Concept of Federated Synthetic Datasets Generation as a Service for the EOSC

David Cymbalak
Slovak Centre of Scientific and Technical Information
Kosice, Slovak Republic
david.cymbalak@cvtisr.sk

***Abstract*—This paper introduces the concept of Federated Synthetic Datasets Generation as a Service (SDGaaS), a proposed architecture for the European Open Science Cloud (EOSC) ecosystem intended to shift the paradigm from dataset storage to on-demand dataset creation. The design leverages Large Language Models to generate source code for dataset generators, which would then be validated through a human-in-the-loop feedback cycle, containerized, executed on federated computing infrastructure, and deposited into research product repositories. Current AI methods for both code-based and direct dataset generation are analyzed, including the emerging role of agentic AI systems. A conceptual solution design comprising five abstract federated node roles with persistent identifier traceability is presented, along with a discussion of how this approach could extend the FAIR principles by enabling researchers to create what they cannot find. The concept is situated within the context of EOSC Node Slovakia as an illustrative case for smaller national nodes seeking to evolve from passive data stewardship toward active data generation capabilities.**

***Keywords*—synthetic data generation, EOSC, SDGaaS, federated infrastructure, FAIR**

I. Introduction and Motivation

Synthetic data are gaining significant momentum. The World Economic Forum's September 2025 briefing paper *"Synthetic Data: The New Data Frontier"* positioned synthetic data as a foundational technology for the next decade of AI development and a critical component of responsible data governance [1].

Multiple factors contribute to this growing relevance. Modern foundation models require vast amounts of curated training data, yet high-quality text and image corpora on the public internet are approaching saturation [2]. Real-world data collection remains expensive and increasingly constrained by regulation. The EU AI Act, whose general-purpose AI obligations took effect on August 2, 2025, mandates that AI providers publicly disclose training data summaries. From August 2026, the majority of provisions for high-risk AI systems become applicable, including requirements for bias detection and correction where synthetic data can serve as a compliance mechanism [3]. The intersection of the AI Act and EU copyright law further requires that AI developers verify whether training data sources carry copyright reservations, creating additional incentives for synthetic data as a compliant alternative [4].

Synthetic datasets, meaning data artificially generated rather than collected from real-world events, can be produced at arbitrary scale and tailored to specific distributions while inherently sidestepping many privacy concerns. Applications span healthcare, where federated synthetic data networks enable cross-border data access without exposing patient records [5], natural language processing for low-resource languages [6], environmental monitoring for augmenting sparse pollution and climate observations [7], and cybersecurity for training network intrusion detection systems with synthetic attack patterns [8]. However, synthetic data generation also carries known risks, including the potential to encode and amplify biases, produce realistic-looking but statistically misleading samples, and enable generation of deceptive content such as deepfakes [9]. Addressing these risks requires robust validation mechanisms in any generation pipeline.

The European Open Science Cloud (EOSC) represents a strategic initiative to provide European researchers with a federated environment for sharing and reusing research data across borders and disciplines. The EOSC Federation has entered its operational build-up phase, with multiple national and thematic nodes being onboarded and an expanding catalogue of services [10].

However, the current EOSC paradigm is fundamentally passive: it assumes that the dataset a researcher needs already exists somewhere. This assumption breaks down when the required data distribution is highly specific, when the domain is nascent, or when privacy constraints prevent sharing source data.

Federated SDGaaS is proposed as a paradigm shift: rather than solely enabling researchers to find existing datasets, such a platform would empower them to create datasets on demand through a federated, and FAIR-compliant workflow. This concept envisions transforming EOSC from a data warehouse into a data factory.

II. The Slovak EOSC Node and Federated SDGaaS

The EOSC architecture is designed as a distributed system of national and thematic nodes. Slovakia's emerging national EOSC node slowly provides a case study of how a smaller European research ecosystem is navigating the path toward full EOSC integration, and how this creates both the opportunity and necessity for services such as Federated SDGaaS.

A. Current State of EOSC Node Slovakia

EOSC Node Slovakia operates as a pilot entry point under the auspices of the Slovak Centre of Scientific and Technical Information (CVTI SR), which serves as the National Coordinator of Open Science in Slovakia and an EOSC Association Mandated Organization [11].

The node is starting to offer research information systems (KOMIS), a data management platform, electronic information resources, and national registries, with related connection to HPC infrastructure under development. Its use case portfolio includes the AMR Federated Pathogen Analysis implementing cross-border bioinformatics workflows across institutions in multiple countries [12].

B. Motivation for Federated SDGaaS

EOSC Node Slovakia illustrates a gap characteristic of the broader ecosystem: infrastructure for cataloguing research outputs exists, yet no mechanism for generating new data on demand is available. The AMR use case demonstrates institutional capability for cross-border federated workflows, and Federated SDGaaS would extend this pattern to synthetic data generation with the added benefit that synthetic data avoids the privacy constraints that make real data sharing difficult.

The EU AI Act creates regulatory demand for synthetic data as a compliance mechanism, particularly for bias detection in high-risk AI systems [3], yet smaller national nodes may lack the scale to build bespoke generation infrastructure. A federated generation service would let any node access capabilities beyond its individual capacity while contributing its own resources. Overall, the evolution from passive data stewardship to active data generation is a recognized next step for the EOSC federation.

III. Analysis of AI Methods for Dataset Generation

Synthetic dataset generation can be approached through two established ways: AI-assisted source code generation, where an AI model produces the program that generates the dataset, and direct AI-based data generation, where an AI model itself produces the synthetic samples. A third way, agentic AI systems, is gaining practical relevance. This section surveys all three and justifies the architectural choice proposed for Federated SDGaaS.

A. AI-Assisted Source Code Generation

Large Language Models have shown strong capability in generating functional source code. By early 2026, the landscape has matured considerably. Jiang et al. benchmarked thirteen leading models in their ACM survey, with several achieving high functional correctness on standard benchmarks [13]. Frontier models including Claude Opus 4.6, GPT-5.3, and Gemini 3.0 Pro and their successors have since achieved competitive results on complex specification-driven software construction tasks [14].

Nadas et al. reviewed how LLMs are transforming synthetic data creation through prompt-based generation, retrieval-augmented pipelines, and iterative self-refinement [6]. Specialized frameworks such as Curator, Distilabel, and Augmentoolkit now provide infrastructure for LLM-driven dataset creation through agentic workflows [15].

When applied to dataset generation, an LLM receives a specification describing the desired schema, statistical distributions, and domain constraints, then produces a self-contained program capable of generating the dataset. The generated code is fully inspectable and auditable, and it is reproducible given the same random seed. The source code itself constitutes a research product that can be shared, cited, and reused independently of the data it produces.

B. Direct AI-Based Data Generation

A different approach uses generative models to directly synthesize data samples. Shi et al.'s 2025 survey organizes the field into statistical methods, GAN-based approaches, VAE variants, diffusion models, and transformer-based methods [16]. Recent advances include encoder-decoder transformer denoisers for tabular data [17], Diffusion Transformers for heterogeneous time-series [18], the TTVAE architecture for complex distributional patterns [19], and the TabularARGN framework with formal differential privacy guarantees [20].

LLMs can also directly generate samples by treating data generation as text completion, which is effective for data requiring world knowledge such as clinical notes. However, direct methods typically require representative training data and produce outputs through an opaque process. They also need specialized and high-performance GPU hardware for inference.

C. Agentic AI Systems

Beyond these two approaches, agentic AI systems offer a third option for data generation tasks. Recent surveys characterize LLM-based multi-agent systems as architectures where specialized agents collaborate through planning, tool use, and iterative self-correction [21]. Orchestration frameworks such as AutoGen, CrewAI, and LangGraph enable the composition of multi-agent pipelines, while autonomous coding agents such as Devin, OpenHands, and Claude Code demonstrate that end-to-end software engineering tasks can be handled with minimal human oversight.

In the context of synthetic dataset generation, an agentic system could interpret a dataset specification, generate code, execute it in a sandbox, evaluate output quality against statistical criteria, and revise the code iteratively until thresholds are met. A dual-agent design is particularly promising: a Generator Agent that writes and refines the code paired with an Evaluator Agent that autonomously runs statistical tests against the user's specification parameters. Such a system can iterate and self-correct without requiring the researcher to possess advanced software engineering skills, and aligns with the feedback loop envisioned in Federated SDGaaS.

D. Justification of the Code Generation Approach

Federated SDGaaS is proposed to adopt AI-assisted source code generation as its primary mechanism. Many scientific use cases involve creating data according to theoretical specifications, for example generating molecular structures with specific properties or simulating sensor readings under defined conditions, where no training data exists for direct methods.

The code generation approach aligns naturally with the FAIR principles [22]: the generated code is findable, accessible, interoperable, and reusable. It produces multiple research products (the generator software and the dataset), and the lightweight generated code, typically just kilobytes, can be transmitted to any federated node for execution. Direct generation models, by contrast, require gigabytes of model weights and co-located GPU hardware.

Several commercial and open-source platforms for synthetic data generation exist, including Gretel, Mostly AI, NVIDIA's tools, and the Synthetic Data Vault [23]. These platforms primarily target enterprise use cases with direct generation methods and operate as standalone products. Federated SDGaaS differs by focusing on federated scientific infrastructure, code-based generation for auditability, and integration with the EOSC research product lifecycle.

The proposed architecture does not preclude direct generation methods. The LLM-generated source code may itself invoke diffusion model libraries or LLM APIs. In this sense, the LLM serves as a meta-generator: it generates the generator, providing a uniform and auditable interface regardless of the underlying synthesis technique. Future integration of agentic capabilities, such as the dual-agent architecture described in Section III-C, could further enhance this process through automated multi-step refinement.

IV. Proposed Solution Design

This section presents the proposed Federated SDGaaS architecture, organized around five abstract federated node roles (A through E). Any EOSC national or thematic node could fulfill one or more of these roles depending on its capabilities. The architecture concept (Fig. 1) emphasizes reproducibility, traceability through persistent identifiers (PIDs), and production of multiple reusable research products. Figs. 2–6 present mockup designs illustrating how the workflow could function.

A. Architecture Overview

The proposed pipeline would transform a researcher's natural language dataset specification into a validated, full-scale synthetic dataset deposited in a research products repository and registered in EOSC Federation Catalogues. Five stages are envisioned: (1) specification capture and source code generation, (2) sample validation with iterative refinement, (3) containerized full-scale execution, (4) distributed storage, and (5) multi-product deposition and cataloguing. Each stage is represented by an abstract node role communicating through specified interfaces, enabling deployment across different infrastructure providers.

B. User Interface and Dataset Specification

The proposed entry point is a user interface through which an EOSC researcher provides the dataset specification via natural language description, optional sample data upload, and metadata requirements (licensing, provenance, compliance). Fig. 2 shows a mockup of this specification interface.

C. Node Role A: Source Code Generation

The first node role would host a source code generator powered by a Large Language Model, receiving the dataset specification and producing the source code of a dataset generator (typically a Python script or Jupyter Notebook). Generated code would be restricted to a curated set of approved libraries to ensure reliable execution and prevent supply-chain risks, with security sandboxing preventing external network access.

The generation process is envisioned as a multi-stage pipeline: initial code generation, static analysis, automated sandbox testing, and iterative refinement where errors are fed back to the LLM. Fig. 3 illustrates a mockup of the generated code.

D. Node Role B: Sample Generation and Validation

The second node role implements the human-in-the-loop quality assurance process. The generated source code from Node Role A would be executed to produce a small batch of synthetic samples sufficient for the researcher to evaluate output quality.

This node role would provide sample inspection through an interactive interface with statistical summaries, distribution plots, and quality metrics, along with an integrated code editor for direct modifications. A feedback loop connecting back to Node Role A forms the defining iterative mechanism: the researcher provides natural language feedback triggering new code generation iterations until the output is approved. The validated source code output could receive a persistent identifier (PID n.1).

As a further enhancement, this node could employ the dual-agent architecture from Section III-C, where the Generator Agent and Evaluator Agent iterate autonomously before presenting results to the researcher, reducing the number of manual review cycles required. Fig. 4 shows a mockup of the validation interface.

E. Node Role C: Orchestration and Execution

Once approved, Node Role C acts as an orchestration layer that distributes the validated source code alongside a lightweight environment manifest (e.g., a Dockerfile or requirements.txt) to the target computing node. Rather than transmitting pre-built container images across the federation, the actual container compilation occurs locally at the execution site, leveraging local caching layers to minimize network transit.

This approach preserves the bandwidth efficiency of distributing kilobyte-sized source code and manifests while still ensuring reproducible execution environments through version-pinned dependencies and cryptographic verification.

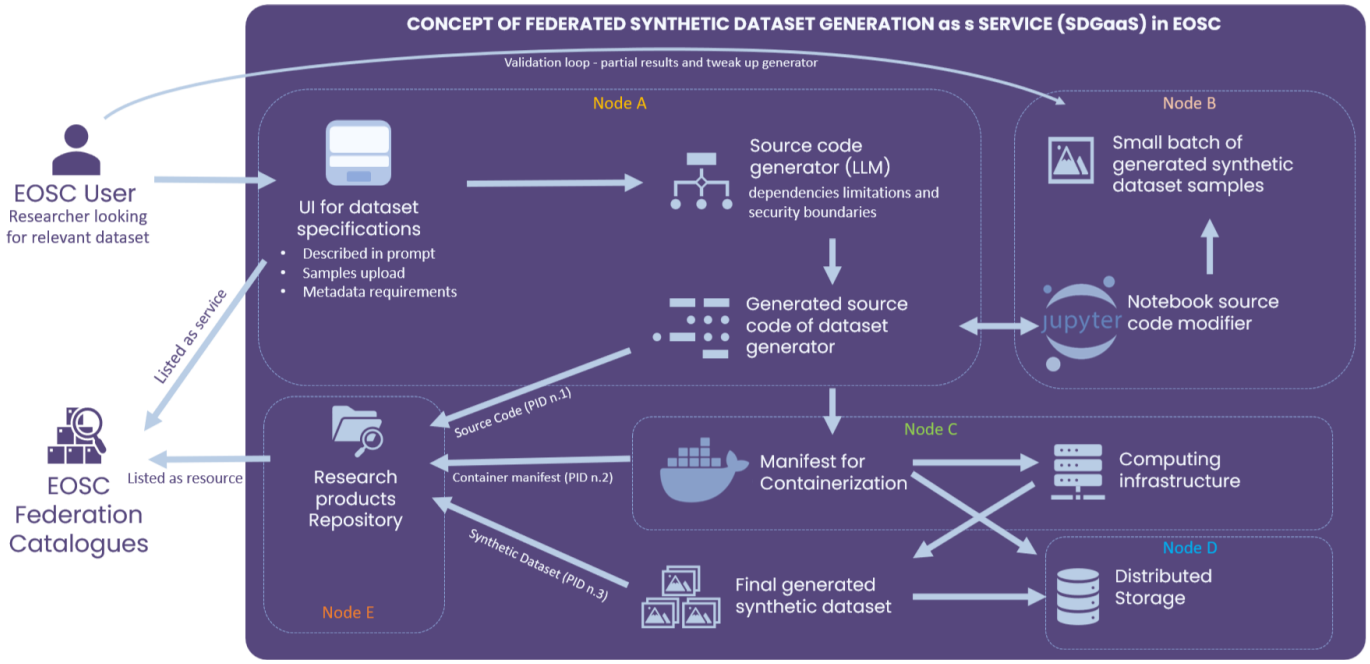

Fig. 1. Proposed Federated SDGaaS architecture showing five abstract federated node roles (A through E), the human-in-the-loop feedback cycle, persistent identifier (PID) traceability, and envisioned integration with EOSC Federation Catalogues.

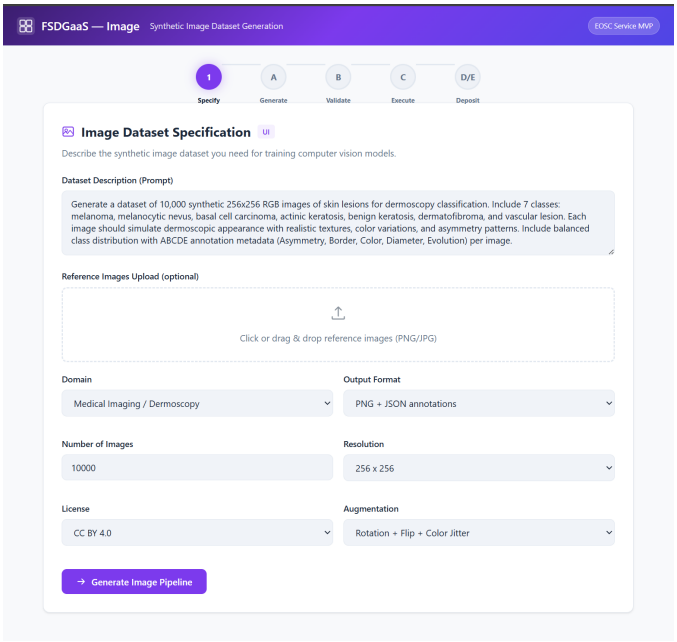

Fig. 2. Federated SDGaaS mockup design – Step 1 (Specify): The researcher defines the dataset specification including a natural language prompt, domain selection, output format, and licensing metadata.

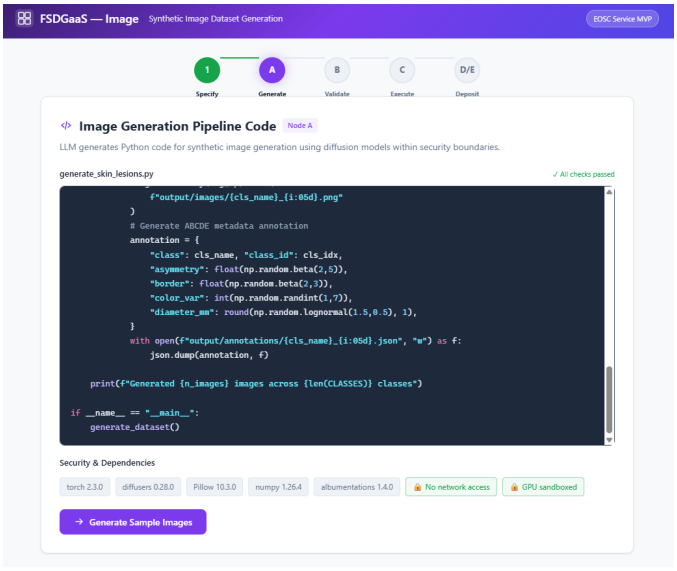

Fig. 3. Federated SDGaaS mockup design – Node Role A (Generate): LLM-generated Python code for synthetic image generation with dependency and security checks. PID n.1 is assigned to the source code.

The resulting container would be deployed on EOSC federation computing infrastructure, whether HPC clusters or cloud resources, with a scheduling layer matching jobs to resources. Fig. 5 presents a mockup of the orchestration and execution.

F. Node Roles D and E: Storage, Repository, and EOSC Integration

The fourth node role would provide distributed storage for the final dataset, ensuring reliability through replication across storage nodes (f.e. ONEDATA - onedata.org) and compliance with data jurisdiction requirements.

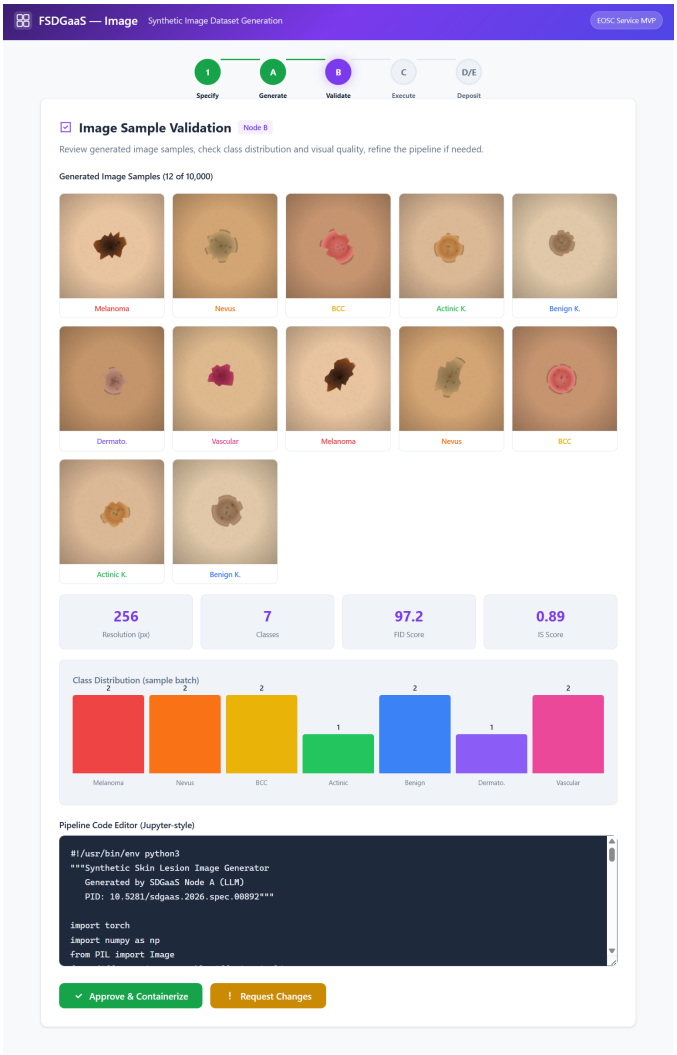

Fig. 4. Federated SDGaaS mockup design – Node Role B (Validate): Sample inspection with generated images, quality metrics, class distribution analysis, and an integrated code editor for iterative refinement.

The fifth node role would host a research products repository, interfacing with established repositories (such as Zenodo or B2SHARE) within the EOSC ecosystem. Federated SDGaaS is designed to produce up to three research products from a single request:

1) The source code of the dataset generator (PID n.1), deposited as a reusable software artifact.
2) The environment manifest packaging (for containerization) the validated generator (PID n.2), enabling exact reproduction of the execution environment.
3) The generated synthetic dataset (PID n.3).

PID assignment and repository deposition are not mandatory for every generated artifact. To avoid overloading repositories with experimental or low-quality outputs, deposition would be curated through quality assessment: only artifacts that pass defined quality thresholds and that the researcher explicitly marks for publication would receive persistent identifiers and be registered in EOSC catalogues. All deposited products

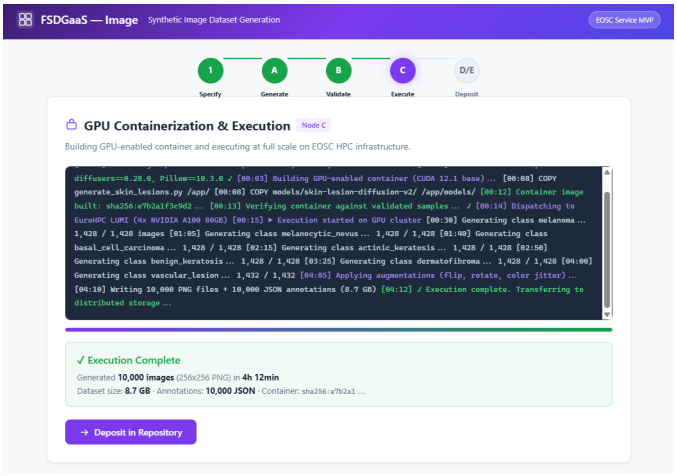

Fig. 5. Federated SDGaaS mockup design – Node Role C (Execute): Orchestration and full-scale execution on federated HPC infrastructure, showing build logs and execution progress.

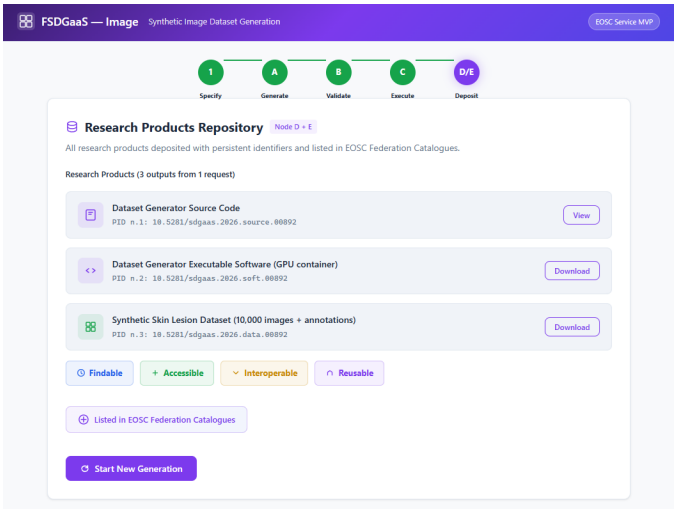

Fig. 6. Federated SDGaaS mockup design – Nodes D+E (Deposit): The research products repository listing all three outputs from a single generation request, each assigned a persistent identifier.

would be listed as resources in the EOSC Federation Catalogues, indexed with metadata, PIDs, and provenance links. The links between PIDs would create a coherent package allowing researchers to trace from any artifact to the others. Fig. 6 shows a mockup of the repository view.

G. Federated Distribution

The five node roles need not reside on the same infrastructure or in the same country. Any EOSC node could take on one or more roles depending on its resources. Inter-node communication would follow a message-passing architecture with standardized API contracts, using PIDs as the primary linking mechanism. This also addresses data sovereignty concerns, as sensitive artifacts could be processed within specific jurisdictions.

V. Conclusion

The Federated SDGaaS concept addresses a gap in the research data ecosystem: the inability to generate, rather than merely discover, datasets on demand. By combining LLM-powered code generation with containerized execution on federated infrastructure, the proposed architecture aims to be flexible and reproducible while remaining transparent. Human-in-the-loop validation with its iterative feedback loop would mitigate the primary risk of AI-generated code, while the proposed dual-agent architecture could further reduce the need for manual intervention by automating quality assessment. The federated design would enable deployment across the heterogeneous EOSC landscape.

A distinctive aspect is the production of up to three citable, FAIR-compliant research products from a single request: the source code (PID n.1), the containerization environment manifest (PID n.2), and the generated synthetic dataset (PID n.3), aligning with the recognition of research software as a first-class scholarly output [22]. Federated SDGaaS also proposes to go beyond FAIR by introducing generative FAIRness: when a researcher cannot find a suitable dataset in any catalogue, the service would enable them to create what they cannot find.

Future work would focus on prototype implementation and empirical validation of the proposed architecture, development of cost models for federated computation, and governance frameworks for generated data quality and provenance. Security considerations, including code injection risks, LLM prompt manipulation, supply-chain attacks via generated libraries, and adversarial uses of synthetic data, will require dedicated threat modeling as part of any production deployment [24]. Additional directions include automated quality assurance with statistical fidelity tests, analysis of domain coverage limitations, a generator marketplace for sharing parameterized generators, and deeper integration with domain-specific ontologies as agentic AI systems mature.

Acknowledgment

Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I02-03-V01-00029. The author acknowledges the use of AI writing assistance for language and stylistic refinement of this manuscript, in line with emerging academic norms for transparent disclosure of AI tools in scholarly writing.

References

[1] World Economic Forum, "Synthetic Data: The New Data Frontier," Briefing Paper, Global Future Council on Data Frontiers, Sep. 2025. [Online]. Available: https://reports.weforum.org/docs/WEF_Synthetic_Data_2025.pdf

[2] World Economic Forum, "AI training data is running low – but we have a solution," Dec. 2025. [Online]. Available: https://www.weforum.org/stories/2025/12/data-ai-training-synthetic/

[3] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act)," *Official Journal of the European Union*, L series, 2024.

[4] A. Guadamuz, "The EU's Artificial Intelligence Act and copyright," *J. World Intellectual Property*, vol. 28, no. 1, pp. 213–219, 2025, doi: 10.1111/jwip.12330.

[5] E. H. Wang *et al.*, "Crossing borders securely: synthetic data and federated networks for privacy-preserving access to real-world data and emerging use cases," *npj Digital Medicine*, vol. 8, art. 758, 2025, doi: 10.1038/s41746-025-02126-8.

[6] M. Nadas, L. Diosan, and A. Tomescu, "Synthetic Data Generation Using Large Language Models: Advances in Text and Code," *arXiv preprint arXiv:2503.14023*, 2025, doi: 10.48550/arXiv.2503.14023.

[7] J. Morales-Garcia *et al.*, "Exploiting synthetic data generation to enhance pollution prediction," *Applied Soft Computing*, vol. 175, art. 113076, 2025, doi: 10.1016/j.asoc.2025.113076.

[8] X. Zhao, K. W. Fok, and V. L. L. Thing, "Enhancing network intrusion detection performance using generative adversarial networks," *Computers & Security*, vol. 146, art. 104005, 2024, doi: 10.1016/j.cose.2024.104005.

[9] D. B. Resnik, M. Hosseini, J. J. H. Kim, G. Epiphaniou, and C. Maple, "GenAI synthetic data create ethical challenges for scientists," *Proc. Natl. Acad. Sci. USA*, vol. 122, no. 9, 2025, doi: 10.1073/pnas.2409182122.

[10] European Commission, "European Open Science Cloud (EOSC)," 2025. [Online]. Available: https://eosc.eu

[11] EOSC Node Slovakia, "Pilot Entry Point of Slovak National EOSC Node," CVTI SR, 2025. [Online]. Available: https://www.eosc.sk

[12] D. Cymbalak and J. Budis, "EOSC Node – Slovakia," presented in expo booth at the EOSC Symposium 2025, Brussels, Belgium, Nov. 3–5, 2025. [Online]. Available: https://indico.cern.ch/event/1543880/contributions/6751326/attachments/3171286/5638055/EOSC%20Node%20-%20Slovakia.pdf

[13] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A Survey on Large Language Models for Code Generation," *ACM Trans. Softw. Eng. Methodol.*, vol. 35, no. 2, art. 58, 2026, doi: 10.1145/3747588.

[14] Z. Zhang *et al.*, "SWE-AGI: Benchmarking Specification-Driven Software Construction with MoonBit in the Era of Autonomous Agents," *arXiv preprint arXiv:2602.09447*, 2026, doi: 10.48550/arXiv.2602.09447.

[15] A. Alismail and C. Lanquillon, "A survey of LLM-based methods for synthetic data generation and the rise of agentic workflows," in *Artificial Intelligence in HCI (HCII 2025)*, LNCS, vol. 15821. Cham, Switzerland: Springer, 2025, pp. 119–135, doi: 10.1007/978-3-031-93418-6_9.

[16] R. Shi *et al.*, "A Comprehensive Survey of Synthetic Tabular Data Generation," *arXiv preprint arXiv:2504.16506*, 2025, doi: 10.48550/arXiv.2504.16506.

[17] M. Villaizan-Vallelado *et al.*, "Diffusion Models for Tabular Data Imputation and Synthetic Data Generation," *ACM Trans. Knowl. Discov. Data*, vol. 19, no. 6, 2025, doi: 10.1145/3742435.

[18] F. Garuti *et al.*, "Diffusion Transformers for Tabular Data Time Series Generation," *arXiv preprint arXiv:2504.07566*, 2025, doi: 10.48550/arXiv.2504.07566.

[19] A. X. Wang and B. P. Nguyen, "TTVAE: Transformer-based generative modeling for tabular data generation," *Artificial Intelligence*, vol. 340, art. 104292, 2025, doi: 10.1016/j.artint.2025.104292.

[20] A. Sidorenko and P. Tiwald, "Privacy-Preserving Tabular Synthetic Data Generation Using TabularARGN," *arXiv preprint arXiv:2508.06647*, 2025, doi: 10.48550/arXiv.2508.06647.

[21] S. Chen, Y. Liu, W. Han, W. Zhang, and T. Liu, "A survey on LLM-based multi-agent system: Recent advances and new frontiers in application," *arXiv preprint arXiv:2412.17481*, 2025, doi: 10.48550/arXiv.2412.17481.

[22] E. A. Jensen and D. S. Katz, "Awareness of FAIR and FAIR4RS among international research software funders," *Scientific Data*, vol. 12, art. 627, 2025, doi: 10.1038/s41597-025-04820-4.

[23] A. Lautrup, T. Hyrup, A. Zimek, and P. Schneider-Kamp, "Systematic Review of Generative Modelling Tools and Utility Metrics for Fully Synthetic Tabular Data," *ACM Comput. Surv.*, vol. 57, no. 4, 2025, doi: 10.1145/3704437.

[24] S. Gulyamov *et al.*, "Prompt injection attacks in large language models and AI agent systems: A comprehensive review," *Information*, vol. 17, no. 1, art. 54, 2026, doi: 10.3390/info17010054.