	Notes
RF1	The extension must use the BPMN4ES modeling language.	Yes	The extension is compatible with the current version of BPMN4ES by allowing processing of the keyEnvironmentalIndicator Element.
RF2	The extension must be extending a conventional and widely used BPMN workflow engine.	Yes	The extension extends the Camunda 7 engine which is widely used and adheres to the WfMS-reference model.
RF3	The extension must be fully contained and usable within the chosen engine platform.	Yes	No external tools or set ups are required in order to run the extension successfully. While the BPMN4ES modeler developed by M. Medema, B. Popescu, V. Andrikopolos, and D. Karastoyanova can be used to efficiently create and generate the bpmn file, it is not strictly needed to run the extension.
RF4	The extension must calculate the emissions of a monitored task in real time as the process instance is running.	Yes	The extension calculates the emissions while the task is running meaning the calculations are completed as the process instance is running instead of waiting for the whole process to complete.
RF5	The extension must return the final emission value of the task to the engine to be used for future decisions.	Yes	The extension returns the final emission value of the task to the engine by setting the variable "co2Emissions" which is then able to be used by the engine in the future.

ID	Requirement	Achieved?	Notes
RF6	The extension must accurately calculate GHG emissions per process instance, on the level of a task.	Yes	The emission calculation is accurate based on the work by Popescu [7] however the duration can be inaccurate by up to 100ms.
RNF1	The extension must be designed to easily allow additional KEIs to be monitored and calculated.	Yes	The extension easily allows for other KEIs to be monitored as long as a calculator is available for the KEI. Additional information on how to monitor other KEIs can be found in section 7.1.1.
RNF2	The code must be well documented and easy to understand.	Yes	The code is fully documented with Javadocs for all classes, methods and important fields.
RNF3	The extension should be able to be run on multiple software systems.	Yes	The extension works on all software system able to run Camunda 7.

6.1 Discussion

While all our requirements are achieved, the system still has limitations. One such limitation has already been briefly mentioned in section 5.6, mainly the margin of error of the emissions calculation which is introduced by the polling system and negatively affects RF6. Since there is no way to get the finished duration of a task and store it into the engine in Camunda 7, this margin of error will likely exist in most time based emission calculations. Due to the aggressive polling of the standard polling system, the margin of error should already be negligible unless the task has high emission factors. However, as mentioned before, the user can change the frequency of which the polling system is triggered and thus increase or decrease the margin or error. Another limitation is in relation to RNF1, since while the extension is able to monitor additional KEIs, this relies on a calculator for the KEI being available to the extension. Assuming these calculators exist and are accessible to the extension, the requirement is met. A final assumption made for the requirements is that the BPMN4ES modeling language is not expanded further than the keyEnvironmentIndicator element. The extension is able to handle the keyEnvironmentIndictaor expanding to tackle other types of KEIs through RNF1, however, it has no support for future elements without manually extending it. It does have the potential to handle future BPMN4ES versions, however, it is very dependent on what kind of logic the future versions require and is not something we are able to say is possible with full certainty currently.

7 Conclusion

Sustainability has become an increasing concern for many businesses due to outside pressures. These pressures include legislation which states large companies must report on their environmental impact, such as the CSRD (Corporate Sustainability Reporting Directive), and societal pressures such as protests demanding businesses focus on the environment, like the Amazon Employees Climate Walkout. There is currently no easy way for companies to monitor the sustainability of their workflows and while there has been developments which attempt to bridge this gap, none of the developments have been fully functional and environmental specific. Extending a conventional BPMN workflow engine (Camunda 7 in this case) to process KEI and Resource modeling attributes and calculate task emissions automatically in the process instance allows for easy monitoring and enables adapting of the workflows with a sustainability focus to be seamless. The extension utilizes features of the Camunda 7 engine like listeners in order to automatically calculate the emissions using an external emission calculator and is set up to be easily extendable to allow the other main KEIs to be monitored with minimal set up. There is still much to do to make workflows fully sustainable; however, this extension sets up a nice base for future work and is a realization of previous work within the field.

7.1 Future Work

There are still a lot of areas of improvement when it comes to the extension. Below are some ideas for future work for the extension to be further realized.

7.1.1 Additional KEI types

Currently the extension only fully supports the monitoring and calculation of a process instance's emissions, however it has been designed to support additional KEIs. In order to add a KEI, one must simply define an associated calculator to calculate the KEI's consumption and add it in the relevant case of the switch statement present in the listeners. In the case where additional external information is needed for the KEI calculation, simply add it to the resource model (both in the bpmn file and the Resource model in the code) and extract the information in the attribute by adding the following code in the handle method and changing the resourceNamePlaceholder to accureatly represent the new information:

```
String resourceNamePlaceholder = resourceDOM.getAttribute("
    resourceNamePlaceholder");
```

Listing 7: Example code for extracting additional resources.

This allows the extension to be able to use the information stored in the attributes which can then be passed onto the external calculator.

7.1.2 More Accurate Calculations

Currently, the emission calculation is accurate in most cases; however, since it is based on

task duration, it struggles in situations where downtime occurs. A more accurate emission calculation would allow for more accurate reporting and thus better informed environmental decisions. Once a better calculator has been developed, integrating it with the extension is the same process as adding an addition KEI type. Simply call the new calculator in place of the old one in the associated switch case.

7.1.3 Additional Modeling Capabilities

The current version of the extension only calculates the requested KEI values and stores the final requested values as variables in the engine due to the limited capabilities of the BPMN4ES modeling tool currently developed. Due to how the extension is set up and it storing the final values as variables in the engine, it is set up to allow for future modeling capabilities like decision gates based on the KEI values. One limitation of the current solution is its inability to handle boundary events based on task emission as the emission calculation is based on task duration and thus needs to be completed in order to calculate the emission value. The new solution would need to have a different emission calculation and would be unable to use listeners.

7.1.4 Reporting

Camunda 7 allows for reporting of the standard KPI values, allowing for easier viewing of the KPIs. This would be very useful to have for KEIs was well. Additional research would need to be completed in order to see how feasable this is, however, since the KEI values are stored in the engine it should be possible. In the case that complications arise with using existing infrastructure in Camunda to achieve it, one can extract the variables manually and store them in a json file externally.

8 Acknowledgments

I would like to thank Michel Medema for his foundational work as well providing me with relevant literature in the start of the project. I would also like to thank him for making time for meeting with me and guiding me through the process.

Additionally, I would like to thank Dimka Karastoyanova for allowing me the opportunity to work on such an interesting topic.

References

[1] Jan Vom Brocke, Stefan Seidel, and Jan Recker. *Green Business Process Management: To- wards the Sustainable Enterprise*. Springer, Berlin, Heidelberg, 2012.

[2] Camunda bpm platform. https://github.com/camunda/camunda-bpm-platform.

[3] Rainer Endl and Martin Meyer. Potential of business process modelling with regard to available workflow management systems. 01 1999.

[4] Ivo Hristov and Antonio Chirico. The role of sustainability key performance indicators (kpis) in implementing sustainable strategies. *Sustainability*, 11(20), 2019.

[5] Michel Medema, Bogdan Popescu, Vasilios Andrikopoulos, and Dimka Karastoyanova. A life cycle and enabling concepts for green business process management. 2025.

[6] Idil Oksuz. Modeling sustainability in business processes. Master's thesis, University of Groningen, 2024.

[7] Bogdan-Petrut Popescu. Environmental sustainability calculator service for business processes. Master's thesis, University of Groningen, 2024.

[8] J. Recker, M. Rosemann, A. Hjalmarsson, and M. Lind. Modeling and analyzing the carbon footprint of business processes. 2012.

[9] UNDP. Sustainable development goals.

[10] Valerio van den Broek. Going greener through bpm: A method for assessing processes environmental footprint and supporting continuous improvement. Master's thesis, Einhoven University of Technology, 2015.

[11] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Berlin Heidelberg, Germany, 2019.