

BACHELOR'S THESIS



UNIVERSITY OF GRONINGEN

FACULTY OF SCIENCE AND ENGINEERING

Modeling Sustainability in Business Processes

Author:
Idil Oksuz

Supervisors:
Dimka Karastoyanova
Michel Medema

Submitted
September 18, 2024

Abstract

The modeling of sustainability in business processes has become increasingly critical as organizations strive to align their operations with environmental and social responsibilities. Traditional business process management frameworks, such as Business Process Model and Notation (BPMN), are effective in representing operational workflows but lack the capability to incorporate sustainability metrics. This limitation results in missed opportunities for organizations to assess and improve the environmental impact of their business activities.

This thesis aims to address this gap by developing a custom extension to BPMN, enabling the integration of Key Environmental Indicators (KEIs) directly into business process diagrams. The primary objective is to enhance the ability of organizations to visualize, manage, and optimize the ecological footprint of their operations, thereby fostering more sustainable business practices.

Acknowledgements

I would like to express my gratitude to my supervisors, Dimka Karastoyanova and Michel Medema, for their continuous support throughout the project and for providing me with the opportunity to work on the development of this research. Also, I would like to thank the researchers at Eindhoven University, V. van den Broek for their expandable work.

Contents

1	Introduction	1
2	Literature Review	3
2.1	Background Information	3
2.1.1	Business Process Model and Notation (BPMN)	3
2.1.2	Diagrams	4
2.1.3	Versions and Extensions	4
2.1.4	BPMN and Sustainability	4
2.1.5	Integration of Sustainability Modeling into BPMN	5
2.1.6	The Role of XML Meta-Model in BPMN	5
2.1.7	The Role of Web-based BPMN Editor	6
2.2	Related Work	7
2.2.1	Incorporating the Environmental Dimension into BPMN	7
2.2.2	Approach	8
2.2.3	Literature Search Methodology	9
2.2.4	Existing Extensions of BPMN	9
3	Requirements	11
3.1	Functional Requirements (FR)	11
3.2	Non-Functional Requirements (NFR)	12
4	Materials and Methods	13
4.1	Data Collection Methods	13
4.2	Tools and Technologies Used	13
4.2.1	Programming Language and Frameworks	13
4.2.2	Libraries and Packages	14
4.2.3	Development Frameworks	14
4.2.4	User Interface	14
4.3	Documentation and Version Control	14
4.3.1	Run the Application	14
5	Implementation	15
5.1	Extending the XML Meta-Model	15
5.1.1	New Elements and Attributes	15
5.1.2	Integration with Existing BPMN Elements	16
5.2	Alternatives to the Approach	17
5.2.1	Decorator Pattern	17
5.3	Extending the BPMN-js	17
5.3.1	Custom Renderer	17
5.3.2	Custom Context Pad	20
5.4	Save Button Implementation	25
6	Results	27
7	Conclusion	29
7.1	Future Work	29
	Bibliography	32

List of Figures

1.1	The devil's quadrangle with extension	2
2.1	Key elements of BPMN	4
2.2	List of KEIs	5
2.3	"bpmn-js-example-custom-element" Example View	6
2.4	"bpmn-js-example-model-extension" Example View	7
2.5	Example security metric	9
2.6	Example use-case of security metric	10
2.7	Remote healthcare monitoring system with security extensions	10
5.1	Example Task With KEI Value	19
5.2	Custom KEI Menu Example	20
5.3	Ask User For If The Task is Measured	24
5.4	Example of Asking User Measurement Value for KEI	25
5.5	Ask User For If The Task is Monitored	25
5.6	Example Saved XML File	26

Chapter 1

Introduction

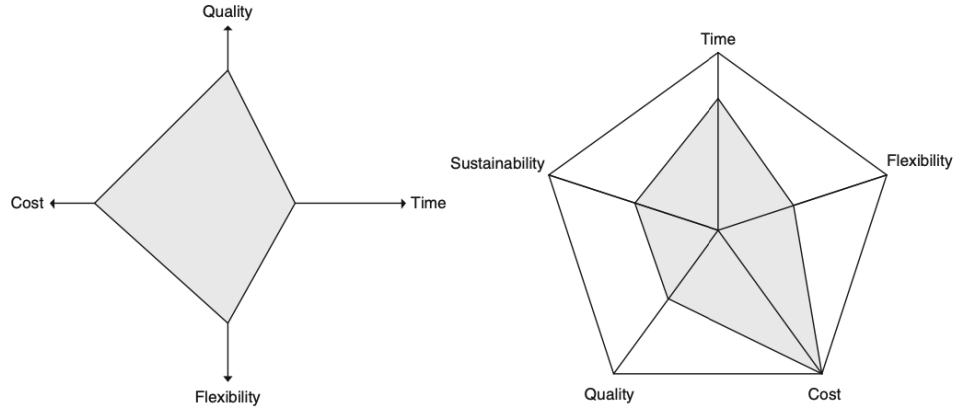
Businesses are under increasing pressure to incorporate sustainability into their basic operations in an era marked by increasing environmental awareness and a growing need for sustainable development [1, 2]. Shifting to more environmentally friendly methods is not just required by law but also represents a strategic move that can enhance the reputation of a company, increase efficiency, and open new avenues for growth [3]. Even with this shift, there remains a noticeable gap in the way business processes are modeled to reflect sustainability concerns, particularly when utilizing standardized frameworks like Business Process Model and Notation (BPMN) [4].

BPMN is a widely adopted standard for representing business processes, offering a comprehensive graphical notation that bridges the communication gap between business analysts and technical developers [5]. While BPMN excels in modeling operational processes, it lacks built-in capabilities for capturing and representing sustainability metrics [6]. This limitation hinders businesses from effectively assessing and improving the environmental impact of their processes, leading to missed opportunities for continual improvement in sustainability performance.

To address these challenges, the concept of Green Business Process Management (Green BPM) [6] has emerged as a crucial paradigm. Green BPM extends traditional Business Process Management (BPM) by explicitly incorporating environmental objectives into business process design and management. It focuses on minimizing the environmental footprint of business processes through the integration of sustainability metrics and the continuous improvement of processes based on these metrics. Green BPM is not just about optimizing resource usage but also about fostering a culture of sustainability within organizations. It provides a framework for companies to systematically assess, monitor, and enhance the sustainability of their operations [7].

As organizations become more conscious of the necessity to operate sustainably, they increasingly seek business processes that are enabled by information technology (IT) and successful not only in economic terms but also in their ecological and, potentially, social impacts [8]. Traditionally, business process management has focused on four key dimensions: time, cost, quality, and flexibility, often referred to as the "devil's quadrangle" [9]. However, with the growing importance of sustainability, this classical model is evolving into what can be termed the "devil's pentagon" [9], where sustainability is recognized as a vital, emerging dimension in business process management. Key ecological performance indicators, such as carbon emissions, data center energy usage, and renewable energy consumption, are now becoming central to managerial agendas. As shown in the figure below (1.1), sustainability is considered as important as the other values.

Figure 1.1: The devil’s quadrangle with extension



According to Vom Brocke [6], Green BPM involves the adaptation of BPM methodologies to include environmental performance indicators alongside traditional business metrics. This holistic approach enables businesses to make informed decisions that balance economic goals with environmental responsibilities. As organizations increasingly recognize the financial and ethical imperatives of sustainable practices, Green BPM is becoming an integral part of corporate strategies aimed at achieving long-term sustainability goals [10].

The key research question that this thesis seeks to address is:

- How can BPMN be extended to model environmental sustainability in business processes?

In exploring this, the thesis will answer the following sub-questions:

- What are the key environmental sustainability metrics that are relevant for integration into BPMN?
- How can these metrics be integrated into the current open-source code of BPMN modeling?

This thesis seeks to contribute to the growing body of knowledge in Green BPM by exploring how BPMN can be extended to incorporate environmental sustainability metrics, enabling businesses to model and optimize their processes with a clear focus on sustainability. By integrating key environmental indicators (KEIs) into BPMN, this research aims to enhance the capacity of organizations to visualize and manage the ecological footprints of their operations. The study will explore existing literature on BPMN and sustainability, identify relevant metrics, and implement these metrics within the BPMN framework through a custom extension module.

The results of this study will have a major impact on the academic study of BPM as well as the real-world use of BPMN in promoting sustainable business practices. This work extends BPMN’s technical capabilities and encourages a more sustainable approach to company operations in a world with limited resources by offering an organized method for incorporating sustainability into business process modeling.

Chapter 2

Literature Review

In the section below, we will present the literature review for the bachelor’s project. We start with the background information in sub-section 2.1. Next, we will continue with the related work in the sub-section 2.2.

2.1 Background Information

2.1.1 Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) is a standardized graphical notation used to depict business processes in a workflow. Developed by the Object Management Group (OMG) [11], BPMN provides a comprehensive set of symbols and diagrams that help stakeholders—including business analysts, process participants, and technical developers—understand and communicate complex business processes with ease.

BPMN is designed to bridge the gap between business process design and implementation. Its primary goal is to provide a notation that is easily understandable by all business users, while detailed enough to support the technical implementation of complex process workflows. The BPMN standard encompasses various elements such as events, activities, gateways, and flows that can be combined to create detailed process models.

Key Elements

The following key elements [11] of BPMN are crucial for understanding which components will be extended:

Flow Objects

- **Events:** Represent occurrences that trigger or influence the flow of the process. Events can be classified into start, intermediate, and end events.
- **Activities:** Denote work that needs to be performed. Activities can be simple tasks or sub-processes.
- **Gateways:** Control the divergence and convergence of process flows, determining branching, forking, merging, and joining of paths.

Connecting Objects

- **Sequence Flows:** Indicate the order in which activities are performed.
- **Message Flows:** Show the flow of messages between different participants.
- **Associations:** Link artifacts such as text annotations and data objects to flow objects.

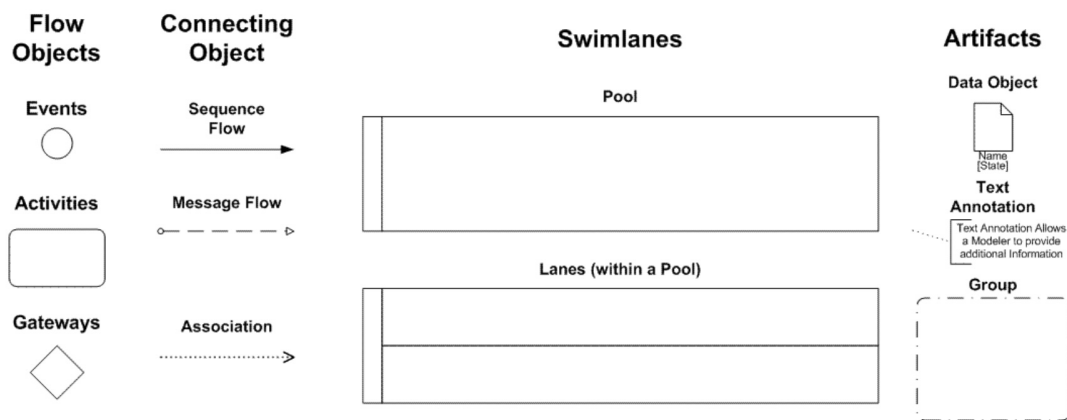
Swimlanes

- **Pool:** Represents a participant in a process.
- **Lanes within a Pool:** Sub-partitions within a pool, used to organize and categorize activities.

Artifacts

- **Data Objects:** Represent data required or produced by process activities.
- **Text Annotations:** Provide additional information about a process element.
- **Groups:** Visual mechanisms to group elements of a diagram informally.

Figure 2.1: Key elements of BPMN



2.1.2 Diagrams

BPMN diagrams, also known as Business Process Diagrams (BPDs), are visual representations of business processes. They serve as blueprints for designing, analyzing, and improving business workflows [12]. BPMN diagrams are composed of the key elements mentioned above, organized to clearly represent the sequence of activities and their interactions. At the conclusion of our bachelor's thesis, the developed application will be capable of saving specific process diagrams that include sustainability metrics with the push of a button.

2.1.3 Versions and Extensions

Over time, BPMN has evolved through several versions, each introducing new features and improvements. The current version, BPMN 2.0, provides advanced capabilities for modeling complex workflows and integrating them with other systems. BPMN 2.0 also supports the extension of its core elements, allowing for customization and the addition of new attributes to meet specific requirements. Therefore, we will use the latest version of BPMN, BPMN 2.0, as it provides easier methods for extension. This is advantageous because we aim to extend the notation to include sustainability metrics.

2.1.4 BPMN and Sustainability

While BPMN offers robust tools for modeling business processes, it currently lacks dedicated elements for capturing sustainability metrics. This gap presents an opportunity to extend BPMN to include environmental sustainability metrics. By integrating these metrics into BPMN, organizations can model and analyze the sustainability impact of their processes, leading to better-informed decision-making and continuous improvement in sustainability performance.


















2.1.5 Integration of Sustainability Modeling into BPMN

In order to incorporate sustainability modeling into BPMN, BPMN 2.0 must be expanded to include environmental issues, focusing on the environmental impacts of processes. This expansion allows for the visualization and management of ecological footprints across business processes.

The identification and definition of Key Environmental Indicators (KEIs) are the first steps in incorporating sustainability into BPMN. These indicators provide insights into the environmental impact of various business activities. KEIs are chosen based on their applicability to specific processes and their capacity to provide useful data. Understanding the environmental impact of each activity allows KEIs to guide process designers on sustainability performance during the process design stage.

The study conducted by V. van den Broek [13] from the University of Eindhoven compiled and summarized sustainability reporting frameworks from both academic and non-academic sources. The results, shown in Figure 2.2, reveal both similarities and differences between academic and business-focused approaches.

Figure 2.2: List of KEIs

Category	Indicators	Description
Energy	Total energy	 Total energy consumed
	Renewable energy	 Wind, Solar, Run-Of-River Hydro, Reservoir Hydro, Wood, Food Products, Biomass from agriculture, Geothermal Energy
	Non-Renewable energy	 Fossil (Hard Coal, Lignite, Crude Oil, Natural Gas, Coal Mining Off-Gas, Peat), Nuclear, Primary Forest (Wood and Biomass from primary forests)
	Indoor energy	 Energy used for indoor activities
	Transportation energy	 Energy used for transportation
	[Single source of energy]	Energy provided by a source that could be particularly relevant for the business
Waste	Total waste	 Total waste produced
	Recyclable waste	 Waste rendered recyclable in Annex III of Directive 2008/98/EC
	Non-Recyclable waste	 Waste rendered non-recyclable in Annex III of Directive 2008/98/EC
	Hazardous waste	 Waste rendered hazardous in Annex III of Directive 2008/98/EC
	[Single waste material]	Single waste material produced in the process
Water	Total water withdrawal	 Total water withdrawn
	Water Non-consumptive use	 Water physically withdrawn from the environment and returned
	Water Use	 Water use that either reduces the quality or quantity of water that is returned
	Water Pollution	 Volume of water polluted, namely grey water
Emissions To Air	Total emissions to air	 Total emissions to air
	GHGs emissions	 Greenhouse gas emissions expressed in g of CO _{2eq} (CO ₂ , CH ₄ , N ₂ O, O ₃ , CCL ₂ F ₂ , CCL ₂ F ₂ , SF ₆)
	CO ₂ emissions	 Amount of CO ₂ emissions
	NOx and SOx emissions	 Total emissions of nitrogen oxides and oxides of sulfur
	[Other single gas]	Amount of specific gas produced by the process relevant for the business

*the categories presented are non-exclusive; depending on the organization's criticalities, a choice can be made between the classifications proposed

2.1.6 The Role of XML Meta-Model in BPMN

The XML meta-model serves as the foundation for representing BPMN elements and their interactions in a machine-readable format, facilitating seamless integration and analysis of process information.

To implement the selected sustainability metrics effectively, extending the meta-model of the BPMN notation is crucial, as the application will adhere to this meta-model. The XML code generated and saved by users will represent an instance of this extended meta-model. It was, therefore, necessary to research and identify the appropriate XML meta-model code to extend.

The global specification of BPMN, maintained by the Object Management Group (OMG), provides a framework for business process modeling. The Semantic.xsd file within the BPMN specification is crucial, as it defines the rules of BPMN elements within the XML code. This file offers a comprehensive structure for ensuring compatibility with existing BPMN standards, which is essential when extending the meta-model to incorporate new elements.

For example, consider the <task> element in the BPMN XML schema. The following snippet from the Semantics.xsd file demonstrates a basic task definition:

Listing 2.1: Example of a BPMN Task Element

```
<xsd:complexType name="tTask">
  <xsd:complexContent>
    <xsd:extension base="tActivity"/>
  </xsd:complexContent>
</xsd:complexType>
```

In this snippet, the `<task>` element is defined as an extension of the `<tActivity>` element, which is a core part of the BPMN flow elements. This ensures that the task element inherits all properties and behaviors of an activity while allowing further customization as needed.

2.1.7 The Role of Web-based BPMN Editor

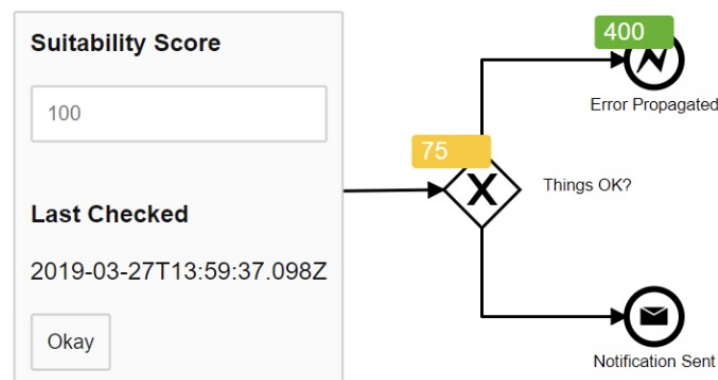
Web-based BPMN editors play a crucial role in extending BPMN to incorporate sustainability metrics. These editors provide a user-friendly interface for creating and managing BPMN diagrams, and their underlying codebase can be extended to support additional functionalities.

"BPMN-js" is a widely-used open-source BPMN modeling tool developed by Camunda. It is built using JavaScript and supports BPMN 2.0. The core library of BPMN-js offers extensive features for rendering BPMN diagrams, managing elements, and ensuring compliance with BPMN standards.

In exploring how to extend BPMN-js to include sustainability metrics, two extension possibilities from the GitHub repository of bpmn.io were identified: "bpmn-js-example-custom-elements" and "bpmn-js-example-model-extension."

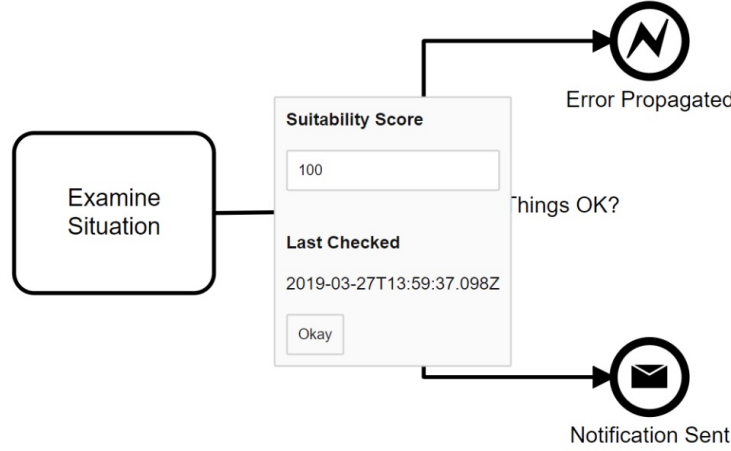
"BPMN-js-example-custom-elements" demonstrates how to extend BPMN-js by adding custom elements to BPMN diagrams. It provides a clear framework for adding custom elements, allowing developers to implement specific features without altering the entire BPMN-js infrastructure. This example focuses on visually manipulating tasks in the diagram, allowing new elements to be rendered within the BPMN diagrams. It also includes custom rendering and editor controls to display these custom elements.

Figure 2.3: "bpmn-js-example-custom-element" Example View



In contrast, "bpmn-js-example-model-extension" focuses on extending the BPMN-js model itself rather than the visual aspects of the diagrams. This approach demonstrates how to read, modify, and write BPMN 2.0 diagrams containing custom extension attributes and elements. The primary advantage of this approach is that it allows for the addition of new data to the BPMN model without changing the visual representation of the tasks.

Figure 2.4: "bpmn-js-example-model-extension" Example View



An example XML element from this approach looks like this:

```
<bpmn2:task id="Task_1" name="Examine Situation" qa:suitable="70">
  <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
  <bpmn2:extensionElements>
    <qa:analysisDetails lastChecked="2015-01-20" nextCheck="2015-07-15">
      <qa:comment author="Klaus">
        Our operators always have a hard time to figure out.
      </qa:comment>
      <qa:comment author="Walter">
        I believe this can be split up in a number of activities.
      </qa:comment>
    </qa:analysisDetails>
  </bpmn2:extensionElements>
</bpmn2:task>
```

While both extension examples offer valuable methods for extending BPMN-js, the "bpmn-js-example-custom-elements" approach focuses on the visual manipulation of tasks, making it more suitable for this project, as it requires direct visual representation of new elements. On the other hand, "bpmn-js-example-model-extension" focuses on enhancing the model with additional data without necessarily changing the visual representation.

2.2 Related Work

2.2.1 Incorporating the Environmental Dimension into BPMN

The inclusion of the *environmental dimension* in Business Process Model and Notation (BPMN) is a growing area of research, particularly as businesses are increasingly expected to monitor and mitigate their environmental impact. Addressing the *Labeling of Impact* on the environment requires meticulous monitoring and benchmarking of a system's energy footprint and other key performance indicators (KPIs) or Key Environmental Indicators (KEIs). One framework that aims to provide this level of insight is the *Sustainability Balanced Scorecard (SBSC)*, which merges the traditional *Balanced Scorecard (BSC)* concept with the *Triple Bottom Line (TBL)* approach. The *TBL* focuses on measuring economic, social, and environmental impacts, while the *BSC* tracks performance through financial, customer, internal business processes, and learning and growth perspectives, creating a comprehensive approach to sustainability assessment [14].

While SBSC holds promise for aligning organizational performance with sustainability goals, it faces significant challenges. The vast number of performance indicators, such as energy usage, carbon emissions, and social equity, increases the complexity of its implementation. Additionally, the absence of efficient tools to collect, assess, and analyze these indicators in real time limits the

framework’s practicality in many business contexts. As a result, while SBSC provides a structured approach to measuring sustainability, its effectiveness is constrained by the difficulty of managing and interpreting the extensive data required to evaluate environmental, social, and economic impacts [2] .

Methods for Modeling Environmental Impact

A promising approach to incorporating the environmental dimension is through process modeling. By quantifying resource consumption and waste production, individual process steps can be analyzed to determine their cumulative environmental impact [15]. Studies such as that of Opitz et al. [15] highlight the importance of process design in both *BPM* and *Green BPM*. In particular, *green modeling* can help organizations visualize the environmental impact of their processes by setting key factors and environmental metrics for activities and resources. Various process modeling languages, such as BPMN and *Event-driven Process Chains (EPCs)*, can be extended to model environmental factors like the total carbon footprint of a process.

BPMN Extensions for Sustainability

Several studies have proposed extensions to the BPMN framework to include environmental metrics. Recker et al. [16] introduced specific BPMN elements to represent the consumption of fuel, paper, and electricity. Their approach also characterizes activities that produce *Greenhouse Gas (GHG)* emissions, allowing businesses to track sources and types of emissions, such as those from fuel or electricity. This work demonstrates how BPMN can be modified to account for energy consumption and CO₂ production.

Houy et al. [17] further explored the incorporation of the environmental dimension in *Green BPM*, emphasizing the need for sustainability metrics in process modeling to improve sustainability awareness. This approach allows businesses to monitor resource consumption, waste generation, and emissions in real-time and refine their processes to enhance sustainability.

Challenges in Environmental Modeling

One of the primary challenges in integrating the environmental dimension into BPMN is the large number of *performance indicators* that need to be monitored. The complexity of establishing an energy footprint, for example, can make it difficult to assign appropriate labels and benchmark processes accurately. Small changes in system architecture or processes can also drastically alter these indicators [15].

Another challenge is the *visualization of sustainability metrics*. While modeling tools such as BPMN offer clear ways to represent the control and data flow, their capability to fully integrate sustainability metrics remains limited.

2.2.2 Approach

Sustainability is critical for organizational operations in a world of resource scarcity. Therefore, incorporating sustainability metrics into business processes is highly relevant. We will conduct a literature review to identify the most important environmental sustainability metrics. Based on our findings, we will then implement these metrics into our BPMN schema, since there has been little to no implementation of this extension in a programming environment.

This leads to the research question: *How can BPMN be extended to model environmental sustainability in business processes?*

After reviewing possible environmental sustainability metrics, we will incorporate these into the meta-model of the BPMN schema. As the research focuses on modeling Business Process Management (BPM), we will also examine how these sustainability metrics integrate with the current BPMN and how they can be accurately modeled. Finally, we will extend the source code of BPMN to allow the inclusion of sustainability metrics.

Sources used as a starting point for this literature research include: "Going greener through BPM: a method for assessing processes’ environmental footprint and supporting continuous improvement" [13], and "Green Business Process Management" [6].

2.2.3 Literature Search Methodology

The literature review for this project was conducted systematically to ensure comprehensiveness and reproducibility. The objective was to identify key environmental sustainability metrics relevant to Business Process Management (BPM), particularly in the context of BPMN modeling. A multi-stage process was used to achieve this.

The search was carried out across major academic databases, including Google Scholar, IEEE Xplore, Springer, and Scopus, with a focus on the following keywords:


- "Sustainability in BPM"
- "BPMN extensions"
- "Green BPM"
- "Key Environmental Indicators (KEIs)"

2.2.4 Existing Extensions of BPMN

Security is a critical aspect often overlooked in the initial stages of business process modeling. Traditional BPMN does not adequately support the representation of security requirements such as confidentiality, integrity, and availability. To address this gap, several researchers [18] have proposed extensions to BPMN that incorporate security elements.

Rodriguez [19] introduced a comprehensive BPMN extension aimed at integrating security requirements into business process diagrams. The proposed extensions include security-specific artifacts and annotations, such as non-repudiation, integrity, privacy, and access control. An example of these extensions is shown in Figure 2.5.









Figure 2.5: Example security metric

Name	SecurityRequirement	Notation
Description	Abstract class containing security requirements specifications. Each security requirement type must be indicated in some of its subclasses.	
Constrains	<ul style="list-style-type: none"> - A security requirement must be associated with a secure activity context SecurityRequirement inv: self.SecureActivity ->size()=1 - The notation must be completed for each security requirement. It must be used one security requirement type. 	
Tagged Values	SecurityRequirementType: SecReqType	

Each security requirement is associated with specific BPMN elements, such as pools, lanes, and message flows, enhancing the ability to model secure business processes from the perspective of business analysts.

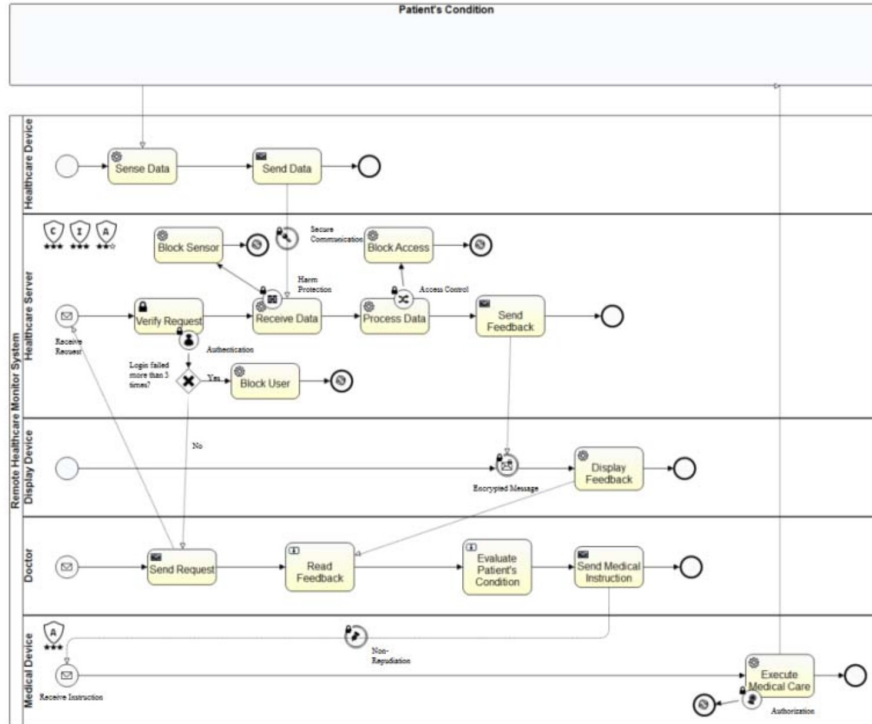
Koh and Zhou [18] focused on extending BPMN to model security requirements specifically for healthcare processes. With the increasing use of digital information and IoT technology in healthcare, securing electronic health records (EHR) is essential. Their extension introduces new BPMN elements, such as security tasks, security events, and security indicators, which are seamlessly integrated with existing BPMN notations, as shown in Figure 2.6.

Figure 2.6: Example use-case of security metric

Element	Type	Design
Security Task	Task	
Authentication	Boundary Event	
Access Control	Boundary Event	
Authorization	Boundary Event	
Harm Protection	Boundary Event	
Encrypted Message	Intermediate Event	
Non Repudiation	Intermediate Event	
Secure Communication	Intermediate Event	

The authors also proposed security boundary events, such as authentication, access control, and harm protection, which can be attached to standard BPMN elements to represent security checkpoints. This allows explicit modeling of security processes, as seen in Figure 2.7, which shows an example of a remote healthcare monitoring system with security extensions.

Figure 2.7: Remote healthcare monitoring system with security extensions



These security extensions provide a blueprint for how new elements and attributes can be seamlessly integrated into BPMN. Similarly, sustainability tasks and events can be introduced to capture environmental metrics. The process of defining and validating new BPMN elements for security can be mirrored in the development of sustainability extensions. This includes creating new graphical symbols, setting constraints for their interaction with existing BPMN elements, and ensuring that they are comprehensively integrated into the BPMN framework.

Chapter 3

Requirements

The primary requirement for this project was to create a system that enhances BPMN (Business Process Model and Notation) diagrams with sustainability metrics. Below are the requirements categorized by the acronyms F and NF, which signify whether the requirement is Functional or Non-Functional respectively. Functional requirements define what a product must do and what its features and functions are, while Non-Functional requirements describe the general properties of a system. Additionally, the requirements have been assigned keys, so that they can be referred to when analyzing the results of the software.

The functional requirements focus on characteristics such as task management, KEI (Key Environmental Indicator) handling, contextual menus, and diagram rendering. The non-functional requirements describe the software's characteristics which involve performance, scalability, reliability, maintainability, usability, and security.

3.1 Functional Requirements (FR)

- **FR1:** The system shall extend BPMN to incorporate sustainability metrics by integrating a custom sustainability extension module (`smExtension`).
- **FR2:** The system shall allow the definition and assignment of Key Environmental Indicators (KEIs) to BPMN tasks and elements, including energy consumption, renewable energy, non-renewable energy, indoor energy, transportation energy, single source of energy, carbon dioxide emissions, water usage, and waste generation.
- **FR3:** The system shall provide a custom context pad that allows users to edit KEI properties for BPMN elements, including setting whether a task is measured or monitored.
- **FR4:** The system shall ensure that KEI properties are preserved when BPMN tasks are replaced or modified, including the transfer of these properties to new or updated elements.
- **FR5:** The system shall render KEI values visually on BPMN elements, displaying associated icons, values, and monitoring status directly on the diagram.
- **FR6:** The system shall allow the user to edit KEI values through a custom user interface, including options to input values, units, and monitoring status for each KEI.
- **FR7:** The system shall update the BPMN XML to include KEI data as part of the extension elements, ensuring all sustainability metrics are stored in a standardized format.
- **FR8:** The system shall enable the visualization of KEI changes in real-time, ensuring that any modifications to KEI properties are immediately reflected in the BPMN diagram.
- **FR9:** The system shall ensure that when saving the BPMN diagram, all KEI properties and their values are accurately captured and saved within the BPMN XML structure.
- **FR10:** The specification of sustainability metrics (KEIs) for BPMN elements shall be optional. Users can choose whether to assign KEIs to individual BPMN tasks and elements.

3.2 Non-Functional Requirements (NFR)

- **NFR1:** The visual representation of KEI data on BPMN elements should be clear and unobtrusive, ensuring that users can easily distinguish KEI information without overwhelming the BPMN diagram.
- **NFR2:** The KEI data and its visualization should be accessible and usable across different platforms and screen sizes, ensuring a consistent user experience in web-based environments.
- **NFR3:** The sustainability extension should be documented thoroughly, providing clear guidance for developers who may need to extend BPMN in the future.
- **NFR4:** The KEI data and its visualization should be accessible and usable across different platforms and screen sizes, ensuring a consistent user experience in web-based environments.

Chapter 4

Materials and Methods

This section describes the choices behind the technologies used in the software and the rationale for these choices, along with a description of the major data types that the software supports.

4.1 Data Collection Methods

For this project, we utilized BPMN diagrams enhanced with Key Environmental Indicators (KEIs) data to provide insights into sustainability metrics.

BPMN Diagrams

BPMN diagrams were the core data type used in this project. These diagrams represent business processes in a graphical notation, which is both understandable by business users and executable by workflow engines. We mentioned BPMN diagrams in the Literature Review section.

Key Environmental Indicators (KEIs)

KEIs were integrated into the BPMN diagrams to enhance them with sustainability metrics. For this project, a focus was placed on the energy category to facilitate future extensions into other categories. However, we additionally implemented one indicator from each category other than energy. The key KEIs included:

- **Energy Consumption:** Measures the amount of energy used by a task (unit: kWh).
- **Renewable Energy:** Tracks the use of renewable energy sources (unit: kWh).
- **Non-Renewable Energy:** Tracks the use of non-renewable energy sources (unit: kWh).
- **Water Usage:** Measures the amount of water used (unit: liters).
- **Waste Generation:** Measures the amount of waste produced (unit: kg).
- **Carbon Dioxide Emissions:** Measures the amount of CO2 emissions (unit: kg).

4.2 Tools and Technologies Used

The following sections elaborate on the technologies employed, along with their respective benefits and drawbacks.

4.2.1 Programming Language and Frameworks

JavaScript was chosen as the primary programming language for both client-side and server-side development in our project to extend BPMN due to its versatility and extensive nature. Its library and framework selection and its ability to interact with different aspects of the BPMN extension application, such as custom rendering of BPMN components and dynamic changes to BPMN models, make it the right decision for this project. For instance, Node.js enables the effective

administration of server-side operations associated with BPMN model management, while tools such as ‘diagram-js’ and ‘bpmn-js’ facilitate the development of customized BPMN renderers and context pads.

4.2.2 Libraries and Packages

BPMN.js was employed for rendering and editing BPMN 2.0 diagrams, offering comprehensive tools that ensure precise and interactive diagram creation. This library is beneficial due to its depth of functionality and support for BPMN standards.

Diagram.js, which serves as the foundation for BPMN.js, is highly flexible and enables extensive diagram manipulation.

Tiny-svg was utilized for precise Scalable Vector Graphics (SVG) element handling, crucial for the visual rendering of BPMN elements.

Min-dash provided functional programming utilities that streamlined data manipulation, simplifying the codebase.

4.2.3 Development Frameworks

Node.js was adopted as the runtime environment for server-side JavaScript execution. Its non-blocking I/O model and ability to handle asynchronous tasks efficiently make it ideal for building scalable applications. Additionally, Node.js allows for the use of a unified language across the server and client, streamlining development. Express, a lightweight web framework for Node.js, facilitated the construction of the backend API and server-side logic. Its simplicity and extensive middleware ecosystem are its advantages.

4.2.4 User Interface

React was chosen for developing a dynamic and responsive user interface due to its component-based architecture, which promotes reusable and maintainable UI components. Its virtual DOM implementation ensures efficient rendering and performance optimization. HTML5 and CSS3 were used for structuring and styling the web interface, providing industry-standard techniques for creating functional and visually appealing designs.

4.3 Documentation and Version Control

Git was used for version control, facilitating collaborative development and efficient tracking of changes. GitHub provided a platform for repository hosting, enhancing project management. The link to the project: <https://github.com/idiloksuz/thesis>.

4.3.1 Run the Application

You need a NodeJS development stack with npm installed to build the project.

To install all project dependencies execute:

npm install

To start the example execute:

npm start

To build the example into the public folder execute

npm run all

Chapter 5

Implementation

The implementation of the BPMN diagram enhancement system with sustainability metrics was carried out through a series of structured steps, ensuring that the application met all functional and non-functional requirements. The following sections detail the major components and processes involved in the implementation.

5.1 Extending the XML Meta-Model

To integrate sustainability metrics into BPMN, we needed to extend the existing BPMN 2.0 meta-model to accommodate additional attributes and elements related to sustainability. This involved modifying the XML Schema Definition (XSD) file that defines the structure of BPMN models.

5.1.1 New Elements and Attributes

The primary extension involved creating new elements and attributes within the BPMN schema to represent sustainability metrics. The main addition was the **SustainabilityMetricsType**, a complex type that includes various KEIs such as energy consumption, carbon emissions, water usage, and waste produced. Each metric is represented as a **MetricType** with attributes for value and unit.

Listing 5.1: New Elements and Attributes for Sustainability Metrics

```
<xs:complexType name="SustainabilityMetricsType">
  <xs:sequence>
    <xs:element name="energyConsumption" type="MetricType"
      minOccurs="0"/>
    <xs:element name="renewableEnergy" type="MetricType"
      minOccurs="0"/>
    <xs:element name="nonRenewableEnergy" type="MetricType"
      minOccurs="0"/>
    <xs:element name="indoorEnergy" type="MetricType"
      minOccurs="0"/>
    <xs:element name="transportationEnergy" type="MetricType"
      minOccurs="0"/>
    <xs:element name="singleSourceOfEnergy" type="MetricType"
      minOccurs="0"/>
    <xs:element name="wasteGeneration" type="MetricType"
      minOccurs="0"/>
    <xs:element name="waterUsage" type="MetricType"
      minOccurs="0"/>
    <xs:element name="carbonDioxideEmissions" type="MetricType"
      minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="monitored" type="xs:boolean"
    default="false"/>
</xs:complexType>
```

```

</xs:complexType>

<xs:complexType name="MetricType">
  <xs:simpleContent>
    <xs:extension base="xs:decimal">
      <xs:attribute name="unit" type="xs:string"
        use="required"/>
      <xs:attribute name="unknown" type="xs:boolean"
        default="false"/>
    </xs:extension>
    <xs:attribute name="unit" type="xs:string" use="required"/>
  </xs:simpleContent>
</xs:complexType>

```

Advantages of This Approach

- **Seamless Integration with BPMN:** By extending the existing BPMN meta-model, this approach allows for the seamless integration of sustainability metrics into BPMN diagrams. This makes it easy for BPMN users to work with sustainability metrics without learning a new modeling language or tool.
- **Flexibility:** The `SustainabilityMetricsType` is highly customizable, allowing different types of sustainability metrics to be attached to various BPMN elements. This flexibility can accommodate a wide range of use cases, such as tracking energy consumption, water usage, carbon emissions, etc.
- **Preserves Existing BPMN Structure:** The approach builds on top of the existing BPMN schema by extending key elements like `tFlowNode` and `tActivity`. This ensures that the core structure and functionality of BPMN remains intact while enhancing it with new features related to sustainability.

Disadvantages of This Approach

- **Complexity:** Introducing new elements and attributes increases the complexity of the BPMN model. It can also lead to more complex diagrams, which might be harder to interpret.
- **Limited to BPMN:** Since this extension is specific to BPMN, it might not easily integrate with other process modeling languages or tools. Organizations using multiple frameworks may find it difficult to apply sustainability metrics universally across all their tools.
- **Backward Compatibility:** Extending the BPMN schema can potentially lead to issues with backward compatibility if existing BPMN tools or parsers do not support the new `SustainabilityMetricsType` or its attributes.

5.1.2 Integration with Existing BPMN Elements

To ensure the new sustainability metrics could be used seamlessly with existing BPMN elements, we integrated the `SustainabilityMetricsType` into relevant elements such as `tFlowNode`, `tActivity`, and `tGateway`. This integration allows any BPMN element that inherits from these types to include sustainability metrics.

Listing 5.2: Integration with Existing BPMN Elements

```

<xs:complexType name="tFlowNode" abstract="true">
  <xs:complexContent>
    <xs:extension base="tFlowElement">
      <xs:sequence>
        <xs:element name="incoming" type="xs:QName"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="outgoing" type="xs:QName"

```

```

                                minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="sustainabilityMetrics"
                    type="SustainabilityMetricsType"
                    minOccurs="0"/>
    </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

5.2 Alternatives to the Approach

While the current approach of extending the BPMN meta-model using the XML schema provides a direct and structured way to integrate sustainability metrics, there are object-oriented design patterns that could provide greater flexibility and scalability if more metrics were added in the future.

5.2.1 Decorator Pattern

The decorator pattern is an ideal alternative when additional responsibilities, such as new sustainability metrics, need to be added to BPMN elements dynamically.

The Decorator pattern enables adding sustainability metrics to existing BPMN elements without altering their base classes. For instance, BPMN elements like `tFlowNode` or `tActivity` can be decorated with different metrics such as energy consumption, water usage, or carbon emissions. This is particularly advantageous if more sustainability metrics are introduced, as the pattern allows for flexibility by decoupling the metrics from the core BPMN structure.

By applying the Decorator pattern, a `SustainabilityMetricsDecorator` class can wrap around BPMN elements, providing additional attributes (such as metrics) without affecting the base functionality of the original BPMN elements. Each new metric can be added as an additional layer of decoration, ensuring that future expansions do not involve modifying the existing structure.

5.3 Extending the BPMN-js

To support the integration of sustainability metrics into BPMN diagrams, modifications and extensions were made to the BPMN-js library. This included creating custom renderers, extending the context pad, and managing KEI properties.

To manage KEI properties effectively, helper functions were created to ensure that KEI properties are correctly assigned, updated, and preserved. These functions handle the underlying data structures that store KEI values, ensuring consistency and integrity across the BPMN diagrams.

Functionalities are mentioned in the Requirements section of the paper.

This approach to extending the BPMN-js library ensured that the new sustainability metrics were seamlessly integrated into the BPMN tool. By enhancing the rendering and context pad functionalities, the tool now provides users with improved capabilities to manage and visualize the sustainability aspects of their processes.

Here are the crucial functions and classes that were needed for the extension:

5.3.1 Custom Renderer

A custom renderer was developed to visually represent the new sustainability metrics on BPMN elements. This renderer was responsible for displaying KEI properties on tasks and other BPMN elements, ensuring that KEIs were effectively communicated through the diagrams.

Key steps involved in the custom renderer include:

- Extending the base renderer to support additional KEI properties.
- Checking if the element has KEI properties and setting the appropriate color and icon.
- Drawing additional icons and text to represent the KEI values.

Algorithm 1 Constructor

```
1: Function Constructor(eventBus, bpmnRenderer, customContextPad)
2:   CALL BaseRenderer Constructor with eventBus and HIGH_PRIORITY
3:   SET this.bpmnRenderer to bpmnRenderer
4:   SET this.customContextPad to customContextPad
5: End Function =0
```

The ‘Constructor’ function for the ‘CustomRenderer’ class initializes the renderer by setting up its properties and registering an event listener for changes to elements in the BPMN diagram. The ‘super(eventBus, HIGH-PRIORITY)’ call ensures that the ‘CustomRenderer’ inherits from the base renderer with a higher priority, meaning it will take precedence over the bpmn-js rendered.

Algorithm 2 canRender Function

```
1: Function canRender(element)
2:   SET businessObject to element.businessObject
3:   RETURN True if element does not have a labelTarget AND businessObject exists AND
   businessObject.kei exists, otherwise return False
4: End Function =0
```

The ‘canRender’ function is responsible for determining whether the ‘CustomRenderer’ should handle a specific BPMN element. It first retrieves the ‘businessObject’ associated with the element, which contains its core data and properties. The function then checks two conditions: first, that the element is not a label (indicated by the absence of ‘labelTarget’), and second, that the element has a KEI property (‘businessObject.kei’). If both conditions are met, the function returns ‘true’, indicating that the element can be rendered by this renderer.

Algorithm 3 drawShape Function

```
1: Function drawShape(parentNode, element)
2:   CALL bpmnRenderer.drawShape with parentNode and element, and store result in shape
3:   CALL drawKEI with element and parentNode
4:   RETURN shape
5: End Function =0
```

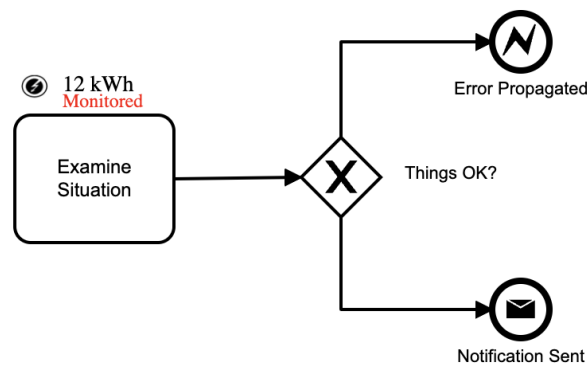
The ‘drawShape’ function is tasked with rendering the visual representation of a BPMN element. It begins by invoking the ‘bpmnRenderer.drawShape’ function, which draws the base shape of the element. After the base shape is drawn, the ‘drawKEI’ function is called to add any relevant KEI icons and values to the shape, enhancing it with sustainability metrics.

Algorithm 4 drawKEI Function

```
1: Function drawKEI(element, parentNode)
2:   IF parentNode is undefined
3:     LOG error message
4:     RETURN
5:   END IF
6:   SET shapeHeight to element.height
7:   SET businessObject to element.businessObject
8:   SET kei to businessObject.kei
9:   SET keiValue to CALL getKEIValue with businessObject and kei
10:  IF kei exists
11:    CREATE an image element img
12:    SET img's attributes for image source, position, width, and height
13:    APPEND img to parentNode
14:    CREATE a text element
15:    SET text's attributes for position, font size, color, and opacity
16:    SET text content to keiValue
17:    APPEND text to parentNode
18:    IF businessObject.monitored is true
19:      CREATE a monitored text element
20:      SET monitored text's attributes for position, font size, color, and opacity
21:      SET monitored text content to "Monitored"
22:      APPEND monitored text to parentNode
23:    END IF
24:  END IF
25: END Function =0
```

The 'drawKEI' function is designed to add the KEI visual elements to a BPMN shape. It first checks if the 'parentNode' where the KEI will be appended is defined; if not, it logs an error and exits. If the 'parentNode' is valid, the function retrieves the shape's height, the KEI type, and its corresponding value. If a KEI is present, the function creates an SVG image element and sets its attributes, including the image path based on the KEI type, its position (now adjusted to be above the shape), and its size. The KEI value is then displayed as text next to the image. If the KEI is monitored, an additional "Monitored" label is displayed in red. Both the image and text are appended to the 'parentNode'.

Figure 5.1: Example Task With KEI Value



Algorithm 5 getKEIValue Function

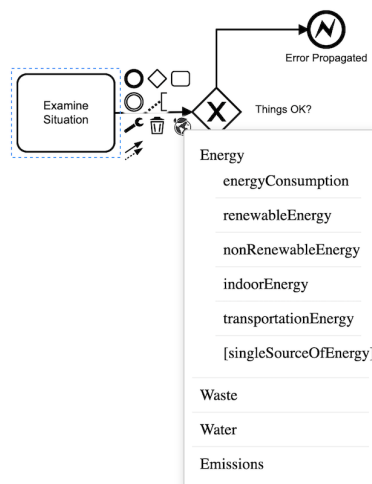
```
1: Function getKEIValue(businessObject, kei)
2:   SET value to businessObject[kei]
3:   SET unit to empty string
4:   SWITCH kei
5:     CASE 'energyConsumption', 'renewableEnergy', 'nonRenewableEnergy', 'indoorEnergy',
       'transportationEnergy', '[singleSourceOfEnergy]'
6:       SET unit to 'kWh'
7:     CASE 'carbonDioxideEmissions'
8:       SET unit to 'kg CO2'
9:     CASE 'waterUsage'
10:      SET unit to 'liters'
11:     CASE 'wasteGeneration'
12:      SET unit to 'kg'
13:     DEFAULT
14:      RETURN "
15:   END SWITCH
16:   IF businessObject.measured is true AND (value is undefined OR value is null)
17:     RETURN "undefined " + unit
18:   ELSE
19:     RETURN (value OR 'undefined') + " " + unit
20:   END IF
21: END Function =0
```

The 'getKEIValue' function retrieves and formats the KEI value from the 'businessObject'. It assigns the appropriate unit based on the KEI type. If the KEI is measured but no value is provided, the function returns "undefined" followed by the unit (e.g., "undefined kWh"). Otherwise, it returns the value with the corresponding unit.

5.3.2 Custom Context Pad

The BPMN modeler was extended with a custom context pad that allows users to manage KEIs directly from BPMN elements. This extension introduces a new context menu to edit and preserve KEI properties. The 'CustomContextPad' class was implemented to achieve this functionality.

Figure 5.2: Custom KEI Menu Example



The main steps involved in extending the application with KEI are:

- Creating a custom context pad that integrates with the existing BPMN modeler.
- Implementing functions to extract, preserve, and manage KEI properties.
- Introducing a user interface to prompt for KEI values and manage their visibility on BPMN elements.

Algorithm 6 Constructor

```

1: Function Constructor(bpmnFactory, config, contextPad, create, elementFactory, injector,
   translate, eventBus, modeling, elementRegistry, popupMenu)
2:   SET this.bpmnFactory to bpmnFactory
3:   SET this.create to create
4:   SET this.elementFactory to elementFactory
5:   SET this.translate to translate
6:   SET this.eventBus to eventBus
7:   SET this.modeling to modeling
8:   SET this.elementRegistry to elementRegistry
9:   SET this.popupMenu to popupMenu
10:  SET this.bpmnModeler to null
11:  IF config.autoPlace is not false
12:    SET this.autoPlace to injector.get('autoPlace', false)
13:  END IF
14:  CALL contextPad.registerProvider with this
15:  eventBus.on('commandStack.shape.replace.postExecute', function(event))
16:    SET context to event.context
17:    SET newShape to context.newShape
18:    # Reapply KEI properties after the shape is replaced
19:    SET keiProperties to CALL this.extractKEIProperties with
       context.oldShape.businessObject
20:    CALL assign(newShape.businessObject, keiProperties)
21:    # Ensure KEI extension elements are correctly handled
22:    CALL this.ensureKEIExtensionElements with newShape.businessObject and
       keiProperties
23:    # Trigger re-rendering of the KEI properties
24:  END Function =0

```

The ‘Constructor’ function initializes the ‘CustomContextPad’ by injecting various dependencies, such as ‘bpmnFactory’, ‘contextPad’, ‘modeling’, and others. The constructor registers this class as a context pad provider and sets up an event listener for shape replacement (‘commandStack.shape.replace.postExecute’). This ensures that KEI properties are preserved when a BPMN element is replaced.

The ‘extractKEIProperties’ function is responsible for retrieving KEI properties from a given BPMN element’s ‘businessObject’. It checks if each KEI property is defined, returning ‘undefined’ if not, and sets default values where necessary.

Algorithm 7 preserveKEIProperties Function

```
1: Function preserveKEIProperties(oldShape, newShape)
2:   SET oldBusinessObject to oldShape.businessObject
3:   SET newBusinessObject to newShape.businessObject
4:   SET keiProperties to a dictionary containing:
5:     kei : oldBusinessObject.kei
6:     energyConsumption : oldBusinessObject.energyConsumption
7:     renewableEnergy : oldBusinessObject.renewableEnergy
8:     nonRenewableEnergy : oldBusinessObject.nonRenewableEnergy
9:     indoorEnergy : oldBusinessObject.indoorEnergy
10:    transportationEnergy : oldBusinessObject.transportationEnergy
11:    singleSourceOfEnergy : oldBusinessObject.singleSourceOfEnergy
12:    carbonDioxideEmissions : oldBusinessObject.carbonDioxideEmissions
13:    waterUsage : oldBusinessObject.waterUsage
14:    wasteGeneration : oldBusinessObject.wasteGeneration
15:    monitored : oldBusinessObject.monitored
16:   CALL assign with newBusinessObject and keiProperties
17:   CALL this.ensureKEIExtensionElements with newBusinessObject and keiProperties
18:   CALL this.modeling.updateProperties with newShape and a dictionary containing
    businessObject: newBusinessObject
19:   FIRE 'element.changed' event on eventBus with element: newShape
20:   LOG 'Preserved KEI properties:' and keiProperties
21: END Function =0
```

The 'preserveKEIProperties' function ensures that KEI properties are maintained when a BPMN element is replaced. It extracts the KEI properties from the old element and assigns them to the new one. This function calls 'ensureKEIExtensionElements' to handle the KEI extension elements in the BPMN XML structure, ensuring that the KEI data is correctly updated.

Algorithm 8 ensureKEIExtensionElements Function

```
1: Function ensureKEIExtensionElements(businessObject, keiProperties)
2:   SET moddle to this.bpmnModeler.get('moddle')
3:   SET extensionElements to businessObject.extensionElements
4:   IF extensionElements is undefined
5:     CREATE extensionElements using moddle.create('bpmn:ExtensionElements')
6:     SET businessObject.extensionElements to extensionElements
7:   END IF
8:   IF extensionElements.values is undefined
9:     SET extensionElements.values to an empty list
10:  END IF
11:  # Filter out non-KEI elements
12:  SET extensionElements.values to filter elements that are either 'sm:Kei' or do not start
    with 'sm:'
13:  SET keiElement to find element in extensionElements.get('values') where $type is
    'sm:Kei'
14:  IF keiElement is undefined
15:    CREATE      keiElement      using      moddle.create('sm:Kei', { value:
    keiProperties.kei })
16:    APPEND keiElement to extensionElements.get('values')
17:    LOG 'Created KEI element:' and keiElement
18:  ELSE
19:    SET keiElement.value to keiProperties.kei
20:  END IF
21:  IF keiProperties.monitored is true
22:    SET keiElement.monitored to 'true'
23:  ELSE
24:    DELETE keiElement.monitored
25:  END IF
26:  # Ensure values are stored even if only measured
27:  CALL      this.handleKEIElement      with      arguments      'energyConsumption',
    'sm:energyConsumption', businessObject, 'energyConsumption', 'kWh',
    keiProperties.monitored, keiProperties.energyConsumption, moddle,
    extensionElements
28:  # Call for every KEI
29: END Function =0
```

The 'ensureKEIExtensionElements' function is crucial for managing the BPMN XML structure. It ensures that KEI-related data is stored as extension elements in the XML. The function creates or updates the necessary KEI elements and assigns values, units, and monitored status accordingly. It calls the 'handleKEIElement' function for each specific KEI type to update the corresponding values.

Algorithm 9 handleKEIElement Function

```
1: Function handleKEIElement(keiType, elementType, businessObject, property, unit, monitored, measuredValue, moddle, extensionElements)
2:   IF businessObject.kei equals keiType
3:     SET keiElement to find element in extensionElements.get('values') where $type is elementType
4:     IF keiElement is undefined
5:       CREATE keiElement using moddle.create(elementType, {})
6:       APPEND keiElement to extensionElements.get('values')
7:     END IF
8:     # Update the value, ensuring it's handled even if previously empty
9:     IF measuredValue is not undefined AND not null
10:      SET keiElement.value to measuredValue.toString()
11:    END IF
12:    # Set or update the unit attribute
13:    IF unit exists
14:      SET keiElement.unit to unit
15:    ELSE
16:      DELETE keiElement.unit
17:    END IF
18:    # Set or update the monitored attribute
19:    IF monitored is true
20:      SET keiElement.monitored to 'true'
21:    ELSE
22:      DELETE keiElement.monitored
23:    END IF
24:  END IF =0
```

The 'handleKEIElement' function is responsible for creating or updating a specific KEI element within the BPMN XML structure. It ensures that the KEI value, unit, and monitored status are correctly set and stored as attributes of the extension elements.

Below are examples of user input in the application.

Figure 5.3: Ask User For If The Task is Measured

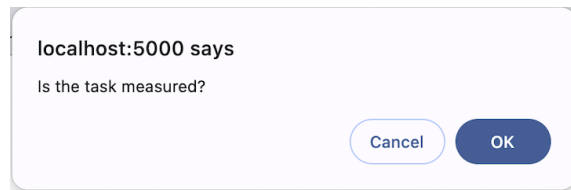


Figure 5.4: Example of Asking User Measurement Value for KEI

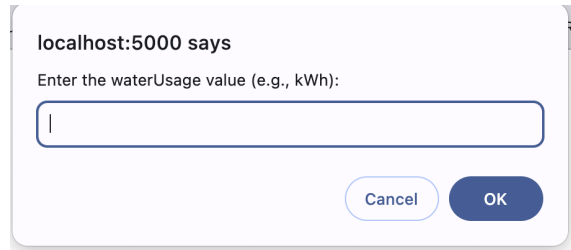
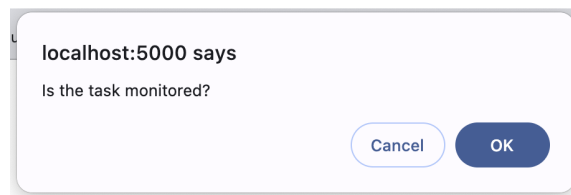


Figure 5.5: Ask User For If The Task is Monitored



5.4 Save Button Implementation

The save functionality in the BPMN modeler application is implemented through a simple HTML structure combined with JavaScript logic. This section will explore the key elements involved in this implementation.

HTML Structure

The HTML file that supports the BPMN modeler interface contains a setup designed to initialize and manage the BPMN modeler, along with a button to save the diagram. The main components of the HTML structure are as follows:

- **Container and Save Button:**

```
<div id="container">  
  <button id="save-button">Save Diagram</button>  
</div>
```

The `<div id="container">` element serves as the container for the BPMN modeler. Within this container, a `<button>` element is provided, labeled "Save Diagram". This button triggers the save functionality when clicked by the user.

- **Including External Libraries:**

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js"></script>  
<script src="app.js"></script>
```

External libraries such as `FileSaver.js` are included via `<script>` tags to provide additional functionalities like saving the diagram as a file. The `app.js` script contains the core logic of the BPMN modeler, including the interaction between the save button and the BPMN modeler instance.

JavaScript Logic

The `app.js` script orchestrates the BPMN modeler's behavior, including the save button's functionality. The key aspects of this implementation are:

- **BPMN Modeler Initialization:**

```
const bpmnModeler = new BpmnModeler({
  container: '#container',
  additionalModules: [ ... ],
  moddleExtensions: { sm: smExtension }
});
```

The BPMN modeler is instantiated with the `container` element, where the BPMN diagram is displayed and edited. Custom modules and extensions, such as `smExtension`, are included to enhance the modeler's functionality.

- **Save Button Event Listener:**

```
document.getElementById('save-button').addEventListener('click', (e) => {
  e.preventDefault();
  bpmnModeler.saveXML({ format: true }).then(result => {
    const blob = new Blob([result.xml], { type: 'application/xml' });
    saveAs(blob, 'diagram.bpmn');
  }).catch(err => console.error(err));
});
```

The `save-button` is linked to a click event listener that triggers the save process. When the button is clicked, the `saveXML` method of the BPMN modeler is called, which generates the XML representation of the current BPMN diagram. This XML data is then converted into a Blob and saved as a `.bpmn` file using the `FileSaver.js` library.

Figure 5.6: Example Saved XML File

```
<bpmn2:task id="Task_1" name="Examine Situation">
  <bpmn2:extensionElements>
    <sm:kei value="indoorEnergy" monitored="true" />
    <sm:indoorEnergy value="23" unit="kWh" monitored="true" />
  </bpmn2:extensionElements>
  <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
</bpmn2:task>
```

As seen in the figure above, the BPMN diagram saved includes an extension element in one of its tasks. As read, the KEI is indoor energy with the value 23 and unit kWh. The task is also monitored.

Chapter 6

Results

The results of this thesis are presented by referring back to both functional and non-functional requirements that were established in Section 3 of the paper. Each subsection below is a functional or non-functional requirement and describes whether this requirement has been met.

No.	Requirement	Status
FR1	The system shall extend BPMN to incorporate sustainability metrics by integrating a custom sustainability extension module (smExtension).	Met - The system successfully integrates sustainability metrics through smExtension, enabling the modeling of sustainability factors directly in BPMN.
FR2	The system shall allow the definition and assignment of Key Environmental Indicators (KEIs) to BPMN tasks and elements, including energy consumption, renewable energy, non-renewable energy, indoor energy, transportation energy, single source of energy, carbon dioxide emissions, water usage, and waste generation.	Met - Users can define and assign various KEIs, covering a wide range of environmental metrics.
FR3	The system shall provide a custom context pad that allows users to edit KEI properties for BPMN elements, including setting whether a task is measured or monitored.	Met - A custom context pad allows users to easily edit KEI properties, including monitoring settings.
FR4	The system shall ensure that KEI properties are preserved when BPMN tasks are replaced or modified, including the transfer of these properties to new or updated elements.	Met - KEI properties are preserved and transferred during task modifications or replacements.
FR5	The system shall render KEI values visually on BPMN elements, displaying associated icons, values, and monitoring status directly on the diagram.	Met - KEI values are visually rendered with icons and values on BPMN elements.
FR6	The system shall allow the user to edit KEI values through a custom user interface, including options to input values, units, and monitoring status for each KEI.	Met - A custom user interface enables editing of KEI values, units, and monitoring status.
FR7	The system shall update the BPMN XML to include KEI data as part of the extension elements, ensuring all sustainability metrics are stored in a standardized format.	Met - The system updates the BPMN XML to store KEI data in a standardized format.
FR8	The system shall enable the visualization of KEI changes in real-time, ensuring that any modifications to KEI properties are immediately reflected in the BPMN diagram.	Met - Real-time KEI changes are reflected immediately in the BPMN diagram.
FR9	The system shall ensure that when saving the BPMN diagram, all KEI properties and their values are accurately captured and saved within the BPMN XML structure.	Met - KEI properties and values are accurately saved within the BPMN XML structure when diagrams are saved.
FR10	The specification of sustainability metrics (KEIs) for BPMN elements shall be optional. Users can choose whether to assign KEIs to individual BPMN tasks and elements.	Met - KEIs are optional, allowing users to decide whether to assign them to BPMN elements.
NFR1	The visual representation of KEI data on BPMN elements should be clear and unobtrusive, ensuring that users can easily distinguish KEI information without overwhelming the BPMN diagram.	Met - KEI data is clearly and unobtrusively represented on BPMN elements without overwhelming the diagram.
NFR2	The KEI data and its visualization should be accessible and usable across different platforms and screen sizes, ensuring a consistent user experience in web-based environments.	Not fully met - While functional, the system currently runs in a localhost environment, limiting accessibility across different platforms.
NFR3	The sustainability extension should be documented thoroughly, providing clear guidance for developers who may need to extend BPMN in the future.	Met - The extension is well-documented, providing clear guidance for developers.
NFR4	The system should perform efficiently, maintaining responsiveness and speed during the editing and saving of BPMN diagrams with integrated KEI data.	Met - The system performs efficiently, maintaining responsiveness during the editing and saving of BPMN diagrams.

Table 6.1: Functional and Non-Functional Requirements with Status

Chapter 7

Conclusion

This thesis has tackled the challenge of integrating sustainability into Business Process Model and Notation (BPMN) by extending the BPMN 2.0 meta-model. In an era where businesses are increasingly expected to operate sustainably, the need to incorporate environmental metrics into core business operations is more critical than ever. The growing focus on sustainable development has prompted organizations to not only comply with regulations but also view sustainability as a strategic advantage, offering benefits such as enhanced reputation, operational efficiency, and new growth opportunities. However, despite the widespread adoption of BPMN as a standard for modeling business processes, it lacks built-in capabilities to represent sustainability metrics, thereby limiting businesses' ability to assess and improve the environmental impact of their processes.

This research was motivated by this gap. The introduction of sustainability metrics into BPMN models is essential for organizations seeking to minimize the environmental footprint of their operations and pursue continuous sustainability improvements. The thesis aimed to address this by developing a custom extension module (**smExtension**) that integrates Key Environmental Indicators (KEIs) directly into BPMN diagrams.

The initial phase of the research involved identifying the most relevant environmental sustainability metrics to incorporate into the BPMN framework, focusing on metrics such as energy consumption, renewable energy, non-renewable energy, water usage, waste generation, and carbon dioxide emissions. The BPMN 2.0 meta-model was then extended to accommodate these KEIs by modifying the XML Schema Definition (XSD) that underlies BPMN models. This extension ensured that the new sustainability-related elements and attributes were seamlessly integrated with existing BPMN components.

Custom renderers and context pads were developed to support the visualization and management of these KEIs within BPMN diagrams. These tools allow users to assign, edit, and visualize KEI values directly within BPMN elements, making it easier for users to assess the environmental impact of specific tasks and processes. The system also supports real-time visualization of KEI changes, ensuring that any modifications to sustainability metrics are immediately reflected in the BPMN diagrams. Additionally, the extended diagrams are stored in a standardized XML format, ensuring easy sharing and further analysis of sustainability data.

The practical application of this extended BPMN framework was validated through case studies, demonstrating the effectiveness of the **smExtension** in real-world scenarios. The results confirmed that the extension not only meets the functional and non-functional requirements but also enhances BPMN's capability to incorporate sustainability as a key dimension in business process management. This research has effectively laid the groundwork for further exploration of Green BPM and has shown that the integration of sustainability metrics into BPMN is both feasible and beneficial for organizations committed to sustainable operations.

7.1 Future Work

While this thesis has laid the groundwork for integrating sustainability into BPMN, there are several avenues for future research and development. One significant area for future work is the expansion of the sustainability categories beyond those initially implemented. The current system primarily focuses on energy-related metrics, but there is potential to extend the BPMN framework to include additional sustainability categories, such as waste, water, and emissions. This would

provide a more comprehensive tool for organizations aiming to achieve broader sustainability goals.

Another area for development is the assignment of multiple metrics to tasks. Currently, each task in the BPMN diagram can be associated with a single KEI. Expanding the system to allow for the assignment of multiple metrics to a single task would enable a more detailed and nuanced analysis of the environmental impact of complex processes. This enhancement would better reflect the interconnected nature of sustainability factors in real-world business scenarios.

Moreover, while the system functions well in its current form, future development could focus on improving the user interface to enhance usability and accessibility across different platforms and devices. This could include optimizing the system for mobile devices, ensuring that it can be used effectively in a variety of business contexts. Additionally, to maximize its utility, the extended BPMN framework could be integrated with other sustainability reporting tools and frameworks. This integration would enable organizations to have a more holistic view of their sustainability performance, combining process-level insights with broader corporate sustainability metrics.

Finally, the KEIs used in this research were chosen for their general applicability, but different industries may require tailored metrics that better reflect their unique sustainability challenges. Future work could involve customizing the BPMN extension for specific sectors, such as manufacturing, healthcare, or logistics, making the tool more relevant and impactful in different contexts. By addressing these areas, future work can build on the achievements of this thesis to further enhance the integration of sustainability into business process management, ultimately contributing to more sustainable and responsible business practices.

Bibliography

- [1] Michael E. Porter and Mark R. Kramer. Creating shared value. *Harvard Business Review*, 89 (1/2):62–77, 2011.
- [2] John Elkington. *Cannibals with Forks: The Triple Bottom Line of 21st Century Business*. Capstone Publishing, 1997.
- [3] Stefan Schaltegger and Roger Burritt. *Business Cases for Sustainability and the Role of Business Models: Developing a Conceptual Framework*. Routledge, 2015.
- [4] Jan C. Recker. Opportunities and constraints: the current struggle with bpmn. *Business Process Management Journal*, 16(1):181–201, 2010.
- [5] OMG. Business process model and notation, Dec 2010. URL <https://www.omg.org/spec/BPMN/2.0/About-BPMN>.
- [6] Jan Vom Brocke, Stefan Seidel, and Jan Recker. *Green Business Process Management: Towards the Sustainable Enterprise*. Springer, Berlin, Heidelberg, 2012.
- [7] Stefan Seidel, Jan C. Recker, and Jan Vom Brocke. Sensemaking and sustainable practicing: Functional affordances of information systems in green transformations. *MIS Quarterly*, 37 (4):1275–1299, 2017.
- [8] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.
- [9] H.A. Reijers and S.L. Mansar. Best practices in business process redesign: An overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306, 2005. doi: 10.1016/j.omega.2004.04.012.
- [10] Nigel P. Melville. Information systems innovation for environmental sustainability. *MIS Quarterly*, 34(1):1–21, 2010.
- [11] Mark von Rosing. Business process model and notation—bpmn, 2015. URL http://www.omg.org/news/whitepapers/Business_Process_Model_and_Notation.pdf.
- [12] Gustav Aagesen and John Krogstie. Bpmn 2.0 for modeling business processes. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, pages 219–250, 04 2015. doi: 10.1007/978-3-642-45100-3_10.
- [13] V. van den Broek. Going greener through bpm: a method for assessing processes environmental footprint and supporting continuous improvement, Aug 2015.
- [14] Lars Fritsch and Mathias Goeken. Towards a unified framework for measuring sustainability in business processes. *Journal of Business Process Management*, 28:25–38, 2022. doi: 10.1007/springer-example.
- [15] Nico Opitz, Jan Mendling, and Markus Nüttgens. Towards green business process management: A framework for modeling and analyzing eco-efficiency in business processes. *Journal of Business Process Management*, 20:223–240, 2014. doi: 10.1108/14637151411411056.
- [16] Jan Recker, Michael Rosemann, and Marta Indulska. How bpmn can be extended to support sustainability: A case for green bpm. *Journal of Business Process Management*, 17:477–493, 2011. doi: 10.1108/1463715111142810.

- [17] Constantin Houy, Peter Fettke, and Peter Loos. Sustainability in business process management research: A literature review. *Business Process Management Journal*, 18:866–884, 2012. doi: 10.1108/14637151211253724.
- [18] Koh Song Sang and Bo Zhou. Bpmn security extensions for healthcare process. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2340–2345, 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.346.
- [19] Alfonso Rodriguez, Eduardo Fernandez-Medina, and Mario Piattini. A bpmn extension for the modeling of security requirements in business processes. *IEICE TRANSACTIONS on Information and Systems*, E90-D(4):745–752, April 2007. ISSN 1745-1361. doi: 10.1093/ietisy/e90-d.4.745.

Chapter 8

Appendix

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema elementFormDefault="qualified" attributeFormDefault="
  unqualified"
  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.omg.org/spec/BPMN/20100524/MODEL">

  <xsd:element name="activity" type="tActivity"/>
  <xsd:complexType name="tActivity" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="tFlowNode">
        <xsd:sequence>
          <xsd:element ref="ioSpecification"
            minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="property"
            minOccurs="0" maxOccurs="
              unbounded"/>
          <xsd:element ref="
            dataInputAssociation" minOccurs
              ="0" maxOccurs="unbounded"/>
          <xsd:element ref="
            dataOutputAssociation"
            minOccurs="0" maxOccurs="
              unbounded"/>
          <xsd:element ref="resourceRole"
            minOccurs="0" maxOccurs="
              unbounded"/>
          <xsd:element ref="
            loopCharacteristics" minOccurs
              ="0"/>
        </xsd:sequence>
        <xsd:attribute name="isForCompensation"
          type="xsd:boolean" default="false"/>
        <xsd:attribute name="startQuantity" type="
          xsd:integer" default="1"/>
        <xsd:attribute name="completionQuantity"
          type="xsd:integer" default="1"/>
        <xsd:attribute name="default" type="xsd:
          IDREF" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xs:complexType name="SustainabilityMetricsType">
    <xs:sequence>
      <xs:element name="energyConsumption" type="MetricType" minOccurs
```

```

        ="0"/>
<xs:element name="renewableEnergy" type="MetricType" minOccurs
        ="0"/>
<xs:element name="nonRenewableEnergy" type="MetricType" minOccurs
        ="0"/>
<xs:element name="indoorEnergy" type="MetricType" minOccurs="0"/>
<xs:element name="transportationEnergy" type="MetricType" minOccurs
        ="0"/>
<xs:element name="singleSourceOfEnergy" type="MetricType" minOccurs
        ="0"/>
<xs:element name="wasteGeneration" type="MetricType" minOccurs
        ="0"/>
<xs:element name="waterUsage" type="MetricType" minOccurs="0"/>
<xs:element name="carbonDioxideEmissions" type="MetricType"
        minOccurs="0"/>
</xs:sequence>
<xs:attribute name="monitored" type="xs:boolean" default="false"/>
</xs:complexType>

<xs:complexType name="MetricType">
    <xs:simpleContent>
        <xs:extension base="xs:decimal">
            <xs:attribute name="unit" type="xs:string" use="required"/>
            <xs:attribute name="unknown" type="xs:boolean" default="
                false"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xsd:element name="adHocSubProcess" type="tAdHocSubProcess"
        substitutionGroup="flowElement"/>
<xsd:complexType name="tAdHocSubProcess">
    <xsd:complexContent>
        <xsd:extension base="tSubProcess">
            <xsd:sequence>
                <xsd:element name="
                    completionCondition" type="
                    tExpression" minOccurs="0"
                    maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="
                cancelRemainingInstances" type="xsd:
                boolean" default="true"/>
            <xsd:attribute name="ordering" type="
                tAdHocOrdering"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tAdHocOrdering">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Parallel"/>
        <xsd:enumeration value="Sequential"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="artifact" type="tArtifact"/>
<xsd:complexType name="tArtifact" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement"/>
    </xsd:complexContent>
</xsd:complexType>

```



```

<xsd:element name="assignment" type="tAssignment" />
<xsd:complexType name="tAssignment">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement">
      <xsd:sequence>
        <xsd:element name="from" type="tExpression" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="to" type="tExpression" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="association" type="tAssociation" substitutionGroup="artifact"/>
<xsd:complexType name="tAssociation">
  <xsd:complexContent>
    <xsd:extension base="tArtifact">
      <xsd:attribute name="sourceRef" type="xsd:QName" use="required"/>
      <xsd:attribute name="targetRef" type="xsd:QName" use="required"/>
      <xsd:attribute name="associationDirection" type="tAssociationDirection" default="None"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tAssociationDirection">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="None"/>
    <xsd:enumeration value="One"/>
    <xsd:enumeration value="Both"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="auditing" type="tAuditing"/>
<xsd:complexType name="tAuditing">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="baseElement" type="tBaseElement"/>
<xsd:complexType name="tBaseElement" abstract="true">
  <xsd:sequence>
    <xsd:element ref="documentation" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="extensionElements" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
  <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>

<xsd:element name="baseElementWithMixedContent" type="tBaseElementWithMixedContent"/>
<xsd:complexType name="tBaseElementWithMixedContent" abstract="true"

```

```

" mixed="true">
  <xsd:sequence>
    <xsd:element ref="documentation" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element ref="extensionElements" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
  <xsd:anyAttribute namespace="##other" processContents="lax"
    />
</xsd:complexType>

<xsd:element name="boundaryEvent" type="tBoundaryEvent"
  substitutionGroup="flowElement"/>
<xsd:complexType name="tBoundaryEvent">
  <xsd:complexContent>
    <xsd:extension base="tCatchEvent">
      <xsd:attribute name="cancelActivity" type="
        xsd:boolean" default="true"/>
      <xsd:attribute name="attachedToRef" type="
        xsd:QName" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="businessRuleTask" type="tBusinessRuleTask"
  substitutionGroup="flowElement"/>
<xsd:complexType name="tBusinessRuleTask">
  <xsd:complexContent>
    <xsd:extension base="tTask">
      <xsd:attribute name="implementation" type="
        tImplementation" default="##unspecified"
        />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="callableElement" type="tCallableElement"/>
<xsd:complexType name="tCallableElement">
  <xsd:complexContent>
    <xsd:extension base="tRootElement">
      <xsd:sequence>
        <xsd:element name="
          supportedInterfaceRef" type="
            xsd:QName" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element ref="ioSpecification"
          minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="ioBinding"
          minOccurs="0" maxOccurs="
            unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string"
        />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="callActivity" type="tCallActivity"
  substitutionGroup="flowElement"/>
<xsd:complexType name="tCallActivity">
  <xsd:complexContent>
    <xsd:extension base="tActivity">

```

```

        <xsd:attribute name="calledElement" type="
            xsd:QName" use="optional"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="callChoreography" type="tCallChoreography"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tCallChoreography">
    <xsd:complexContent>
        <xsd:extension base="tChoreographyActivity">
            <xsd:sequence>
                <xsd:element ref="
                    participantAssociation"
                    minOccurs="0" maxOccurs="
                        unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="calledChoreographyRef"
                type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="callConversation" type="tCallConversation"
    substitutionGroup="conversationNode"/>
<xsd:complexType name="tCallConversation">
    <xsd:complexContent>
        <xsd:extension base="tConversationNode">
            <xsd:sequence>
                <xsd:element ref="
                    participantAssociation"
                    minOccurs="0" maxOccurs="
                        unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="calledCollaborationRef"
                type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="cancelEventDefinition" type="
    tCancelEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tCancelEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="catchEvent" type="tCatchEvent"/>
<xsd:complexType name="tCatchEvent" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tEvent">
            <xsd:sequence>
                <xsd:element ref="dataOutput"
                    minOccurs="0" maxOccurs="
                        unbounded"/>
                <xsd:element ref="
                    dataOutputAssociation"
                    minOccurs="0" maxOccurs="
                        unbounded"/>
                <xsd:element ref="outputSet"
                    minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="eventDefinition"

```

```

        minOccurs="0" maxOccurs="
        unbounded"/>
        <xsd:element name="
        eventDefinitionRef" type="xsd:
        QName" minOccurs="0" maxOccurs
        ="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="parallelMultiple" type
    ="xsd:boolean" default="false"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="category" type="tCategory" substitutionGroup="
    rootElement"/>
<xsd:complexType name="tCategory">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element ref="categoryValue"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string
            "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="categoryValue" type="tCategoryValue"/>
<xsd:complexType name="tCategoryValue">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="value" type="xsd:
            string" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="choreography" type="tChoreography"
    substitutionGroup="collaboration"/>
<xsd:complexType name="tChoreography">
    <xsd:complexContent>
        <xsd:extension base="tCollaboration">
            <xsd:sequence>
                <xsd:element ref="flowElement"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="choreographyActivity" type="
    tChoreographyActivity"/>
<xsd:complexType name="tChoreographyActivity" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tFlowNode">
            <xsd:sequence>
                <xsd:element name="participantRef"
                    type="xsd:QName" minOccurs="2"
                    maxOccurs="unbounded"/>
                <xsd:element ref="correlationKey"

```

```

minOccurs="0" maxOccurs="
unbounded"/>
</xsd:sequence>
<xsd:attribute name="
initiatingParticipantRef" type="xsd:
QName" use="required"/>
<xsd:attribute name="loopType" type="
tChoreographyLoopType" default="None"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tChoreographyLoopType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="None"/>
<xsd:enumeration value="Standard"/>
<xsd:enumeration value="MultiInstanceSequential"/>
<xsd:enumeration value="MultiInstanceParallel"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:element name="choreographyTask" type="tChoreographyTask"
substitutionGroup="flowElement"/>
<xsd:complexType name="tChoreographyTask">
<xsd:complexContent>
<xsd:extension base="tChoreographyActivity">
<xsd:sequence>
<xsd:element name="messageFlowRef"
type="xsd:QName" minOccurs="1"
maxOccurs="2"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="collaboration" type="tCollaboration"
substitutionGroup="rootElement"/>
<xsd:complexType name="tCollaboration">
<xsd:complexContent>
<xsd:extension base="tRootElement">
<xsd:sequence>
<xsd:element ref="participant"
minOccurs="0" maxOccurs="
unbounded"/>
<xsd:element ref="messageFlow"
minOccurs="0" maxOccurs="
unbounded"/>
<xsd:element ref="artifact"
minOccurs="0" maxOccurs="
unbounded"/>
<xsd:element ref="conversationNode"
minOccurs="0" maxOccurs="
unbounded"/>
<xsd:element ref="
conversationAssociation"
minOccurs="0" maxOccurs="
unbounded"/>
<xsd:element ref="
participantAssociation"
minOccurs="0" maxOccurs="
unbounded"/>
<xsd:element ref="
messageFlowAssociation"

```

```

        minOccurs="0" maxOccurs="
        unbounded"/>
        <xsd:element ref="correlationKey"
        minOccurs="0" maxOccurs="
        unbounded"/>
        <xsd:element name="choreographyRef"
        type="xsd:QName" minOccurs="0"
        maxOccurs="unbounded"/>
        <xsd:element ref="conversationLink"
        minOccurs="0" maxOccurs="
        unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string
    "/>
    <xsd:attribute name="isClosed" type="xsd:
    boolean" default="false"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="compensateEventDefinition" type="
    tCompensateEventDefinition" substitutionGroup="eventDefinition
    "/>
<xsd:complexType name="tCompensateEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition">
            <xsd:attribute name="waitForCompletion"
            type="xsd:boolean"/>
            <xsd:attribute name="activityRef" type="xsd:
            QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="complexBehaviorDefinition" type="
    tComplexBehaviorDefinition"/>
<xsd:complexType name="tComplexBehaviorDefinition">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="condition" type="
                tFormalExpression" minOccurs
                ="1" maxOccurs="1"/>
                <xsd:element name="event" type="
                tImplicitThrowEvent" minOccurs
                ="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="complexGateway" type="tComplexGateway"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tComplexGateway">
    <xsd:complexContent>
        <xsd:extension base="tGateway">
            <xsd:sequence>
                <xsd:element name="
                activationCondition" type="
                tExpression" minOccurs="0"
                maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="default" type="xsd:

```

```

IDREF"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="conditionalEventDefinition" type="
  tConditionalEventDefinition" substitutionGroup="eventDefinition
"/>
<xsd:complexType name="tConditionalEventDefinition">
  <xsd:complexContent>
    <xsd:extension base="tEventDefinition">
      <xsd:sequence>
        <xsd:element name="condition" type="
          tExpression"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="conversation" type="tConversation"
  substitutionGroup="conversationNode"/>
<xsd:complexType name="tConversation">
  <xsd:complexContent>
    <xsd:extension base="tConversationNode"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="conversationAssociation" type="
  tConversationAssociation"/>
<xsd:complexType name="tConversationAssociation">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement">
      <xsd:attribute name="
        innerConversationNodeRef" type="xsd:
        QName" use="required"/>
      <xsd:attribute name="
        outerConversationNodeRef" type="xsd:
        QName" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="conversationLink" type="tConversationLink"/>
<xsd:complexType name="tConversationLink">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement">
      <xsd:attribute name="name" type="xsd:string"
        use="optional"/>
      <xsd:attribute name="sourceRef" type="xsd:
        QName" use="required"/>
      <xsd:attribute name="targetRef" type="xsd:
        QName" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="conversationNode" type="tConversationNode"/>
<xsd:complexType name="tConversationNode" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement">
      <xsd:sequence>
        <xsd:element name="participantRef"
          type="xsd:QName" minOccurs="0"

```

```

        maxOccurs="unbounded"/>
        <xsd:element name="messageFlowRef"
            type="xsd:QName" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:element ref="correlationKey"
            minOccurs="0" maxOccurs="
            unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string
        "/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="correlationKey" type="tCorrelationKey"/>
<xsd:complexType name="tCorrelationKey">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="
                    correlationPropertyRef" type="
                    xsd:QName" minOccurs="0"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string
                " use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="correlationProperty" type="tCorrelationProperty"
    substitutionGroup="rootElement"/>
<xsd:complexType name="tCorrelationProperty">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element ref="
                    correlationPropertyRetrievalExpression
                    " minOccurs="1" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string
                " use="optional"/>
            <xsd:attribute name="type" type="xsd:QName
                "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="correlationPropertyBinding" type="
    tCorrelationPropertyBinding"/>
<xsd:complexType name="tCorrelationPropertyBinding">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="dataPath" type="
                    tFormalExpression" minOccurs
                    ="1" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="correlationPropertyRef
                " type="xsd:QName" use="required"/>
        </xsd:extension>
    </xsd:complexContent>

```



```

</xsd:complexType>

<xsd:element name="correlationPropertyRetrievalExpression" type="
    tCorrelationPropertyRetrievalExpression"/>
<xsd:complexType name="tCorrelationPropertyRetrievalExpression">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="messagePath"
                    type="tFormalExpression"
                    minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="messageRef" type="xsd:
                QName" use="required"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="correlationSubscription" type="
    tCorrelationSubscription"/>
<xsd:complexType name="tCorrelationSubscription">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="
                    correlationPropertyBinding"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="correlationKeyRef"
                type="xsd:QName" use="required"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataAssociation" type="tDataAssociation" />
<xsd:complexType name="tDataAssociation">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="sourceRef" type
                    ="xsd:IDREF" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:element name="targetRef" type
                    ="xsd:IDREF" minOccurs="1"
                    maxOccurs="1"/>
                <xsd:element name="transformation"
                    type="tFormalExpression"
                    minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="assignment"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataInput" type="tDataInput" />
<xsd:complexType name="tDataInput">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="dataState"

```

```

                                minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"
            " use="optional"/>
        <xsd:attribute name="itemSubjectRef" type="
            xsd:QName" />
        <xsd:attribute name="isCollection" type="
            xsd:boolean" default="false"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataInputAssociation" type="
    tDataInputAssociation" />
<xsd:complexType name="tDataInputAssociation">
    <xsd:complexContent>
        <xsd:extension base="tDataAssociation"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataObject" type="tDataObject" substitutionGroup
    ="flowElement"/>
<xsd:complexType name="tDataObject">
    <xsd:complexContent>
        <xsd:extension base="tFlowElement">
            <xsd:sequence>
                <xsd:element ref="dataState"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="itemSubjectRef" type="
                xsd:QName"/>
            <xsd:attribute name="isCollection" type="
                xsd:boolean" default="false"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataObjectReference" type="tDataObjectReference"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tDataObjectReference">
    <xsd:complexContent>
        <xsd:extension base="tFlowElement">
            <xsd:sequence>
                <xsd:element ref="dataState"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="itemSubjectRef" type="
                xsd:QName"/>
            <xsd:attribute name="dataObjectRef" type="
                xsd:IDREF"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataOutput" type="tDataOutput" />
<xsd:complexType name="tDataOutput">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="dataState"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string

```

```

        " use="optional" />
        <xsd:attribute name="itemSubjectRef" type="
            xsd:QName"/>
        <xsd:attribute name="isCollection" type="
            xsd:boolean" default="false"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataOutputAssociation" type="
    tDataOutputAssociation" />
<xsd:complexType name="tDataOutputAssociation">
    <xsd:complexContent>
        <xsd:extension base="tDataAssociation"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataState" type="tDataState" />
<xsd:complexType name="tDataState">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string"
                "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataStore" type="tDataStore" substitutionGroup="
    rootElement"/>
<xsd:complexType name="tDataStore">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element ref="dataState"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                "/>
            <xsd:attribute name="capacity" type="xsd:
                integer"/>
            <xsd:attribute name="isUnlimited" type="xsd:
                boolean" default="true"/>
            <xsd:attribute name="itemSubjectRef" type="
                xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dataStoreReference" type="tDataStoreReference"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tDataStoreReference">
    <xsd:complexContent>
        <xsd:extension base="tFlowElement">
            <xsd:sequence>
                <xsd:element ref="dataState"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="itemSubjectRef" type="
                xsd:QName"/>
            <xsd:attribute name="dataStoreRef" type="
                xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>

```

```

</xsd:complexType>

<xsd:element name="documentation" type="tDocumentation"/>
<xsd:complexType name="tDocumentation" mixed="true">
  <xsd:sequence>
    <xsd:any namespace="##any" processContents="lax"
      minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
  <xsd:attribute name="textFormat" type="xsd:string" default
    ="text/plain"/>
</xsd:complexType>

<xsd:element name="endEvent" type="tEndEvent" substitutionGroup="
  flowElement"/>
<xsd:complexType name="tEndEvent">
  <xsd:complexContent>
    <xsd:extension base="tThrowEvent"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="endPoint" type="tEndPoint" substitutionGroup="
  rootElement"/>
<xsd:complexType name="tEndPoint">
  <xsd:complexContent>
    <xsd:extension base="tRootElement"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="error" type="tError" substitutionGroup="
  rootElement"/>
<xsd:complexType name="tError">
  <xsd:complexContent>
    <xsd:extension base="tRootElement">
      <xsd:attribute name="name" type="xsd:string"
        />
      <xsd:attribute name="errorCode" type="xsd:
        string"/>
      <xsd:attribute name="structureRef" type="
        xsd:QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="errorEventDefinition" type="
  tErrorEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tErrorEventDefinition">
  <xsd:complexContent>
    <xsd:extension base="tEventDefinition">
      <xsd:attribute name="errorRef" type="xsd:
        QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="escalation" type="tEscalation" substitutionGroup
  ="rootElement"/>
<xsd:complexType name="tEscalation">
  <xsd:complexContent>
    <xsd:extension base="tRootElement">
      <xsd:attribute name="name" type="xsd:string"
        />
      <xsd:attribute name="escalationCode" type="

```

```

        xsd:string"/>
        <xsd:attribute name="structureRef" type="
            xsd:QName"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="escalationEventDefinition" type="
    tEscalationEventDefinition" substitutionGroup="eventDefinition
"/>
<xsd:complexType name="tEscalationEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition">
            <xsd:attribute name="escalationRef" type="
                xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="event" type="tEvent" substitutionGroup="
    flowElement"/>
<xsd:complexType name="tEvent" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tFlowNode">
            <xsd:sequence>
                <xsd:element ref="property"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="eventBasedGateway" type="tEventBasedGateway"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tEventBasedGateway">
    <xsd:complexContent>
        <xsd:extension base="tGateway">
            <xsd:attribute name="instantiate" type="xsd
                :boolean" default="false"/>
            <xsd:attribute name="eventGatewayType" type
                ="tEventBasedGatewayType" default="
                Exclusive"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tEventBasedGatewayType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Exclusive"/>
        <xsd:enumeration value="Parallel"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="eventDefinition" type="tEventDefinition"
    substitutionGroup="rootElement"/>
<xsd:complexType name="tEventDefinition" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tRootElement"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="exclusiveGateway" type="tExclusiveGateway"

```

```

        substitutionGroup="flowElement"/>
<xsd:complexType name="tExclusiveGateway">
    <xsd:complexContent>
        <xsd:extension base="tGateway">
            <xsd:attribute name="default" type="xsd:
                IDREF" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="expression" type="tExpression"/>
<xsd:complexType name="tExpression">
    <xsd:complexContent>
        <xsd:extension base="tBaseElementWithMixedContent
            "/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="extension" type="tExtension"/>
<xsd:complexType name="tExtension">
    <xsd:sequence>
        <xsd:element ref="documentation" minOccurs="0"
            maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="definition" type="xsd:QName"/>
    <xsd:attribute name="mustUnderstand" type="xsd:boolean" use
        ="optional" default="false"/>
</xsd:complexType>

<xsd:element name="extensionElements" type="tExtensionElements" />
<xsd:complexType name="tExtensionElements">
    <xsd:sequence>
        <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="flowElement" type="tFlowElement"/>
<xsd:complexType name="tFlowElement" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="auditing"
                    minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="monitoring"
                    minOccurs="0" maxOccurs="1"/>
                <xsd:element name="categoryValueRef"
                    type="xsd:QName" minOccurs
                    ="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string
                "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="flowNode" type="tFlowNode"/>
<xsd:complexType name="tFlowNode" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tFlowElement">
            <xsd:sequence>
                <xsd:element name="incoming" type="
                    xsd:QName" minOccurs="0"

```

```

maxOccurs="unbounded"/>
<xsd:element name="outgoing" type="
  xsd:QName" minOccurs="0"
  maxOccurs="unbounded"/>
<xs:element name="sustainabilityMetrics" type="
  SustainabilityMetricsType" minOccurs="0"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="formalExpression" type="tFormalExpression"
  substitutionGroup="expression"/>
<xsd:complexType name="tFormalExpression">
  <xsd:complexContent>
    <xsd:extension base="tExpression">
      <xsd:attribute name="language" type="xsd:
        anyURI" use="optional"/>
      <xsd:attribute name="evaluatesToTypeRef"
        type="xsd:QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="gateway" type="tGateway" abstract="true"/>
<xsd:complexType name="tGateway">
  <xsd:complexContent>
    <xsd:extension base="tFlowNode">
      <xsd:sequence>
        <!-- New sustainability metrics -->
        <xsd:element name="sustainabilityMetrics" type="
          SustainabilityMetricsType" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="gatewayDirection" type="tGatewayDirection"
        default="Unspecified"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tGatewayDirection">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Unspecified"/>
    <xsd:enumeration value="Converging"/>
    <xsd:enumeration value="Diverging"/>
    <xsd:enumeration value="Mixed"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="globalBusinessRuleTask" type="
  tGlobalBusinessRuleTask" substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalBusinessRuleTask">
  <xsd:complexContent>
    <xsd:extension base="tGlobalTask">
      <xsd:attribute name="implementation" type="
        tImplementation" default="##unspecified"
        "/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="globalChoreographyTask" type="
  tGlobalChoreographyTask" substitutionGroup="choreography"/>

```

```

<xsd:complexType name="tGlobalChoreographyTask">
  <xsd:complexContent>
    <xsd:extension base="tChoreography">
      <xsd:attribute name="
        initiatingParticipantRef" type="xsd:
        QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="globalConversation" type="tGlobalConversation"
  substitutionGroup="collaboration"/>
<xsd:complexType name="tGlobalConversation">
  <xsd:complexContent>
    <xsd:extension base="tCollaboration"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="globalManualTask" type="tGlobalManualTask"
  substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalManualTask">
  <xsd:complexContent>
    <xsd:extension base="tGlobalTask"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="globalScriptTask" type="tGlobalScriptTask"
  substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalScriptTask">
  <xsd:complexContent>
    <xsd:extension base="tGlobalTask">
      <xsd:sequence>
        <xsd:element ref="script" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="scriptLanguage" type="
        xsd:anyURI"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="globalTask" type="tGlobalTask" substitutionGroup
  ="rootElement"/>
<xsd:complexType name="tGlobalTask">
  <xsd:complexContent>
    <xsd:extension base="tCallableElement">
      <xsd:sequence>
        <xsd:element ref="resourceRole"
          minOccurs="0" maxOccurs="
          unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="globalUserTask" type="tGlobalUserTask"
  substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalUserTask">
  <xsd:complexContent>
    <xsd:extension base="tGlobalTask">
      <xsd:sequence>
        <xsd:element ref="rendering"
          minOccurs="0" maxOccurs="
          unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```



```

        </xsd:sequence>
        <xsd:attribute name="implementation" type="
            tImplementation" default="##unspecified
            "/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="group" type="tGroup" substitutionGroup="artifact
    "/>
<xsd:complexType name="tGroup">
    <xsd:complexContent>
        <xsd:extension base="tArtifact">
            <xsd:attribute name="categoryValueRef" type=
                "xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="humanPerformer" type="tHumanPerformer"
    substitutionGroup="performer"/>
<xsd:complexType name="tHumanPerformer">
    <xsd:complexContent>
        <xsd:extension base="tPerformer"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tImplementation">
    <xsd:union memberTypes="xsd:anyURI">
        <xsd:simpleType>
            <xsd:restriction base="xsd:token">
                <xsd:enumeration value="##
                    unspecified" />
                <xsd:enumeration value="##
                    WebService" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:union>
</xsd:simpleType>

<xsd:element name="implicitThrowEvent" type="tImplicitThrowEvent"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tImplicitThrowEvent">
    <xsd:complexContent>
        <xsd:extension base="tThrowEvent"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="inclusiveGateway" type="tInclusiveGateway"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tInclusiveGateway">
    <xsd:complexContent>
        <xsd:extension base="tGateway">
            <xsd:attribute name="default" type="xsd:
                IDREF" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="inputSet" type="tInputSet" />
<xsd:complexType name="tInputSet">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">

```

```

        <xsd:sequence>
            <xsd:element name="dataInputRefs"
                type="xsd:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="
                optionalInputRefs" type="xsd:
                IDREF" minOccurs="0" maxOccurs
                ="unbounded"/>
            <xsd:element name="
                whileExecutingInputRefs" type="
                xsd:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="outputSetRefs"
                type="xsd:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string
            " />
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="interface" type="tInterface" substitutionGroup="
    rootElement"/>
<xsd:complexType name="tInterface">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element ref="operation"
                    minOccurs="1" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string
                " use="required"/>
            <xsd:attribute name="implementationRef"
                type="xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="intermediateCatchEvent" type="
    tIntermediateCatchEvent" substitutionGroup="flowElement"/>
<xsd:complexType name="tIntermediateCatchEvent">
    <xsd:complexContent>
        <xsd:extension base="tCatchEvent"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="intermediateThrowEvent" type="
    tIntermediateThrowEvent" substitutionGroup="flowElement"/>
<xsd:complexType name="tIntermediateThrowEvent">
    <xsd:complexContent>
        <xsd:extension base="tThrowEvent"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ioBinding" type="tInputOutputBinding" />
<xsd:complexType name="tInputOutputBinding">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="operationRef" type="
                xsd:QName" use="required"/>
            <xsd:attribute name="inputDataRef" type="

```

```

        xsd:IDREF" use="required"/>
        <xsd:attribute name="outputDataRef" type="
        xsd:IDREF" use="required"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="ioSpecification" type="tInputOutputSpecification
" />
<xsd:complexType name="tInputOutputSpecification">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="dataInput"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="dataOutput"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="inputSet"
                    minOccurs="1" maxOccurs="
                    unbounded"/>
                <xsd:element ref="outputSet"
                    minOccurs="1" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="itemDefinition" type="tItemDefinition"
    substitutionGroup="rootElement"/>
<xsd:complexType name="tItemDefinition">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:attribute name="structureRef" type="
            xsd:QName"/>
            <xsd:attribute name="isCollection" type="
            xsd:boolean" default="false"/>
            <xsd:attribute name="itemKind" type="
            tItemKind" default="Information"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tItemKind">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Information"/>
        <xsd:enumeration value="Physical"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="lane" type="tLane"/>
<xsd:complexType name="tLane">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="partitionElement
                    " type="tBaseElement" minOccurs
                    ="0" maxOccurs="1"/>
                <xsd:element name="flowNodeRef"
                    type="xsd:IDREF" minOccurs="0"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

        <xsd:element name="childLaneSet"
            type="tLaneSet" minOccurs="0"
            maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"
        "/>
    <xsd:attribute name="partitionElementRef"
        type="xsd:QName"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="laneSet" type="tLaneSet"/>
<xsd:complexType name="tLaneSet">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="lane" minOccurs="0"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="linkEventDefinition" type="tLinkEventDefinition"
    substitutionGroup="eventDefinition"/>
<xsd:complexType name="tLinkEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition">
            <xsd:sequence>
                <xsd:element name="source" type="
                    xsd:QName" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:element name="target" type="
                    xsd:QName" minOccurs="0"
                    maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                " use="required"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="loopCharacteristics" type="tLoopCharacteristics"
    "/>
<xsd:complexType name="tLoopCharacteristics" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="manualTask" type="tManualTask" substitutionGroup
    ="flowElement"/>
<xsd:complexType name="tManualTask">
    <xsd:complexContent>
        <xsd:extension base="tTask"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="message" type="tMessage" substitutionGroup="
    rootElement"/>

```

```

<xsd:complexType name="tMessage">
  <xsd:complexContent>
    <xsd:extension base="tRootElement">
      <xsd:attribute name="name" type="xsd:string"
        "/>
      <xsd:attribute name="itemRef" type="xsd:
        QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="messageEventDefinition" type="
  tMessageEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tMessageEventDefinition">
  <xsd:complexContent>
    <xsd:extension base="tEventDefinition">
      <xsd:sequence>
        <xsd:element name="operationRef"
          type="xsd:QName" minOccurs="0"
          maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="messageRef" type="xsd:
        QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="messageFlow" type="tMessageFlow"/>
<xsd:complexType name="tMessageFlow">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement">
      <xsd:attribute name="name" type="xsd:string"
        use="optional"/>
      <xsd:attribute name="sourceRef" type="xsd:
        QName" use="required"/>
      <xsd:attribute name="targetRef" type="xsd:
        QName" use="required"/>
      <xsd:attribute name="messageRef" type="xsd:
        QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="messageFlowAssociation" type="
  tMessageFlowAssociation"/>
<xsd:complexType name="tMessageFlowAssociation">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement">
      <xsd:attribute name="innerMessageFlowRef"
        type="xsd:QName" use="required"/>
      <xsd:attribute name="outerMessageFlowRef"
        type="xsd:QName" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="monitoring" type="tMonitoring"/>
<xsd:complexType name="tMonitoring">
  <xsd:complexContent>
    <xsd:extension base="tBaseElement"/>
  </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:element name="multiInstanceLoopCharacteristics" type="
    tMultiInstanceLoopCharacteristics" substitutionGroup="
    loopCharacteristics"/>
<xsd:complexType name="tMultiInstanceLoopCharacteristics">
    <xsd:complexContent>
        <xsd:extension base="tLoopCharacteristics">
            <xsd:sequence>
                <xsd:element name="loopCardinality"
                    type="tExpression" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="loopDataInputRef"
                    type="xsd:QName" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="loopDataOutputRef" type="xsd:
                    QName" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="inputDataItem"
                    type="tDataInput" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="outputDataItem"
                    type="tDataOutput" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="
                    complexBehaviorDefinition"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element name="
                    completionCondition" type="
                    tExpression" minOccurs="0"
                    maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="isSequential" type="
                xsd:boolean" default="false"/>
            <xsd:attribute name="behavior" type="
                tMultiInstanceFlowCondition" default="
                All"/>
            <xsd:attribute name="oneBehaviorEventRef"
                type="xsd:QName" use="optional"/>
            <xsd:attribute name="noneBehaviorEventRef"
                type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tMultiInstanceFlowCondition">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="None"/>
        <xsd:enumeration value="One"/>
        <xsd:enumeration value="All"/>
        <xsd:enumeration value="Complex"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="operation" type="tOperation"/>
<xsd:complexType name="tOperation">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="inMessageRef"
                    type="xsd:QName" minOccurs="1"
                    maxOccurs="1"/>
                <xsd:element name="outMessageRef"

```

```

        type="xsd:QName" minOccurs="0"
        maxOccurs="1"/>
        <xsd:element name="errorRef" type="
        xsd:QName" minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string
    " use="required"/>
    <xsd:attribute name="implementationRef"
    type="xsd:QName"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="outputSet" type="tOutputSet" />
<xsd:complexType name="tOutputSet">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="dataOutputRefs"
                type="xsd:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
                <xsd:element name="
                optionalOutputRefs" type="xsd:
                IDREF" minOccurs="0" maxOccurs
                ="unbounded"/>
                <xsd:element name="
                whileExecutingOutputRefs" type
                ="xsd:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
                <xsd:element name="inputSetRefs"
                type="xsd:IDREF" minOccurs="0"
                maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string
            "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="parallelGateway" type="tParallelGateway"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tParallelGateway">
    <xsd:complexContent>
        <xsd:extension base="tGateway"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="participant" type="tParticipant"/>
<xsd:complexType name="tParticipant">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="interfaceRef"
                type="xsd:QName" minOccurs="0"
                maxOccurs="unbounded"/>
                <xsd:element name="endPointRef"
                type="xsd:QName" minOccurs="0"
                maxOccurs="unbounded"/>
                <xsd:element ref="
                participantMultiplicity"
                minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>

```

```

        <xsd:attribute name="name" type="xsd:string"
            "/>
        <xsd:attribute name="processRef" type="xsd:
            QName" use="optional"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="participantAssociation" type="
    tParticipantAssociation"/>
<xsd:complexType name="tParticipantAssociation">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="
                    innerParticipantRef" type="xsd:
                        QName" minOccurs="1" maxOccurs
                        ="1"/>
                <xsd:element name="
                    outerParticipantRef" type="xsd:
                        QName" minOccurs="1" maxOccurs
                        ="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="participantMultiplicity" type="
    tParticipantMultiplicity"/>
<xsd:complexType name="tParticipantMultiplicity">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="minimum" type="xsd:int
                " default="0"/>
            <xsd:attribute name="maximum" type="xsd:int
                " default="1"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="partnerEntity" type="tPartnerEntity"
    substitutionGroup="rootElement"/>
<xsd:complexType name="tPartnerEntity">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element name="participantRef"
                    type="xsd:QName" minOccurs="0"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="partnerRole" type="tPartnerRole"
    substitutionGroup="rootElement"/>
<xsd:complexType name="tPartnerRole">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element name="participantRef"

```



```

                                type="xsd:QName" minOccurs="0"
                                maxOccurs="unbounded"/>
                            </xsd:sequence>
                            <xsd:attribute name="name" type="xsd:string"
                                "/>
                        </xsd:extension>
                    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="performer" type="tPerformer" substitutionGroup="
    resourceRole"/>
<xsd:complexType name="tPerformer">
    <xsd:complexContent>
        <xsd:extension base="tResourceRole"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="potentialOwner" type="tPotentialOwner"
    substitutionGroup="performer"/>
<xsd:complexType name="tPotentialOwner">
    <xsd:complexContent>
        <xsd:extension base="tHumanPerformer"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="process" type="tProcess" substitutionGroup="
    rootElement"/>
<xsd:complexType name="tProcess">
    <xsd:complexContent>
        <xsd:extension base="tCallableElement">
            <xsd:sequence>
                <xsd:element ref="auditing"
                    minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="monitoring"
                    minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="property"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="laneSet"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="flowElement"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="artifact"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="resourceRole"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element ref="
                    correlationSubscription"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
                <xsd:element name="supports" type="
                    xsd:QName" minOccurs="0"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="processType" type="
                tProcessType" default="None"/>
            <xsd:attribute name="isClosed" type="xsd:
                boolean" default="false"/>
            <xsd:attribute name="isExecutable" type="

```

```

        xsd:boolean"/>
        <xsd:attribute name="
            definitionalCollaborationRef" type="xsd
                :QName" use="optional"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tProcessType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="None"/>
        <xsd:enumeration value="Public"/>
        <xsd:enumeration value="Private"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="property" type="tProperty" />
<xsd:complexType name="tProperty">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="dataState"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                />
            <xsd:attribute name="itemSubjectRef" type="
                xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="receiveTask" type="tReceiveTask"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tReceiveTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:attribute name="implementation" type="
                tImplementation" default="##WebService"
                />
            <xsd:attribute name="instantiate" type="xsd
                :boolean" default="false"/>
            <xsd:attribute name="messageRef" type="xsd:
                QName" use="optional"/>
            <xsd:attribute name="operationRef" type="
                xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="relationship" type="tRelationship"/>
<xsd:complexType name="tRelationship">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="source" type="
                    xsd:QName" minOccurs="1"
                    maxOccurs="unbounded"/>
                <xsd:element name="target" type="
                    xsd:QName" minOccurs="1"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="type" type="xsd:string

```

```

        " use="required"/>
        <xsd:attribute name="direction" type="
            tRelationshipDirection"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tRelationshipDirection">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="None"/>
        <xsd:enumeration value="Forward"/>
        <xsd:enumeration value="Backward"/>
        <xsd:enumeration value="Both"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="rendering" type="tRendering"/>
<xsd:complexType name="tRendering">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="resource" type="tResource" substitutionGroup="
    rootElement"/>
<xsd:complexType name="tResource">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element ref="resourceParameter"
                    " minOccurs="0" maxOccurs="
                        unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
                " use="required"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="resourceAssignmentExpression" type="
    tResourceAssignmentExpression"/>
<xsd:complexType name="tResourceAssignmentExpression">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="expression"
                    minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="resourceParameter" type="tResourceParameter"/>
<xsd:complexType name="tResourceParameter">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string"
                "/>
            <xsd:attribute name="type" type="xsd:QName"
                "/>
            <xsd:attribute name="isRequired" type="xsd:
                boolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="resourceParameterBinding" type="
        tResourceParameterBinding"/>
    <xsd:complexType name="tResourceParameterBinding">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element ref="expression"
                        minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute name="parameterRef" type="
                    xsd:QName" use="required"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="resourceRole" type="tResourceRole"/>
    <xsd:complexType name="tResourceRole">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:choice>
                    <xsd:sequence>
                        <xsd:element name="
                            resourceRef" type="xsd:
                                QName"/>
                        <xsd:element ref="
                            resourceParameterBinding
                                " minOccurs="0"
                                    maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:element ref="
                        resourceAssignmentExpression"
                            minOccurs="0" maxOccurs="1"/>
                </xsd:choice>
                <xsd:attribute name="name" type="xsd:string"
                    "/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="rootElement" type="tRootElement"/>
    <xsd:complexType name="tRootElement" abstract="true">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="scriptTask" type="tScriptTask" substitutionGroup
        ="flowElement"/>
    <xsd:complexType name="tScriptTask">
        <xsd:complexContent>
            <xsd:extension base="tTask">
                <xsd:sequence>
                    <xsd:element ref="script" minOccurs
                        ="0" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute name="scriptFormat" type="
                    xsd:string"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

```

```

<xsd:element name="script" type="tScript"/>
<xsd:complexType name="tScript" mixed="true">
    <xsd:sequence>
        <xsd:any namespace="##any" processContents="lax"
            minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="sendTask" type="tSendTask" substitutionGroup="
    flowElement"/>
<xsd:complexType name="tSendTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:attribute name="implementation" type="
                tImplementation" default="##WebService"
                />
            <xsd:attribute name="messageRef" type="xsd:
                QName" use="optional"/>
            <xsd:attribute name="operationRef" type="
                xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="sequenceFlow" type="tSequenceFlow"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tSequenceFlow">
    <xsd:complexContent>
        <xsd:extension base="tFlowElement">
            <xsd:sequence>
                <xsd:element name="
                    conditionExpression" type="
                    tExpression" minOccurs="0"
                    maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="sourceRef" type="xsd:
                IDREF" use="required"/>
            <xsd:attribute name="targetRef" type="xsd:
                IDREF" use="required"/>
            <xsd:attribute name="isImmediate" type="xsd:
                boolean" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="serviceTask" type="tServiceTask"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tServiceTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:attribute name="implementation" type="
                tImplementation" default="##WebService"
                />
            <xsd:attribute name="operationRef" type="
                xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="signal" type="tSignal" substitutionGroup="
    rootElement"/>
<xsd:complexType name="tSignal">

```

```

        <xsd:complexContent>
            <xsd:extension base="tRootElement">
                <xsd:attribute name="name" type="xsd:string"
                    "/>
                <xsd:attribute name="structureRef" type="
                    xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="signalEventDefinition" type="
        tSignalEventDefinition" substitutionGroup="eventDefinition"/>
    <xsd:complexType name="tSignalEventDefinition">
        <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                <xsd:attribute name="signalRef" type="xsd:
                    QName"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="standardLoopCharacteristics" type="
        tStandardLoopCharacteristics" substitutionGroup="
        loopCharacteristics"/>
    <xsd:complexType name="tStandardLoopCharacteristics">
        <xsd:complexContent>
            <xsd:extension base="tLoopCharacteristics">
                <xsd:sequence>
                    <xsd:element name="loopCondition"
                        type="tExpression" minOccurs
                        ="0"/>
                </xsd:sequence>
                <xsd:attribute name="testBefore" type="xsd:
                    boolean" default="false"/>
                <xsd:attribute name="loopMaximum" type="xsd:
                    integer" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="startEvent" type="tStartEvent" substitutionGroup
        ="flowElement"/>
    <xsd:complexType name="tStartEvent">
        <xsd:complexContent>
            <xsd:extension base="tCatchEvent">
                <xsd:attribute name="isInterrupting" type="
                    xsd:boolean" default="true"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="subChoreography" type="tSubChoreography"
        substitutionGroup="flowElement"/>
    <xsd:complexType name="tSubChoreography">
        <xsd:complexContent>
            <xsd:extension base="tChoreographyActivity">
                <xsd:sequence>
                    <xsd:element ref="flowElement"
                        minOccurs="0" maxOccurs="
                        unbounded"/>
                    <xsd:element ref="artifact"
                        minOccurs="0" maxOccurs="
                        unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

```

```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="subConversation" type="tSubConversation"
    substitutionGroup="conversationNode"/>
<xsd:complexType name="tSubConversation">
    <xsd:complexContent>
        <xsd:extension base="tConversationNode">
            <xsd:sequence>
                <xsd:element ref="conversationNode"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="subProcess" type="tSubProcess" substitutionGroup
    ="flowElement"/>
<xsd:complexType name="tSubProcess">
    <xsd:complexContent>
        <xsd:extension base="tActivity">
            <xsd:sequence>
                <xsd:element ref="laneSet"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="flowElement"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="artifact"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="triggeredByEvent" type="xsd:boolean"
                default="false"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="task" type="tTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tTask">
    <xsd:complexContent>
        <xsd:extension base="tActivity">
            <xsd:sequence>
                <!-- New sustainability metrics -->
                <xsd:element name="sustainabilityMetrics" type="SustainabilityMetricsType"
                    minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="terminateEventDefinition" type="tTerminateEventDefinition"
    substitutionGroup="eventDefinition"/>
<xsd:complexType name="tTerminateEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition"/>
    </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:element name="textAnnotation" type="tTextAnnotation"
  substitutionGroup="artifact"/>
<xsd:complexType name="tTextAnnotation">
  <xsd:complexContent>
    <xsd:extension base="tArtifact">
      <xsd:sequence>
        <xsd:element ref="text" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="textFormat" type="xsd:string" default="text/plain"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="text" type="tText"/>
<xsd:complexType name="tText" mixed="true">
  <xsd:sequence>
    <xsd:any namespace="##any" processContents="lax" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="throwEvent" type="tThrowEvent"/>
<xsd:complexType name="tThrowEvent" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="tEvent">
      <xsd:sequence>
        <xsd:element ref="dataInput" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="dataInputAssociation" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="inputSet" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="eventDefinition" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="eventDefinitionRef" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="timerEventDefinition" type="tTimerEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tTimerEventDefinition">
  <xsd:complexContent>
    <xsd:extension base="tEventDefinition">
      <xsd:choice>
        <xsd:element name="timeDate" type="tExpression" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="timeDuration" type="tExpression" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="timeCycle" type="tExpression" minOccurs="0" maxOccurs="1"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```



```

        </xsd:choice>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="transaction" type="tTransaction"
    substitutionGroup="flowElement"/>
<xsd:complexType name="tTransaction">
    <xsd:complexContent>
        <xsd:extension base="tSubProcess">
            <xsd:attribute name="method" type="
                tTransactionMethod" default="##
                Compensate"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tTransactionMethod">
    <xsd:union memberTypes="xsd:anyURI">
        <xsd:simpleType>
            <xsd:restriction base="xsd:token">
                <xsd:enumeration value="##
                    Compensate" />
                <xsd:enumeration value="##Image" />
                <xsd:enumeration value="##Store" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:union>
</xsd:simpleType>

<xsd:element name="userTask" type="tUserTask" substitutionGroup="
    flowElement"/>
<xsd:complexType name="tUserTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:sequence>
                <xsd:element ref="rendering"
                    minOccurs="0" maxOccurs="
                    unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="implementation" type="
                tImplementation" default="##unspecified
                "/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

</xsd:schema>

```