Modeling AI-Driven Workflows for Ecosystem Resilience Prediction

Tiago Sousa[1,a], Nicolas Guelfi[1,b], and Benoît Ries[1,c]

[1]University of Luxembourg, Esch-sur-Alzette, Luxembourg
[a,b,c]{tiago.sousa, nicolas.guelfi, benoit.ries}@uni.lu

Technical Report: 1 July 2025

1 Introduction

The intent with this technical report is to provide a comprehensive documentation of our proposed AI-driven workflow for designing and engineering Ecosystem Resilience Prediction Systems (ERPS). The workflow, modeled using the Business Process Model and Notation (BPMN), offers a structured and repeatable process to guide stakeholders from high-level requirements to a complete, validated system design, in an iterative and incremental manner. Our primary objective is to address the challenge of integrating domain-specific ecological knowledge, robust data engineering practices, and state-of-the-art AI models in a coherent and systematic manner.

In the subsequent sections of this report, we will systematically detail the main components of the BPMN workflow. We will describe each pool, lane, and associated activities, explaining their purpose, their interactions, and the artifacts they consume and produce.

1.1 Foundational Metamodels

The AI-driven workflow is built upon a formal, model-driven foundation comprising two primary metamodels that structure the problem and solution domains.

The **Ecosystem metamodel (*EcoSys*)**, shown in Figure 1, is inspired by the DREF framework [1] and is used to formalize the ecological problem domain. It defines an ecosystem's structural components (*entities*), their measurable indicators (*properties*), and the dimensions along which they change (*evolution axes*), among other aspects. A model instantiated from *EcoSys* is referred to in the workflow as an integrated DREF model.

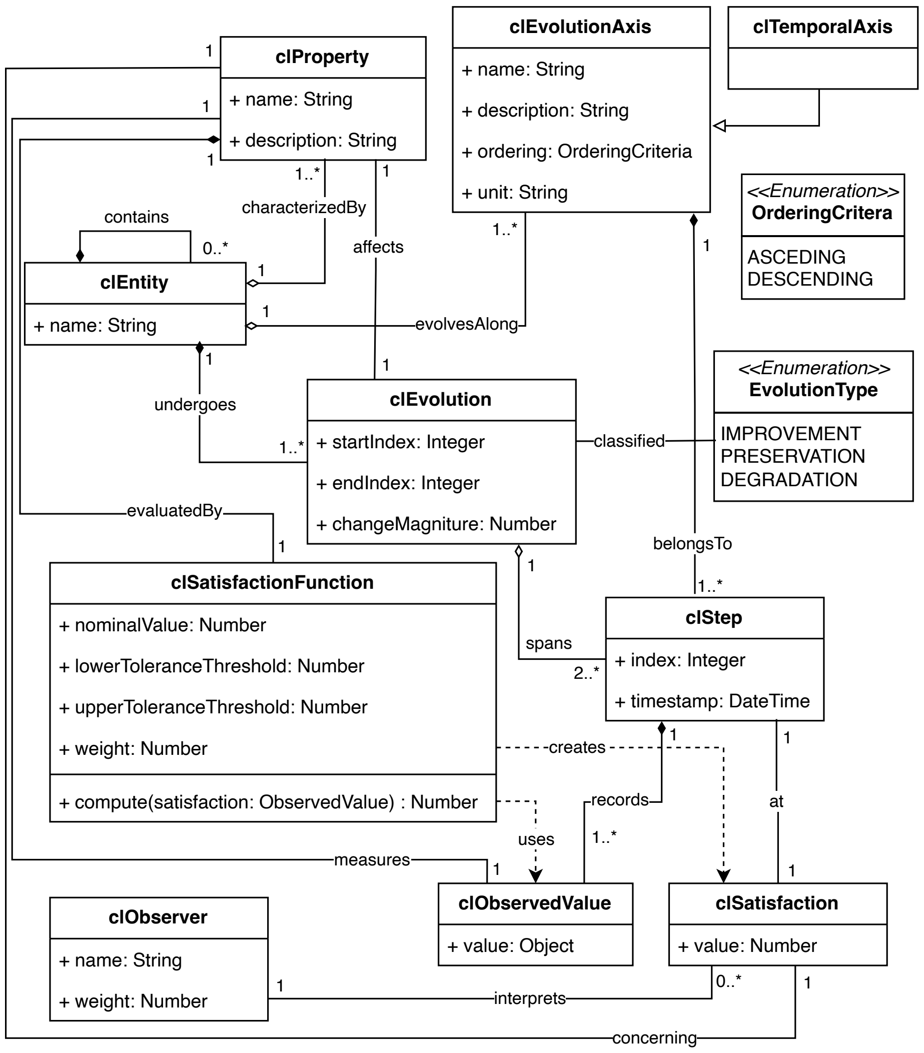
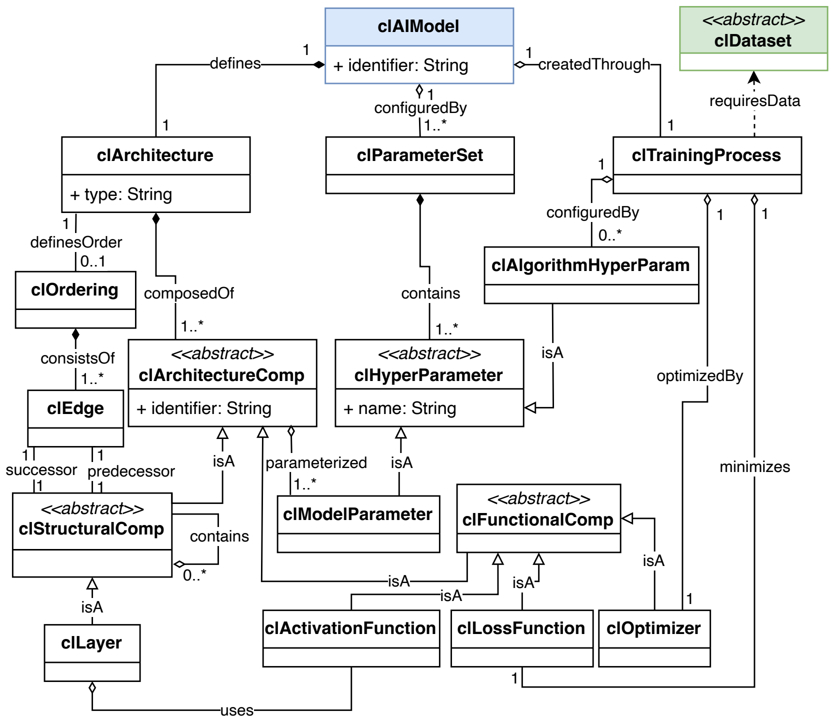


Figure 1: *EcoSys* metamodel.

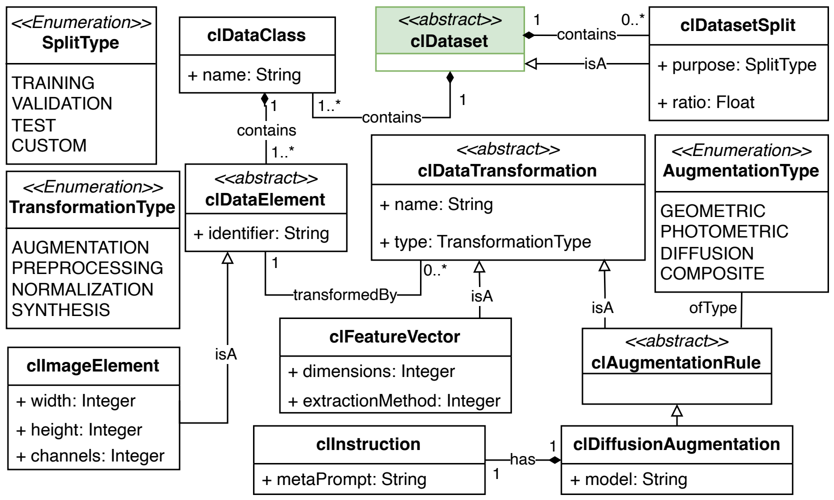
The **Prediction System metamodel (*PredSys*)** defines the solution space and is composed of two parts, shown in Figure 2:

1. The AI metamodel (*PredSys-AI*) provides the constructs to specify neural network architectures, covering their structural components, functional aspects, and training configuration.

2. The dataset metamodel (*PredSys-Dataset*) is used to define the structure of the data for training, validation, and testing, including the specification of data engineering and augmentation.



(a) The AI Part (*PredSys-AI*).



(b) The Dataset Part (*PredSys-Dataset*).

Figure 2: The two parts of the PredSys metamodel.

2 The ERPS Workflow

The workflow is modeled as an adaptive, BPMN-based process designed to manage the complexity of building an ERPS. It ensures the coherent integration of ecosystem requirements, data engineering, and AI modeling. The workflow is structured into distinct pools and processes that represent the logical separation of concerns in the design process. Figure 3 presents a high-level view of the complete workflow.

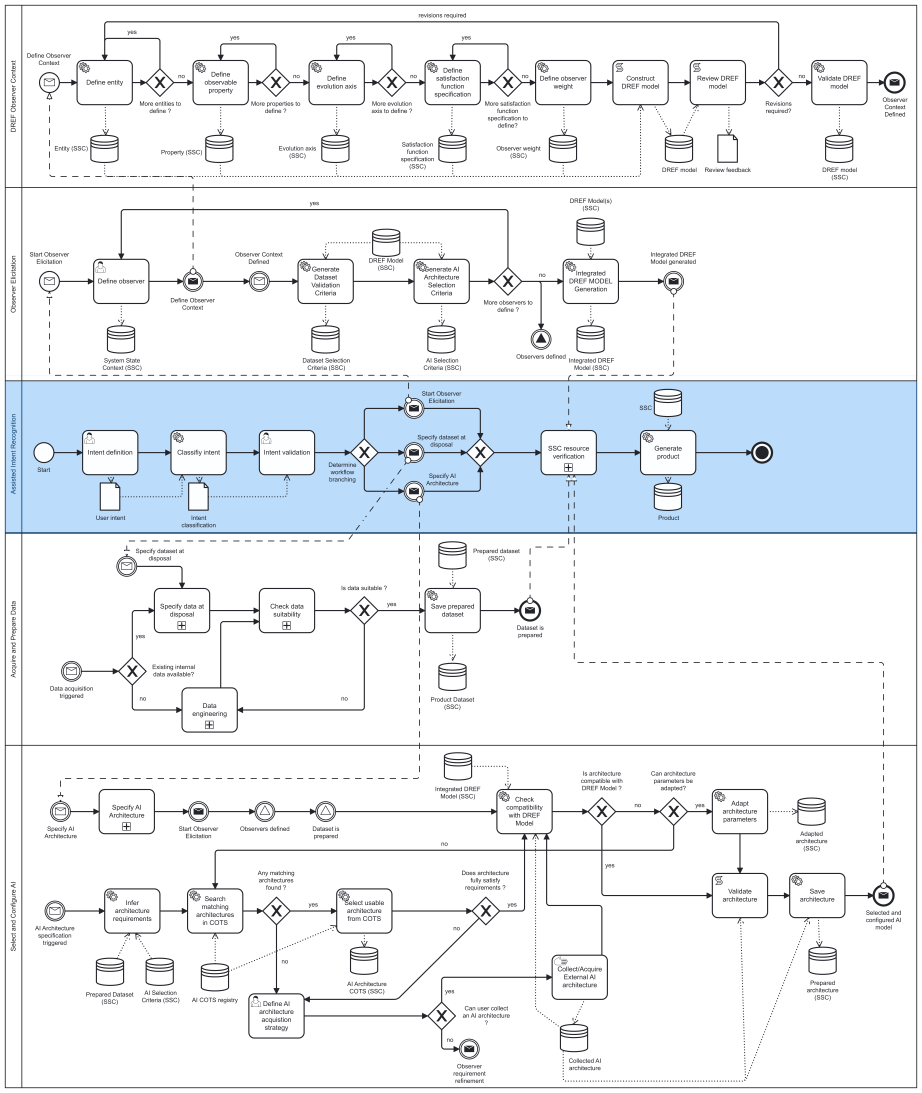

DREF Observer Context
revisions required
yes
Define Observer Context
Define entity
no
More entities to define ?
Define observable property
More properties to define ?
Define evolution axis
More evolution axis to define ?
Define satisfaction function specification
More satisfaction function specification to define?
Define observer weight
Construct DREF model
Review DREF model
Revisions required?
Validate DREF model
Observer Context Defined
Entity (SSC)
Property (SSC)
Evolution axis (SSC)
Satisfaction function specification (SSC)
Observer weight (SSC)
DREF model
Review feedback
DREF model (SSC)
Observer Elicitation
Start Observer Elicitation
Define observer
Define Observer Context
Observer Context Defined
Generate Dataset Validation Criteria
DREF Model (SSC)
Generate AI Architecture Selection Criteria
More observers to define ?
DREF Model(s) (SSC)
Integrated DREF MODEL Generation
Integrated DREF Model generated
System State Context (SSC)
Dataset Selection Criteria (SSC)
AI Selection Criteria (SSC)
Observers defined
Integrated DREF Model (SSC)
Assisted Intent Recognition
Start
Intent definition
Classifiy intent
Intent validation
Determine workflow branching
Start Observer Elicitation
Specify dataset at disposal
Specify AI Architecture
SSC resource verification
SSC
Generate product
User intent
Intent classification
Product
Acquire and Prepare Data
Specify dataset at disposal
Specify data at disposal
Check data suitability
Is data suitable ?
Prepared dataset (SSC)
Save prepared dataset
Dataset is prepared
Existing internal data available?
Data acquisition triggered
Data engineering
Product Dataset (SSC)
Select and Configure AI
Specify AI Architecture
Start Observer Elicitation
Observers defined
Dataset is prepared
Integrated DREF Model (SSC)
Check compatibility with DREF Model
Is architecture compatible with DREF Model ?
Can architecture parameters be adapted?
Adapt architecture parameters
Adapted architecture (SSC)
AI Architecture specification triggered
Infer architecture requirements
Search matching architectures in COTS
Any matching architectures found ?
Select usable architecture from COTS
Does architecture fully satisfy requirements ?
Validate architecture
Save architecture
Selected and configured AI model
Prepared Dataset (SSC)
AI Selection Criteria (SSC)
AI COTS registry
AI Architecture COTS (SSC)
Define AI architecture acquistion strategy
Can user collect an AI architecture ?
Collect/Acquire External AI architecture
Observer requirement refinement
Collected AI architecture
Prepared architecture (SSC)

Figure 3: Overview of the ERPS Workflow.

2.1 Pool: Assisted Intent Recognition

In the proposed workflow, the *Assisted Intent Recognition* pool serves as the main entry point and coordinator of the entire ERPS design process. It contains the sequence of activities that translate a user's goal into a complete, generated Ecosystem Resilience Prediction System.

The process begins with the *"intent definition" user task*, where the user provides their objective, which is captured in a *User Intent* data object. This data object serves as the input to the subsequent *"classify intent" service task*. This automated task processes the user's input and produces a formalized *Intent Classification* data object. To ensure fidelity, this classification is then presented back to the user for review and approval in the *"intent validation" user task*.

Once the intent is validated, an exclusive gateway directs the process flow. This gateway is the mechanism for the workflow's adaptability, routing the process based on the nature of the user's intent by sending a message to trigger one of the three main sub-processes:

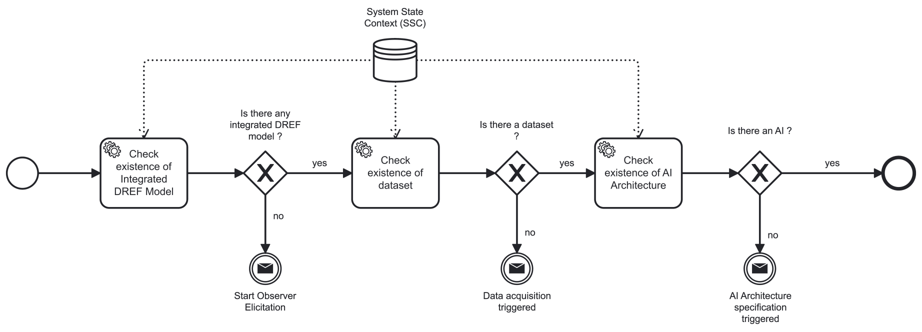
1. **Goal-Driven:** Triggers the *Observer Elicitation* process to formalize ecosystem requirements. For example, a conservation biologist might want to predict the resilience of a coral reef ecosystem to climate change, requiring formal specification of reef health indicators, environmental stressors, and recovery metrics.

2. **Data-Driven:** Triggers the *Acquire and Prepare Data* process to analyze an existing dataset. For example, a researcher with historical satellite imagery and oceanographic data might want to build a prediction system to forecast marine ecosystem changes based on this available data.

3. **Model-Driven:** Triggers the *Select and Configure AI* process to define an application for an existing AI architecture. For example, an organization with a pre-trained deep learning model for image classification might want to adapt it to monitor forest canopy health from aerial photographs.

Following the execution of the appropriate sub-process, all workflow paths converge on the **SSC Resource Verification** sub-process. The System State Context (SSC) is a global data store that facilitates artifact sharing and state management across all pools. The SSC serves as a centralized repository where artifacts are created, read, and updated throughout the workflow execution. This global state management ensures consistency and enables seamless communication between different workflow components. As detailed in Figure 4, this sub-process acts as a gatekeeper that sequentially validates the contents of the SSC. It checks for the existence of three core artifacts: the formal ecosystem specification (the integrated DREF model based on *EcoSys*), the dataset model (*PredSys-Dataset*), and the AI model (*PredSys-AI*). If any artifact is found to be missing, the workflow sends a message to trigger the corresponding process (e.g., *Start Observer Elicitation*, *Data acquisition triggered*) to ensure it is created.

Once this verification step confirms that all required artifacts are present and compatible within the SSC, the pool initiates the final *"generate product" service task*. This is intended to automatically assemble the final, executable ERPS by combining three key components from the SSC: the validated DREF model instance that formalizes the ecosystem requirements, the prepared dataset, and the configured AI architecture adapted to the specific prediction tasks. The complete package is stored in a *Product* data store, after which the workflow concludes.



System State Context (SSC)
Check existence of Integrated DREF Model
Is there any integrated DREF model ?
yes
no
Start Observer Elicitation
Check existence of dataset
Is there a dataset ?
yes
no
Data acquisition triggered
Check existence of AI Architecture
Is there an AI ?
yes
no
AI Architecture specification triggered

Figure 4: Details of the SSC Resource Verification Sub-Process.

2.2 Pool: Observer Elicitation

The *Observer Elicitation* pool is triggered by a message from the *Assisted Intent Recognition* pool when a user's intent is classified as goal-driven. Its purpose is to orchestrate the formalization of requirements from one or more domain experts (Observers). The process flow is as follows:

1. The process starts with the *"define observer" user task*. Following this, a *Define Observer Context* message is sent to trigger the process in the *DREF Observer Context* pool (described in Section 2.3). The workflow then waits until it receives an *Observer Context Defined* message, which signals that a validated **DREF Model** artifact is available in the SSC.

2. Once the confirmation message is received, two sequential *Service Tasks* use the newly created DREF Model as input:

    - *"generate dataset validation criteria"*: This task produces a *Dataset Selection Criteria* artifact, stored in the SSC, to provide formal constraints for the data engineering process. This criteria is used to validate the dataset against the ecosystem requirements, and to select the most relevant data for the prediction task.
    - *"generate AI architecture selection criteria"*: This task produces an *AI Selection Criteria* artifact, also stored in the SSC, to provide formal constraints for the AI model selection process. This criteria

is used to validate the AI model against the ecosystem requirements, and to select the most relevant AI model for the prediction task.

3. A gateway then checks if more observers need to define their context. If "yes", the process loops back to the *"define observer" task*.

4. If "no", a signal indicates that the **Observers Defined** stage is complete, allowing other pools to proceed. The *"integrated dref model generation" service task* is initiated and consumes all the individual **DREF Model(s)** from the SSC and produces a single, consolidated **Integrated DREF Model** artifact.

5. Finally, a message indicates that the integrated DREF Model has been generated, ending the process and allowing the workflow to proceed to the next pool.

2.3 Pool: DREF Observer Context

The *DREF Observer Context* pool becomes active upon receiving a *Define Observer Context* message. It manages the detailed, iterative definition of a single DREF model based on the *EcoSys* metamodel.

The workflow consists of a series of sequential, iterative *Service Tasks* for defining the components of the ecosystem model:

1. *"Define entity"*: Defines a structural component of the ecosystem. The process iterates until all entities are defined, producing *Entity* artifacts in the SSC.

2. *"Define observable property"*: Defines measurable characteristics for the entities. The process iterates until all properties are defined, producing *Property* artifacts in the SSC.

3. *"Define evolution axis"*: Defines the dimensions (e.g., time) along which properties change. It iterates until all axes are defined, creating *Evolution axis* artifacts in the SSC.

4. *"Define satisfaction function specification"*: Defines the functions to evaluate the properties. It iterates until all are specified, creating *Satisfaction function specification* artifacts.

Once these elements are defined, the *"define observer weight" task* sets their relative importance, creating an *Observer weight* artifact. These weights are important when multiple observers are involved, as they allow the system to account for each observer's level of expertise, accountability, or responsibility regarding specific entity properties in the final predictions. For example, an observer with direct accountability for monitoring certain ecosystem properties might be assigned a higher weight for those specific predictions.

The *"construct dref model" script task* then consumes all these artifacts to assemble a complete *DREF model*. This model is passed to the *"review dref* *model" user task*, where the user can review it and validate it. If revisions are required (based on the *Review feedback* produced), the process loops back to the *"define entity" task* to allow for iterative refinement. Otherwise, the *"validate dref model" service task* finalizes the model in the SSC and sends the *Observer Context Defined* message, signaling its completion.

2.4 Pool: Acquire and Prepare Data

The *Acquire and Prepare Data* pool becomes active when a user's intent is classified as data-driven or when a *Data acquisition triggered* message is received. Its purpose is to manage the entire data lifecycle, from specifying an initial dataset to performing the necessary engineering tasks to make it suitable for an AI model. This pool can also be triggered by a message from the *SSC Resource Verification* task if a valid dataset is found to be missing.

The pool contains three main sub-processes, presented more in detail in the following sections, which are themselves defined in separate sub-processes and triggered by message events:

1. **Specify Data at Disposal**: Guides the user in formalizing the characteristics of an existing dataset.
2. **Check Data Suitability**: Validates the specified dataset against the requirements defined in the *Dataset Selection Criteria* artifact in the SSC.
3. **Data Engineering**: Provides a structured workflow for performing transformations, feature engineering, and data augmentation.

Once these steps are complete and a fully prepared and validated dataset model is available in the SSC, a message is sent to signal that the data acquisition and preparation phase is complete.

2.4.1 Specify Data at Disposal

The *Specify Data at Disposal* process, shown in Figure 5, becomes active upon receiving a *Specify dataset at disposal* message or when a *Data acquisition triggered* message is received and there exists a an internal dataset at the disposal of the user. The purpose of this process is to guide the user through the systematic characterization and validation of an existing dataset. The process flow is as follows:

1. The process begins with the *"define data type" user task*, where the user specifies the nature of their dataset (e.g., time-series, images, tabular data).
2. The *"dataset submission" task* allows the user to provide the actual dataset, which is stored as an *Internal dataset(s)* artifact in the SSC.
3. The *"dataset identification" service task* performs an automated analysis of the submitted data, producing a *Dataset analysis* document that characterizes the data's structure and properties.

4. A gateway checks if the dataset is valid for prediction purposes. If the dataset is found to be corrupted or unusable, a *Dataset is invalid* message is sent, and the process terminates.

5. If the dataset is valid, the process continues with the *"define data classes" user task*, where the user specifies the different categories or classes present in the data. This produces *Internal dataset(s) classes* artifacts stored in the SSC.

6. Finally, the *"characterize data elements" service task* performs a detailed analysis of the data elements and their features, producing an *Internal Dataset specification* artifact stored in the SSC.

The process concludes when all dataset characteristics have been formally specified and stored in the SSC, enabling subsequent data engineering and AI configuration steps.

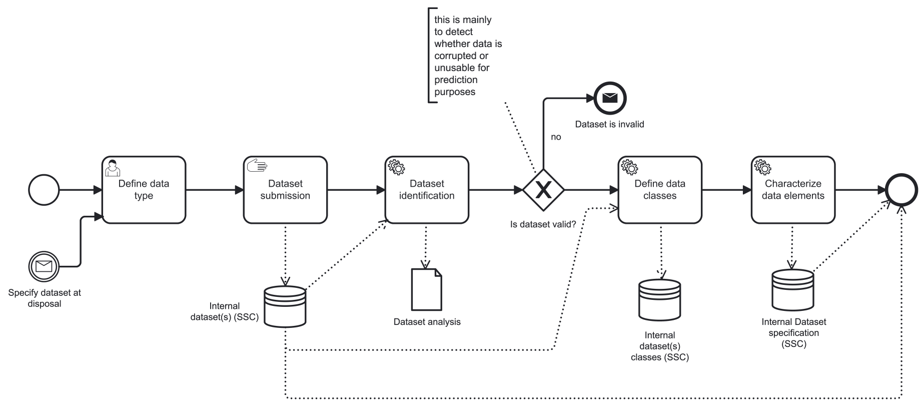

this is mainly to detect whether data is corrupted or unusable for prediction purposes
Dataset is invalid
no
Define data type
Dataset submission
Dataset identification
Is dataset valid?
Define data classes
Characterize data elements
Specify dataset at disposal
Internal dataset(s) (SSC)
Dataset analysis
Internal dataset(s) classes (SSC)
Internal Dataset specification (SSC)

Figure 5: The Specify Data at Disposal Process.

2.4.2 Check Data Suitability

The *Check Data Suitability* process, shown in Figure 6, validates whether the dataset meets the requirements for the ecosystem prediction task. This process ensures that the data can support the intended analysis before proceeding with data engineering. The process flow is as follows:

1. The process begins with the *"check existence of integrated dref model" service task*, which verifies that an integrated DREF model is available in the SSC.

2. A gateway checks if the model already exists. If "no," the process sends a *Start Observer Elicitation* message to trigger the creation of the missing ecosystem specification and waits for an *Observers Defined* signal.

3. If ”yes,” or once the DREF model becomes available, the process proceeds to the *”check suitability” service task*. This task consumes both the *Dataset Selection Criteria* and the *Prepared dataset* artifacts from the SSC to perform the validation.

4. The suitability check determines whether the dataset characteristics align with the requirements derived from the ecosystem model, ensuring compatibility for the intended prediction tasks.

The process concludes when the dataset suitability has been verified, allowing the workflow to proceed with confidence that the data can support the specified ecosystem analysis requirements.

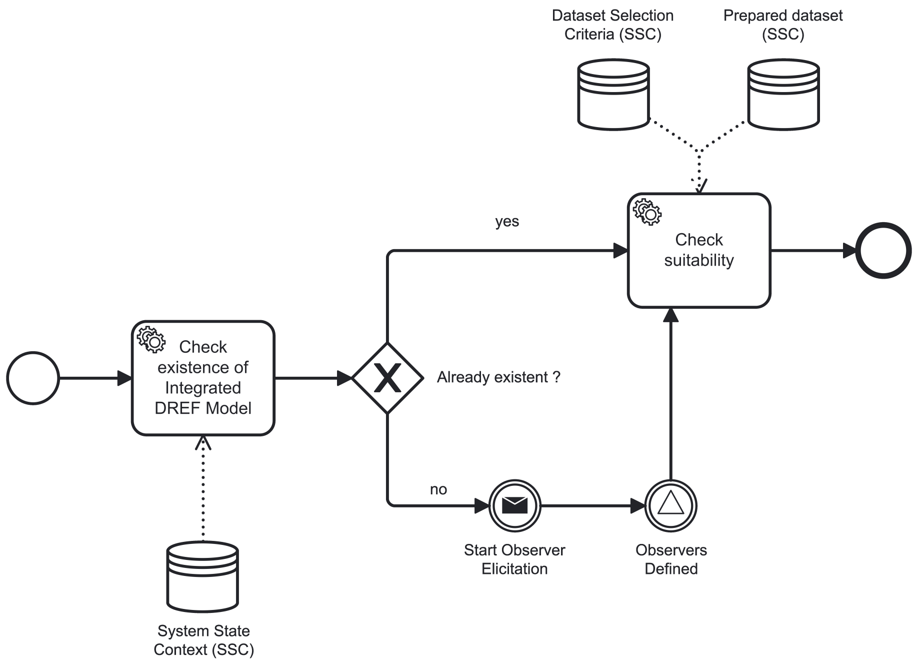

Dataset Selection Criteria (SSC)
Prepared dataset (SSC)
yes
Check suitability
Check existence of Integrated DREF Model
Already existent ?
no
Start Observer Elicitation
Observers Defined
System State Context (SSC)

Figure 6: The Check Data Suitability Process.

2.4.3 Data Engineering

The *Data Engineering* process, shown in Figure 7, addresses the systematic transformation and preparation of data to meet the requirements of the ecosystem prediction system. This process becomes active when data suitability checks indicate that additional data processing is required, or when no suitable dataset exists in the current context.

The process flow encompasses several pathways depending on data availability and requirements:

1. The process begins with the *"search relevant data in cots registry" service task*, which interfaces with external Commercial Off-The-Shelf (COTS) registries to identify potentially suitable data sources. This task enables the system to leverage existing datasets from external repositories, using the integrated DREF model as search criteria.

2. A gateway evaluates whether any matching datasets are found in the external registries.

    - If "yes," the process continues with the *"select usable data from cots" service task*, which extracts applicable data components and stores them as *Usable COTS data* artifacts in the SSC.
    - If no matching datasets are found, or if the selected COTS data is incomplete, the process directs to the *"define data acquisition strategy" user task*, where the user specifies approaches for obtaining the required data.

3. Following the acquisition strategy definition, a gateway determines whether the user can collect or acquire the missing data. If "no," an *Observer requirement refinement* message is sent to revise the ecosystem requirements. If "yes," the process proceeds to the *"collect/acquire external data" manual task*, which produces *Collected data* artifacts.

4. For both COTS and externally collected data, the process includes data completeness validation. If data is determined to be incomplete, the workflow evaluates whether data augmentation is possible through the *"perform data augmentation" service task.* This task supports both traditional geometric transformations and advanced generative AI approaches, providing flexibility in handling different types of ecological data. The augmented data is stored as *Augmented data* artifacts in the SSC.

5. All data sources converge on the *"check compatibility with dref model" service task*, which validates that the processed data aligns with the ecosystem specification requirements.

6. If compatibility issues are detected, the process loops back to data collection activities. If the data is compatible but incomplete, the workflow proceeds to the *"data integration" service task*, which combines multiple data sources into a coherent dataset.

7. The process concludes with the *"save data in ssc" service task*, which stores the final *Prepared dataset* artifact in the SSC. A boundary event signals that the dataset preparation is complete, enabling subsequent workflow steps.

The process includes multiple decision points and feedback loops to ensure data quality and compatibility, with annotations indicating that data can be partial or complete at various stages, providing flexibility in handling diverse data engineering scenarios.

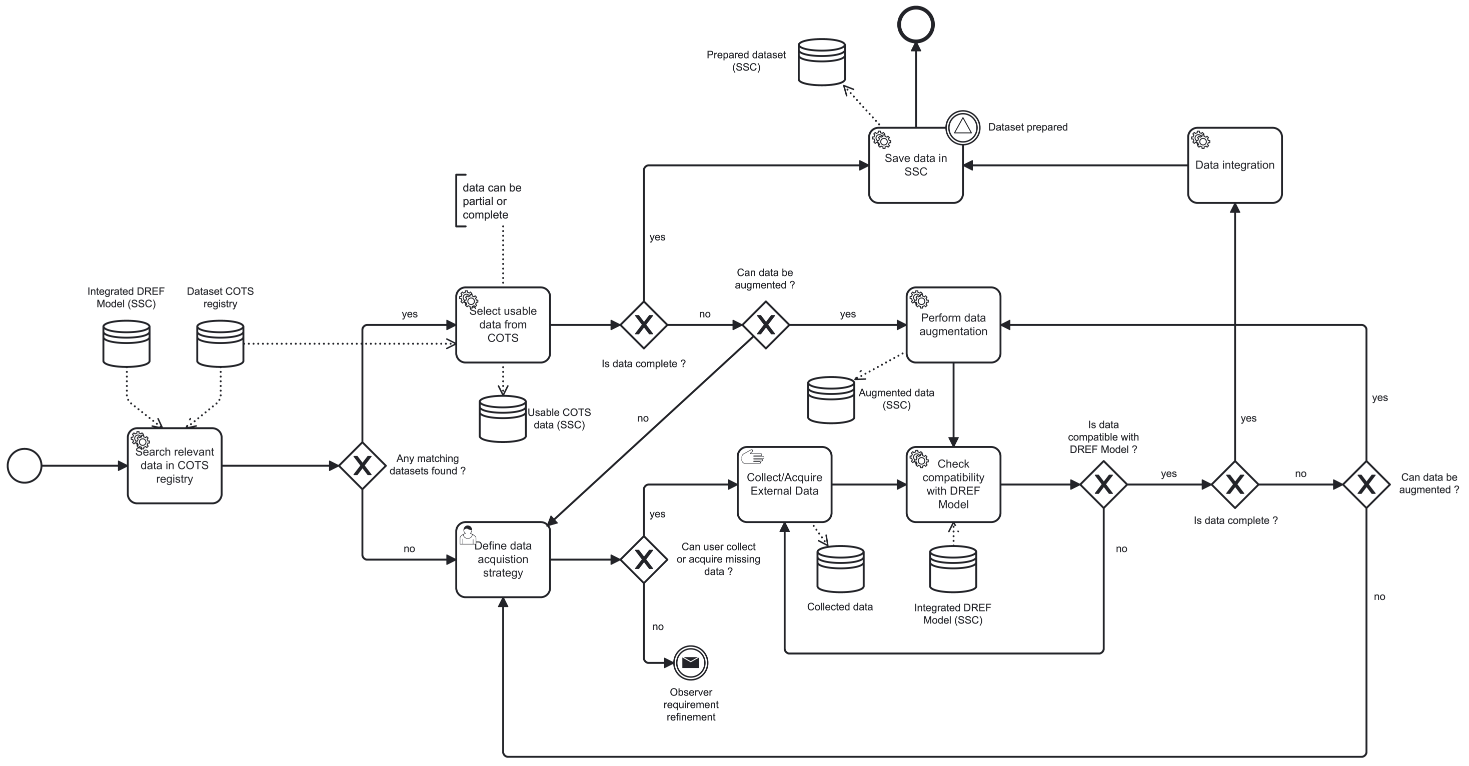

Integrated DREF Model (SSC)
Dataset COTS registry
Search relevant data in COTS registry
Any matching datasets found ?
yes
no
data can be partial or complete
Select usable data from COTS
Usable COTS data (SSC)
Is data complete ?
yes
no
Can data be augmented ?
yes
no
Define data acquistion strategy
Can user collect or acquire missing data ?
yes
no
Observer requirement refinement
Collect/Acquire External Data
Collected data
Perform data augmentation
Augmented data (SSC)
Check compatibility with DREF Model
Integrated DREF Model (SSC)
Is data compatible with DREF Model ?
yes
no
Is data complete ?
yes
no
Can data be augmented ?
yes
no
Data integration
Save data in SSC
Dataset prepared
Prepared dataset (SSC)

Figure 7: The Data Engineering Process.

2.5 Pool: Select and Configure AI

The *Select and Configure AI* pool is responsible for the complete lifecycle of defining, selecting, and preparing an AI model for the ERPS. The process accommodates two distinct initiation paths and includes synchronization with other pools before executing its main logic. The process flow is as follows:

1. The workflow can be initiated via two entry points:
   - A model-driven path begins with a *"specify ai architecture"* message, which triggers the sub-process of the same name (detailed in Section 2.5.1). After the user provides an architecture, this path concludes by sending a *"start observer elicitation"* message, which prompts the formalization of ecosystem requirements to match the provided model.
   - A goal or data-driven path is initiated by an *"ai architecture specification triggered"* message. This path begins with the *"infer architecture requirements" service task*, which uses existing criteria and data to define the requirements for the AI model.
2. Before the main selection process can begin, the workflow synchronizes with other pools by waiting for two confirmation signals: **Observers defined** and **Dataset is prepared**. These events act as gates, ensuring all prerequisites are met.
3. Once synchronized, the *"search matching architectures in cots" service task* queries an *AI COTS registry* to find suitable off-the-shelf models.
4. A gateway assesses if any matching architectures are found. If "yes," the *"select usable architecture from cots" service task* is performed, storing the result as an *AI Architecture COTS* artifact in the SSC. A subsequent gateway then checks if the selected architecture **fully satisfies the requirements**. If it does, the model is sent for compatibility checking.
5. If no matching architecture is found, or if a COTS architecture is incomplete, the workflow proceeds to the *"define ai architecture acquisition strategy" user task.* A gateway then evaluates if the user can **collect an external AI architecture**. If not, an *Observer requirement refinement* message is sent. Otherwise, the *"collect/acquire external ai architecture" manual task* is performed, producing a *Collected AI architecture* artifact.
6. All active paths converge on the *"check compatibility with dref model" service task*, which validates the candidate architecture against the *Integrated DREF Model.*
7. If the model is not compatible ("no"), a gateway checks if its **parameters can be adapted**. If "yes," the *"adapt architecture parameters" task* is executed to make structural adjustments (such as modifying input/output

dimensions or layer configurations) while maintaining the core functionality of the architecture. This creates an *Adapted architecture* artifact, and the result is sent for validation. If "no," the process loops back to the acquisition strategy task.

8. If the model is compatible ("yes"), it is passed to the *"validate architecture" script task*, followed by the *"save architecture" service task*, which stores the final *Prepared architecture* in the SSC.

9. The pool's workflow concludes by throwing a *"selected and configured ai model"* message event.

2.5.1 Specify AI Architecture

The *Specify AI Architecture* process, shown in Figure 8, provides a structured workflow for a user to submit an existing AI model and have it formally registered and validated within the system.

The process flow is as follows:

1. The process begins with the *"upload existing architecture" user task*, where the user provides a model file (e.g., in JSON or ONNX format), which is captured as an *Architecture model* data object.

2. The *"extract architecture metadata" service task* automatically parses this file to identify its structural and functional properties, such as input/output shapes, layers, and operations. The results are stored in an *Architecture metadata analysis* artifact.

3. The *"classify architecture type" script task* then categorizes the model based on its metadata.

4. The *"check architecture completeness" script task* verifies if all required metadata has been successfully extracted. If the specification is incomplete, the user is prompted to *"complete architecture spec"*. If the user cannot provide the missing details, the process terminates.

5. Once the architecture specification is complete, the *"validate architecture model" script task* confirms its integrity.

6. Finally, the *"save architecture model" service task* stores the validated model as an *Architecture model* in the SSC, and a message signals that a *User-defined architecture model* has been prepared, concluding the subprocess.

3 Conclusion

In this technical report, we have detailed a comprehensive, model-driven workflow for the design and engineering of Ecosystem Resilience Prediction Systems

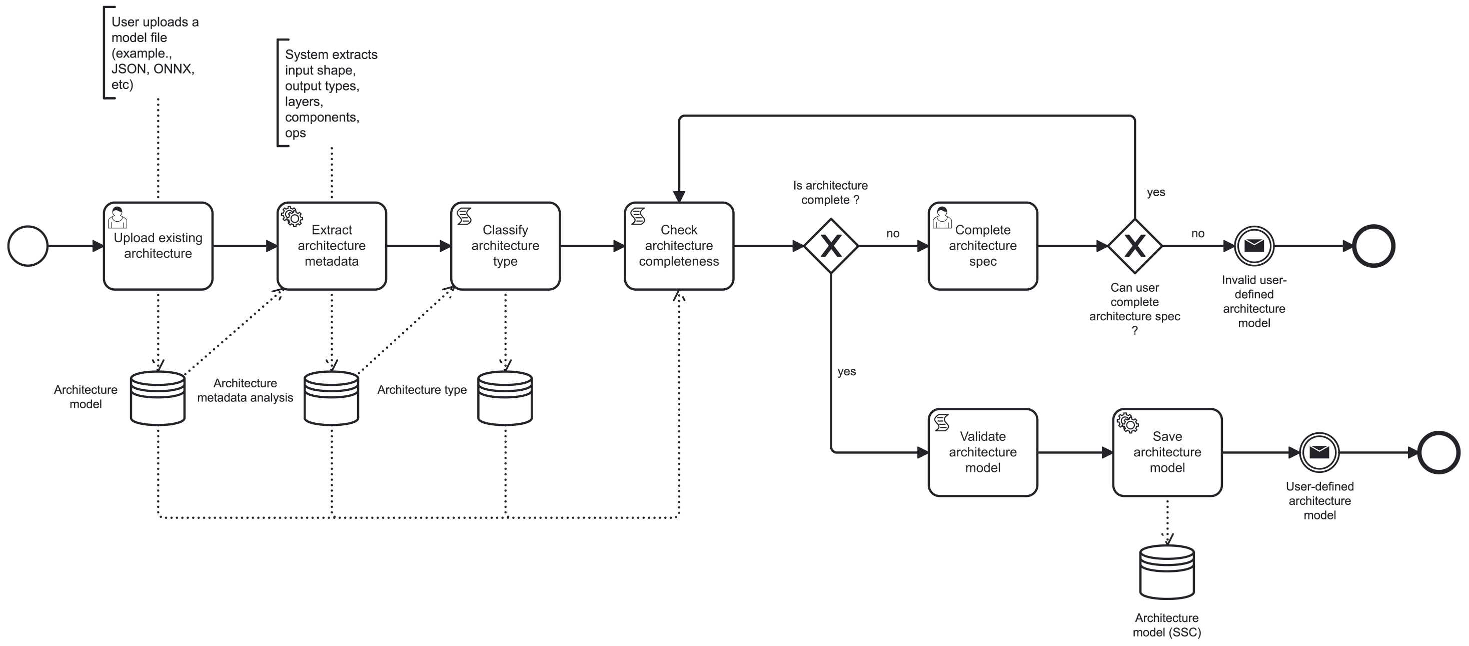

User uploads a model file (example., JSON, ONNX, etc)
System extracts input shape, output types, layers, components, ops
Upload existing architecture
Extract architecture metadata
Classify architecture type
Check architecture completeness
Is architecture complete ?
no
yes
Complete architecture spec
Can user complete architecture spec ?
yes
no
Invalid user-defined architecture model
Validate architecture model
Save architecture model
User-defined architecture model
Architecture model
Architecture metadata analysis
Architecture type
Architecture model (SSC)

Figure 8: The Specify AI Architecture Process.

(ERPS). By leveraging BPMN, we have articulated a structured and adaptive methodology that systematically bridges the gap between high-level ecological requirements and deployable AI-driven solutions. The workflow's modular design, organized into distinct pools for intent recognition, observer elicitation, data acquisition, and AI configuration, ensures a clear separation of concerns while facilitating seamless integration of domain expertise, data engineering, and machine learning. Through detailed descriptions and diagrams of each process, this document provides a foundational guide for researchers, engineers, and domain experts seeking to develop ERPS.

References

[1] Guelfi, N. (2011). A formal framework for dependability and resilience from a software engineering perspective. Open Computer Science, 1(3), 294–328.