IFIP Networking 2024 1571019009
2024 IFIP Networking Conference (IFIP Networking)

Towards Trustworthy Experimental Replication in SLICES-RI

Panayiotis Andreou
UCLan Cyprus
Larnaka, Cyprus
pgandreou@uclan.ac.uk

Artem Osmolovskiy
UCLan Cyprus
Larnaka, Cyprus
aosmolovskiy1@uclan.ac.uk

Panagiotis Hadjidemetriou
UCLan Cyprus
Larnaka, Cyprus
phadjidemetriou1@uclan.ac.uk

Serge Fdida
Sorbonne Université
Paris, France
serge.fdida@sorbonne-universite.fr

***Abstract*—Replication is crucial for maintaining the credibility and integrity of scientific research and is one of Europe's key enablers for Open Science. Several challenges must be addressed to facilitate replication, including applying robust methodologies, holistic data sharing, and detailed data lineage and provenance to allow researchers to leverage insights and findings from prior investigations and introduce novel perspectives or solutions in their field. Research Infrastructures are an important catalyst towards addressing the replication "crisis" by enforcing data sharing by design/default, with appropriate protocols and procedures that provide visibility and transparency to the whole data journey, and compliance with national and international regulations.**

SLICES Research Infrastructure will construct one of Europe's most advanced scientific platforms in the field of digital sciences, promoting scientific research replication through sophisticated policies and services. SLICES-RI transcends traditional data sharing of common digital objects, such as datasets, services and tools, by introducing replication of complex digital objects, such as experimental workflows, which orchestrate advanced tools and services to perform sophisticated experiments in smart networks and systems. This paper presents the preliminary design of the SLICES-RI replication framework, providing insight into its internal structures and mechanisms, and demonstrating how experimental workflows can be replicated. The proposed SLICES Trustworthy Experimental Replication Framework (STEF) provides visibility into the experiment workflow and the underlying data journey, and demonstrates the potential integration of sophisticated indicators, such as explainability, interpretability, and safety, contributing to trustworthy experimentation.

***Index Terms*—experimental replication, trustworthiness, workflow management systems, ML pipelines**

I. Introduction

Experimental replication is fundamental to ensure the reliability and validity of research findings, enabling researchers to repeat and reproduce experiments consistently. The reproducibility of scientific results faces several key challenges [1] in many disciplines, such as inadequate description of experimental methodologies [2], proper accounting for variability in experimental design, execution, and analysis [3], [4], misinterpretation of statistical results [5] and insufficient data sharing [6]. Addressing these challenges is crucial for maintaining the credibility and integrity of scientific research.

The above challenges are more evident in smart networks and systems, due to the highly intricate and dynamic nature of experiments and the complexity of interactions of diverse underlying technologies. In addition, the heavy reliance on big data and computational models, poses unique challenges for replication, such as recording the specific configurations of equipment, software, and analytical pipelines, preserving privacy, ensuring the explainability of predictions, and justifying recommendations. Addressing these challenges requires a multifaceted approach, with greater emphasis on replication-by-design of the full experimental lifecycle. However, replication alone does not necessarily build trust, unless it accounts for several other factors, such as human agency and oversight, technical robustness, security and privacy, and safety.

The SLICES research infrastructure [7] (SLICES-RI) aims to overcome the above challenges and facilitate trustworthy experimental replication, fostering and cultivating cutting-edge research, data-driven science, and scientific data-sharing, fully endorsing Open Science and FAIR principles. To this end, SLICES-RI defines complex metadata models catering to the end-to-end definition, management, and replication of sophisticated experiments, that orchestrate intricate and dynamic AI workflows. The models are flexible to accommodate *trustworthiness indicators*, which can be later used to assess the trustworthiness of each step of the experimental workflow and the workflow as a whole, thus identifying and evaluating pre-, during, and post-anomalies and inconsistencies.

This paper presents the initial design of the SLICES Trustworthy Experimental Replication Framework, coined STEF. We describe the underlying metadata models related to experimentation replication focusing on AI workflows, and show how each step integrates transparent data sharing by design, with guaranteed lineage/provenance. Next, we demonstrate how the models can improve an experimental workflow's trustworthiness, building trust for humans and services. The paper is structured as follows: Section II presents related work that was considered for the design and implementation of STEF. Section III presents the proposed SLICES Trustworthy Experimental Replication Framework (STEF). Next, Section IV presents a prototype implementation of STEF using established frameworks and systems. Finally, Section V concludes the paper and sets directions for future work.

II. Background and Related Work

This section reviews pertinent literature, existing technologies and trends in workflow management systems (WMS),

978-3-903176-63-8 ©2024 IFIP

978-3-903176-63-8/24/$31.00 ©2024 IEEE

Artificial Intelligence as a Service (AIaaS) paradigms, and metadata for experimental workflows, setting a foundation for the discussions and innovations proposed in this work.

A. Workflow Management Systems

Experimental workflows consist of structured definitions of sequences of tasks, enabling equipment configuration, information processing, and service provisioning. Complex experimental workflows may include control structures, such as conditional execution, parallel processes, and branching subflows for enhanced flexibility. Workflow engines are essential tools to consistently automate, orchestrate, and optimize complex experimental workflows, transforming cumbersome experimental procedures into well-defined software-managed operations. Furthermore, these engines integrate additional tools to facilitate automation, efficient resource utilization, optimization, and effective collaboration.

	DS	AF	WF	SM	WEN	BE	EW	PR	CA	WI
Open source	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Popular	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes	Yes
Flexible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Modular	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Version control	No	Yes	No	No	Yes	No	No	Yes	Yes	No
Customizable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Scalable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Designer UI	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes
Drag and drop UI	Yes	No	Yes	No	Yes	No	Yes	No	Yes	Yes
Task Monitor UI	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Pipeline view(DAG)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
CLI	Yes	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes
Runtime control	Yes	No	Yes	No	No	No	Yes	Yes	Yes	No
Long-running WFs	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Event Triggers	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Scheduling	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dynamic Parameters	Yes	No	Yes	No	No	No	Yes	Yes	Yes	No
Parallel workflow	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Branching logic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
For-loops	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Sub-flows	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

TABLE I
QUALITATIVE EVALUATION OF WORKFLOW ENGINES
DS:APACHE DOLPHIN SCHEDULER, AF:APACHE AIRFLOW, WF:WEXFLOW, SM:SNAKEMAKE, WEN:WORKFLOWENGINE .NET, BE:BONITA-ENGINE, EW:ELSA WORKFLOWS, PR:PREFECT, CA:CAMUNDA, WI:WINDMILL

Several open-source workflow engines, such as Apache DolphinScheduler[1], Apache Airflow[2], Wexflow[3], and ELSA Workflows[4], support complex workflows. We conducted a qualitative analysis to compare these workflow engines based on flexibility, modularity, version control, scalability, user interface design, workflow control, and programming characteristics. Our evaluation results are depicted in table I. Apache DolphinScheduler and Apache Airflow lead in development features with their extensive flexibility, modularity, and built-in version control. Prefect[5], Camunda[6], Airflow, DolphinScheduler, and ELSA Workflows are geared towards scalability and stand out for their robust capabilities in handling large-scale workflows efficiently. DolphinScheduler features an intuitive user interface design with a streamlined workflow designer and task monitor features, closely followed by Camunda, Windmill and ELSA Workflows. While these engines prioritize user experience and workflow visualization, others like Wexflow, Snakemake[7], WorkflowEngine .NET[8], bonita-engine[9], and Prefect have simpler designs or lack certain features, such as drag and drop functionality. Furthermore, engines like Airflow, DoplinScheduler, ELSA Workflows, and Snakemake excel in precise control over execution, such as runtime control, event trigger, and scheduling capability.

The choice of a workflow engine depends on application requirements [8], considering the trade-offs between features and functionalities. Open-source workflow engines offer a cost-effective and flexible solution ideal for the dynamic environment of modern RIs.

B. Artificial Intelligence as a Service

AIaaS allows researchers and practitioners to readily access sophisticated AI tools, such as machine learning and data mining models, natural language processing, and image recognition, without requiring extensive expertise in AI. These tools are robust, reliable, and efficient, and typically undergo rigorous testing, validation, and optimization. In addition, they provide several generic adaptation and customization options to support expansion and tailoring for specific applications.

There are several computing paradigms to support the implementation and integration of AIaaS into modern RIs. The Platform as a Service (PaaS) paradigm, like Heroku [10] and Google App Engine [11], is geared toward developing, managing and running applications hiding the complexity of building and maintaining the infrastructure. RIs can use the PaaS paradigm to expose multiple "applications" to allow researchers to orchestrate experiments under a unified platform. However, this limits variability in experimental design, execution, and analysis, which may require more fine-grained control over the execution environment. Infrastructure as Code (IaC), such as Terraform [12], AWS CloudFormation [13], and Pulumi [14], can be used to manage the provision of cloud infrastructure using code offering greater control and flexibility. However, there are multiple experiment scenarios where code needs to be executed in situ in any part of the network, even on individual equipment.

Function as a Service (FaaS), such as AWS Lambda [15], decouples individual AI-/ML- functions enabling code execution in response to events without managing the underlying compute resources offering scalability, simplicity, and cost-efficiency. In addition, FaaS enables individual researchers to

[1]Apache Dolphin Scheduler, https://dolphinscheduler.apache.org/
[2]Apache Airflow, https://airflow.apache.org
[3]Wexflow, https://wexflow.github.io
[4]ELSA Workflows, https://v3.elsaworkflows.io/
[5]Prefect, https://www.prefect.io
[6]Camunda, https://camunda.com
[7]Snakemake, https://snakemake.readthedocs.io/
[8]WorkflowEngine .Net, https://workflowengine.io
[9]Bonita-engine, https://documentation.bonitasoft.com/bonita/
[10]Heroku, https://www.heroku.com
[11]Google App Engine, https://cloud.google.com/appengine
[12]Terraform, https://www.terraform.io
[13]AWS CloudFormation, https://aws.amazon.com/cloudformation/
[14]Pulumi, https://www.pulumi.com/b/
[15]Amazon, Aws lambda-serverless compute-amazon web services, 2023

easily extend existing functions or create new functions accommodating unique experiment requirements. Versioning of functions is also seamlessly supported by all FaaS platforms. A popular alternative to FaaS is Containers as a Service (CaaS), such as Kubernetes [16], Docker Swarm [17], and OpenShift [18], which allow developers to package AI models into containers. CaaS offers more fine-grained control over the execution environment than FaaS and benefits from the scalability and management features. Serverless Containers, such as AWS Fargate [19] and Google Cloud Run [20] go a step further by abstracting away the server and cluster management tasks.

Finally, workflow automation platforms, such as AWS Step Functions [21], Azure Logic Apps [22], and Google Cloud Workflows [23] allow the orchestration of multiple serverless functions and services into complex workflows. This paradigm focuses on orchestrating several microservices and serverless functions to create more complex applications.

C. Metadata Models

The necessary metadata models to describe experimental workflows consist of atomic digital objects, such as datasets and services, and complex ones, such as experimental nodes, and experiment workflows. Several established standards were considered for defining the metadata profiles for experimental workflows and experiment nodes. Workflows can process raw and streaming datasets, and generate one or more final datasets, representing the outcome of the workflow (e.g., scoring dataset). Data Lineage and provenance metadata information is necessary to govern the workflow lifecycle [9], [10] capturing the above and all results produced during/after every workflow step, such as intermediate datasets and AI models [11]. This also includes data handling, such as preparation, metadata labelling [12], dataset split [13], feature extraction/engineering [14] and feature validation [15], and model parameters tuning [16] and versioning [10].

D. Trustworthy factors

Trustworthy experimentation is crucial for ensuring the reliability and validity of research. It is also essential for compliance with the European Union's AI Act and Ethics Guidelines for Trustworthy AI . Several factors contribute to building trust in the full experiment lifecycle, such as (i) *transparency*, providing structured means to investigate any part of the experiment, including generated data, component configurations, and environment influence; (ii) *explainability*, showing how each experiment step works and why, and unveiling the internal operations of complex AI models understandably; (iii) *interpretability*, providing human-centric interpretations of the reasoning behind AI decisions; (iv) *privacy*, ensuring that experiment steps are privacy-preserving, compliant with privacy-by-design/default (as defined by GDPR), and provide indications to track anomalies [17] (e.g., potential model drifts and data leakage; (v) *bias* ensuring that models are unbiased, non-discriminatory, and fair; (vi) *robustness*, demonstrating that experiment steps meet certain Quality of Service (QoS) requirements, such as performance, throughput, and latency; and *safety*, ensuring that experiments operate within safe parameters, especially when humans are involved. Transforming these factors into indicators and fusing them using a formal model [18] can contribute to a trustworthy evaluation of the experiment, further promoting replication.

[16]Kubernetes, https://kubernetes.io
[17]Docker Swarm, https://docs.docker.com/engine/swarm
[18]OpenShift, https://docs.openshift.com/
[19]AWS Fargate, https://aws.amazon.com/fargate/
[20]Google Cloud Run, https://cloud.google.com/run
[21]AWS Step Functions, https://aws.amazon.com/step-functions/
[22]Azure Logic Apps, https://learn.microsoft.com/en-us/azure/logic-apps
[23]Google Cloud Workflows, https://cloud.google.com/workflows

III. Trustworthy Experimental Replication Framework

This section presents the proposed Trustworthy Experimental Replication Framework for the SLICES-RI, coined STEF, including its architecture and main components.

A. Architecture

The proposed architecture of STEF is illustrated in Figure 1. STEF utilizes the *Metadata Registry System (MRS)*, which provides discovery services for all digital objects, such as datasets, experiment nodes and experimental workflows. MRS supports versioning of metadata, which, in turn, allows for multiple versions of all digital objects (e.g., several versions of an AI algorithm). Experiment workflows are the definitions of experiments, describing the interactions between experiment nodes and other digital objects, such as datasets.

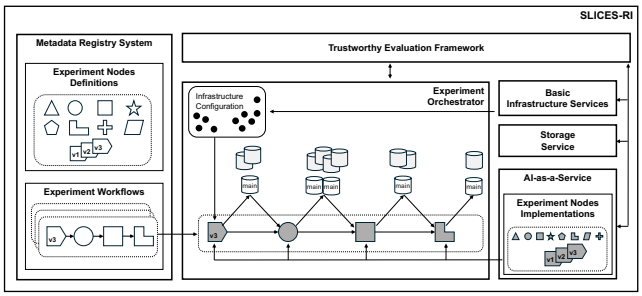

Fig. 1. Trustworthy Experimental Replication Framework

AI-as-a-Service consists of the realization of experiment nodes utilizing one of the serverless architecture paradigms described in section II-B. AI algorithms are stored as atomic units that can be executed as a service or transferred for execution in a specific device (virtual or physical). User-defined end-to-end services are also supported, consisting of predefined AI workflows. *Basic Infrastructure Services* are core services of SLICES-RI offering management and monitoring of all software and hardware resources. Within the context of SLICES-RI, this includes non-traditional RI operations, such as multi-site reservations, configurations, and orchestrations of multiple heterogeneous geo-diverse resources. *Storage Service* is also a core SLICES-RI service offering high-performance storage

services for all digital objects, such as datasets. Furthermore, the Storage Service facilitates the integration of external data sources (e.g., datasets from EOSC) in collaboration with the data management services. The system's heart is the *Experiment Orchestrator*, responsible for the end-to-end execution of experimental workflows, consisting of multiple experiment nodes and workflow control (e.g., conditional execution, loops, parallel processing, and events-based execution).

B. Facilitating Use Case

We utilize a simplified real-world use case of a scientific experiment in smart agriculture, which consists of an experiment workflow comprising four steps. The experiment starts by ingesting data from a wireless sensor network, reserved through the SLICES basic infrastructure service. Assume that the WSN was configured to record environmental parameters (e.g., temperature, humidity, light, wind speed) for a specific period (e.g., one month) at predefined intervals (e.g., every second) and also store the GPS value of each sensor node. Second, the workflow executes a pre-processing node that handles missing values by imputing them using a predefined strategy (e.g., the mean value of each attribute) set by the experimenter. Next, the data are fed into an ML clustering model to detect regions with similarities in the deployment by analyzing patterns. Finally, the processed data, including the new features (i.e., the cluster label), are exported to a format that was selected by the experimenter.

C. Metadata Registry System

The Metadata Registry System realizes the SLICES core metadata schema (SFDO), harmonizing the metadata of all digital objects, such as datasets, experiment nodes, and experimental workflows. In the following sections, we provide insight into the schema of each metadata profile, drawing references to our facilitating example.

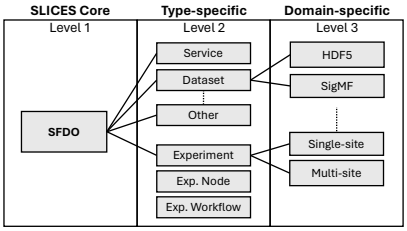

Fig. 2. SLICES Fair Digital Object (SFDO) Hierarchical Metadata Model

1) Digital Objects: SLICES implements a flexible hierarchical metadata model consisting of three levels as illustrated in Figure 2. The first level consists of compulsory domain-agnostic information that can describe any digital object (e.g., data, services, publications, tools), ensuring that it conforms to FAIR principles and beyond. It includes basic information (e.g., identification, creator, description), management information (e.g., version, access), and the object type. SLICES also employs a second level of compulsory metadata attributes, which are type-specific, to enhance machine-actionability for simple and complex types, such as datasets, services, and experiments. Finally, the third level incorporates optional domain-specific attributes to enhance interoperability for specific communities (e.g., incorporating attributes for compliance with a standard, such as SigMF).

2) Experimental Nodes: Experimental nodes are essential components of the experiment workflow, each serving a unique function within the experimentation process. Experimental nodes are designed to be applicable across diverse scientific domains, providing researchers with the openness and flexibility to perform a wide range of experiments and operations transparently. Each node offers significant flexibility in its usage, enabling researchers to create custom nodes and adapt them to different experimental setups and data requirements.

The metadata schema of experiment nodes is a level-2 SFDO object inheriting the metadata described in Section III-C1. Additionally, it comprises experiment-specific information: inputs, outputs, and configuration parameters. Inputs provide essential information for the node to perform its designated task effectively, including datasets and metadata. Outputs consist of processed data, statistical results, visualizations, or other relevant information derived from input data. Configuration parameters enable users to customize the behavior of each node (e.g., setting the k value for k-means), ensuring adaptability to experimental requirements.

```
1 {
2   "sfdo (inherited metadata examples)": {
3     "identifier": "SLICES/EN/15/v1",
4     "type": { "value": "Experiment Node" },
5     "name": { "value": "Impute Node" },
6     "version": { "value": "v1" },
7     "faas": "SLICES/AI/EN/15/v1",
8     "...": { "other sfdo metadata" }
9   },
10  "Experiment Node (type-specific metadata)": {
11    "inputs": {
12      "MainDataset": {"type": "dataset",
13                      "required": "true" },
14      "Method": {"required": "false" },
15      "AttributeList": { "type": "array",
16                         "required": "false" },
17      "...": "other input specifications"
18    }
19    , "configuration": {
20      "Method": { "default": "mean" },
21      "...": { "other configurations" }
22    }
23    , "outputs": {
24      "MainDataset": { "type": "dataset"},
25      "ImputeStatistics": { "type": "dataset"},
26      "...": "other output specifications"
27    }
28 }
```

```
1 {
2   "experiment_info": { "id": "SLICES/EXP/25" },
3   "variables": [
4     {"id": "SFDO:VAR/1", "typeName": "Dataframe"},
5     {"id": "SFDO:VAR/2", "typeName": "Int"}
6   ],
7   "start": {"@sfdo_ref": "SLICES/EXP/25/WRF/1"},
8   "workflows": [
9     {"id": "SLICES/EXP/25/WRF/1",
10      "nodes": [
11        {"id": "SFDO:ENI/1", "name": "IngestData"},
12        {"id": "SFDO:ENI/2", "name": "ImputeValues"},
13        {"id": "SFDO:ENI/3", "name": "Clustering"
14            "inputs": {
15                "Dataframe": "@sfdo_ref": "SFDO:VAR/1"}
16          },
17        },
18        {"id": "SFDO:ENI/4", "name": "Evaluate"}
19      ],
20      "start": {"@sfdo_ref": "SLICES/EXP/25/WRF/1/ENI/1"},
21      "connections": [
22        {"source": "SFDO:ENI/1", "target": "SFDO:ENI/2"},
23        {"source": "SFDO:ENI/2", "target": "SFDO:ENI/3"},
24        {"source": "SFDO:ENI/3", "target": "SFDO:ENI/4"}
25      ]
26    }
27  ]
28 }
```

Fig. 3. Metadata for (left) Experiment Node (right) Experiment Workflow

An example of the *Impute Node* metadata is illustrated in Figure 3 (left). Only a subset of the metadata is presented to support the example. Lines 1-8 show a subset of the inherited metadata from the SFDO model, enabling the findability and accessibility of the digital object. Next, lines 10-26 provide the type-specific metadata of the experiment node: (i) lines 9-16 list the input specifications; (ii) lines 19-21 list the configuration of the node; and (iii) lines 23-26 list the output specifications. The Impute Node consists of a required main input dataset (line 10), the optional imputation method (line 12), and the optional attribute list that imputation will be performed (line 15). The node's output includes the imputed main dataset (line 24) and the labeled output ImputeStatistics, which can be used to interpret the node's results.

3) Experimental Workflows: An experiment workflow consists of interconnected experimental nodes orchestrated to ex-

ecute scientific experiments and operations. These workflows are highly flexible and customizable, enabling researchers to construct complex data processing pipelines tailored to specific experimental requirements. Researchers can orchestrate experimental nodes in various configurations, ranging from linear workflows such as those observed in Machine Learning Operations (MLOps), to more complex arrangements involving multiple paths and programming aspects. In linear workflows, experimental nodes are arranged sequentially, with data/results linearly flowing through each node. In comparison, complex workflows involve multiple paths and may have convoluted dependencies between nodes. Researchers can design workflows with branching paths, parallel processing, and conditional execution, enabling the creation of advanced pipelines capable of handling diverse data analysis and processing tasks.

The metadata schema of experimental workflows is a level-2 SFDO object and, similar to the experiment node, inherits the same metadata for discoverability and access. Additionally, each definition comprises experimental workflow-specific information, such as variables and configuration parameters, experiment nodes, and interactions between nodes. The workflow metadata representing the facilitating example of Section III-B is illustrated in Figure 3 (right). Lines 11-18 show that the workflow initializes four experiment nodes, named IngestData, ImputeValues, Clustering, and Evaluate, which are later connected to form a pipeline in Lines 22-24.

D. Trustworthiness

Experimental nodes and workflows promote trustworthiness through adhering to trustworthy-by-design principles and monitoring selected factors mentioned in Section II-D. The framework prioritizes lineage- and provenance-by-default by employing automated procedures tracking the origin and history of data inputs, transformations, and outputs, ensuring traceability. Provenance metadata captures contextual information, such as experimental conditions and software versions. Furthermore, experiment nodes are designed to be interpretable and explainable, providing meaningful insights into underlying processes and decision-making logic through carefully crafted data and metadata. These are extracted and processed during the node's processing, and geared to enhance end-to-end interpretation (process, model, and results).

IV. Prototype Workflow Management System

This section provides an overview of the prototype implementation of the experimental workflow management prototype, realizing the metadata design presented in the previous section. The prototype orchestrates experimental nodes to construct and execute various scientific workflows, showcasing the flexibility and effectiveness of our experimental model in practical research scenarios. The prototype system is composed of 4 main components: (i) the experiment orchestrator, which serves as the entry and the heart of the system, (ii) the FaaS service, which is responsible for executing the compute-oriented experiment nodes, (iii) the storage service, which is responsible for storing node-intermediary outputs and the final results, and (iv) the MRS which is populated by the orchestrator and enables long-term discovery of the various digital objects. The entire system is deployed in a single Kubernetes cluster. The following sections provide insight into the implementation of each system component.

A. Orchestrator

The orchestrator is the entry point into the system for experimenters. It presents a web-based UI for visual experiment design, inspection and on-demand execution. When the experimenter requests an execution of the experiment, the orchestrator invokes the FaaS service for each experiment node with execution-specific parameters (e.g., the dataset produced by the previous node). The orchestrator then waits for the function to complete and executes the following node. Several nodes can execute in parallel depending on the configuration of the experiment workflow. Once the experiment is complete, the orchestrator invokes the MRS to register the results, the generated datasets, etc. Currently, Elsa Workflows[24] is used as the basis for the orchestrator. It was selected as it is a proven open-source project with significant out-of-box capabilities. However, the full implementation of SLICES-RI, will require a bespoke orchestrator to be implemented, to accommodate the unique requirements of the SLICES-RI.

B. Workflow Engine Implementation

Custom activities within the ELSA environment represent individual experimental nodes. These activities encapsulate the functionality of each node, including its inputs, outputs, configuration parameters, and the associated Python scripts. By categorizing these activities under relevant domains and categories, such as "SLICES" the implementation maintains clarity and organization within the workflow.

C. Node Implementation

The core logic for data processing tasks is implemented using Python scripts associated with each experimental node. These scripts utilize libraries like pandas for efficient data manipulation and follow predefined strategies tailored to specific experimental requirements. All nodes conform to the same specification, which realizes all data: inputs, outputs, and configuration parameters, as described in Section III-C2. All data are implemented using key-value pairs for efficient referencing, especially in the case of default inputs/outputs, such as the main dataset of an ML pipeline. By separating the data processing logic into Python scripts, the implementation ensures modularity and flexibility, allowing for easy customization and adaptation to different experimental scenarios.

D. Node Execution

To orchestrate the execution of Python scripts within the ELSA environment, the experiment scripts are deployed to a FaaS framework. The FaaS paradigm was selected as the baseline configuration for experiment nodes as it supports

[24]https://v3.elsaworkflows.io/

versatile scenarios and separates the nodes into small, easily reusable pieces while providing optimal use of the experimenting infrastructure. Knative was selected as the FaaS execution environment. When ELSA triggers the execution of a particular node, it sends a JSON request to Knative, containing information regarding inputs, outputs, and configuration parameters. Knative then executes the Python script and returns the results to ELSA, completing the workflow step.

E. Storage service

The Storage Service stores intermediary node outputs and the final results. It is implemented using the object-storage paradigm, which can be interfaced with using the de-facto standard AWS S3 API. Due to its widespread popularity and adoption, the Minio[25] software was selected as for implementation. Is it possible to replace Minio due to the use of the de-facto standard AWS S3 API. The API is also implemented by other software packages, such as SeaweedFS with S3 gateway[26], GarageHQ[27] and OpenStack Swift with Swift3 middleware[28]. The orchestrator is responsible for managing the layout of the stored data (e.g. names) and issuing temporary access credentials to node executions with access scope limited only to the specific files the node is expected to read/write.

F. Trustworthiness Evaluation

Trustworthiness evaluation is achieved by integrating the outputs discussed in Section III-D with monitoring data generated within SLICES-RI by all collaborating services as illustrated in Figure 1. The data are then transformed into indicators based on predefined QoS requirements. The current prototype adopts a naive approach where if an experiment node supports a trustworthiness factor (e.g., the node supports lineage and explainability if provisions were made to store intermediate datasets and explain operations respectively) then the indicator score is 100%. The indicators are then fused into a *trustworthiness index* using an ensemble system. Assuming I is the set of all indicators, every trustworthy indicator $i \in I$ is of equal weight ($w(i)$ proportional to the number of indicators (i.e., $|I| = k, w(i) = \frac{1}{k}, \forall i \in I$). Finally, the trustworthiness indicators and index are added to the metadata of the experiment.

V. Conclusions and Future Work

This paper presented the preliminary design of the SLICES-RI replication framework, describing how experimental workflows can be defined with structured metadata conforming to Open Science and FAIR principles. The proposed SLICES Trustworthy Experimental Replication Framework (STEF) provides visibility into the experiment workflow and the underlying data journey, and demonstrates the potential integration of sophisticated factors, such as explainability, interpretability, and safety, contributing to trustworthy experimentation. We show a prototype implementation of the STEF framework with selected trustworthiness factors: lineage, explainability, and interpretability. In the future, we plan to develop an enhanced version of STEF that uses a formal framework to assess and integrate the trustworthiness indicators.

[25]https://min.io/

[26]https://github.com/seaweedfs/seaweedfs/wiki/Amazon-S3-API

[27]https://garagehq.deuxfleurs.fr/

[28]https://docs.openstack.org/mitaka/config-reference/object-storage/configure-s3.html

Acknowledgment

This work was supported by the EU HE program under grant agreement No 101079774 for the SLICES-PP project.

References

[1] R. Moonesinghe, M. J. Khoury, and A. C. J. W. Janssens, "Most published research findings are false—but a little replication goes a long way," *PLOS Medicine*, vol. 4, no. 2, pp. 1–4, 02 2007.

[2] E. Desjardins, J. Kurtz, N. Kranke, A. Lindeza, and S. H. Richter, "Beyond Standardization: Improving External Validity and Reproducibility in Experimental Evolution," *BioScience*, vol. 71(5), pp. 543–552, 2021.

[3] M. Lowe, R. Qin, and X. Mao, "A review on machine learning, artificial intelligence, and smart technology in water treatment and monitoring," *Water*, vol. 14, no. 9, 2022.

[4] D. Lakens, "The practical alternative to the p value is the correctly used p value," *Perspectives on Psychological Science*, vol. 16, no. 3, pp. 639–648, 2021.

[5] A. F. Markus, J. A. Kors, and P. R. Rijnbeek, "The role of explainability in creating trustworthy artificial intelligence for health care: a comprehensive survey of the terminology, design choices, and evaluation strategies," *Journal of biomedical informatics*, vol. 113, p. 103655, 2021.

[6] C. G. Begley and J. P. Ioannidis, "Reproducibility in science: improving the standard for basic and preclinical research," *Circulation research*, vol. 116, no. 1, pp. 116–126, 2015.

[7] S. Fdida, N. Makris, T. Korakis, R. Bruno, A. Passarella, P. Andreou, B. Belter, C. Crettaz, W. Dabbous, Y. Demchenko, and R. Knopp, "Slices, a scientific instrument for the networking community," *Computer Communications*, vol. 193, pp. 189–203, 2022.

[8] T. Heinis, C. Pautasso, and G. Alonso, "Design and evaluation of an autonomic workflow engine," in *Second International Conference on Autonomic Computing (ICAC'05)*, 2005, pp. 27–38.

[9] H. Miao, A. Li, L. S. Davis, and A. Deshpande, "Towards unified data and lifecycle management for deep learning," in *2017 IEEE ICDE*, 2017, pp. 571–582.

[10] B. Derakhshan, A. R. Mahdiraji, T. Rabl, and V. Markl, "Continuous deployment of ml pipelines." in *EDBT*, 2019, pp. 397–408.

[11] G. Gharibi, V. Walunj, R. Nekadi, R. Marri, and Y. Lee, "Automated end-to-end management of the modeling lifecycle in deep learning," *Empirical Software Engineering*, vol. 26, pp. 1–33, 2021.

[12] T. Fredriksson, J. Bosch, and H. H. Olsson, "Machine learning models for automatic labeling: A systematic literature review." *ICSOFT*, pp. 552–561, 2020.

[13] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, and G. Szarvas, "On challenges in ml model management," *IEEE Data Engineering Bulletin*, 2015.

[14] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data lifecycle challenges in production machine learning: a survey," *ACM SIGMOD Record*, vol. 47, no. 2, pp. 17–28, 2018.

[15] D. Baylor, E. Breck, H.-T. Cheng *et al.*, "Tfx: A tensorflow-based production-scale machine learning platform," in *ACM SIGKDD*, 2017, pp. 1387–1395.

[16] L. E. Lwakatare, I. Crnkovic, and J. Bosch, "Devops for ai–challenges in development of ai-enabled applications," in *2020 SoftCOM*. IEEE, 2020, pp. 1–6.

[17] C. Duckworth, F. P. Chmiel, D. K. Burns, Z. D. Zlatev, N. M. White, T. W. Daniels, M. Kiuber, and M. J. Boniface, "Using explainable machine learning to characterise data drift and detect emergent health risks for emergency department admissions during covid-19," *Scientific reports*, vol. 11, no. 1, p. 23017, 2021.

[18] C. Agarwal, S. Krishna, E. Saxena, M. Pawelczyk, N. Johnson, I. Puri, M. Zitnik, and H. Lakkaraju, "Openxai: Towards a transparent evaluation of model explanations," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 784–15 799, 2022.