



MDMC

Master in Data Management
and Curation

MASTER IN DATA MANAGEMENT AND
CURATION

Implementing a FAIR-by-Design
Approach for Process Management
in Nanoscience Foundries

APPLICATION IN THE CNR-ISMN CLEANROOM

Supervisors:

Piera MACCAGNANI,
Stefano ZAMPOLLI,
Mariarita DE LUCA

Candidate:

Mattia MARELLA

2024–2025





Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA

This Pilot training activity has been funded by the European Union – NextGenerationEU within the project PNRR "PRP@CERIC" IR0000028 and «NFFA-DI" IR0000015 - Missione 4, "Istruzione e Ricerca" – Componente 2, "Dalla ricerca all'impresa" – Linea di investimento 3.1, "Fondo per la realizzazione di un sistema integrato di infrastrutture di ricerca e innovazione" – Azione 3.1.1, "Creazione di nuove IR o potenziamento di quelle esistenti che concorrono agli obiettivi di Eccellenza Scientifica di Horizon Europe e costituzione di reti".

The supporting projects:

- [NFFA-DI : Nano Foundries Fine Analysis - Digital Infrastructure](#)
- [PRP@CERIC : Pathogen Readiness Platform for Ceric - Eric Upgrade](#)



Author's Declaration

I, Mattia Marella, declare that this thesis entitled, 'Implementing a FAIR-by-design approach for Process Management in Nanoscience Foundries. Application in the CNR-ISMN cleanroom.' and the work presented therein are my own.

I certify that:

- This work was performed wholly or principally during MDMC internship at CNR-ISMN (Bologna).
- If any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have relied on the published work of others, this is always clearly attributed.
- Where the thesis is based on work I have done in collaboration with others, I have made clear exactly what was done by others and what I contributed.
- With respect to AI technologies:
 - I acknowledge the use of ChatGPT (<https://chatgpt.com/?model=gpt-4o>) to identify improvements in the writing style.
 - I acknowledge the use of ChatGPT (<https://chatgpt.com/?model=gpt-4o>) as a source of information to create materials that will be used in my own words.
- Where I have quoted from the work of others, the source is always cited. Except for such quotations, this thesis is entirely my own work.
- I have acknowledged all major sources of help.

Signed:



Date:

April 30, 2025

“Il buon senso c’era; ma se ne stava nascosto, per paura del senso comune.”

– Alessandro Manzoni

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Piera Maccagnani, for her constant guidance and her constructive feedback. Her expertise and encouragement made this thesis not only possible but deeply enriching.

I am also grateful to Dr. Stefano Zampolli and Dr. Ivan Elmi for their technical insights and for generously sharing their knowledge with me. I feel happy and honored to have had the opportunity to work in such an inspiring and brilliant team.

I would also like to extend my heartfelt thanks to Mariarita de Luca and Federica Bazzocchi for their invaluable coordination of the MDMC program and their continuous support throughout the duration of the internship. Their dedication ensured a smooth and enriching experience.

Thanks to all the beautiful people who pursued this master with me; this was a truly unique journey, and sharing it with you made it unforgettable.

Thanks to Matteo Bontorno for the amazing work we carried out together. When I was struggling with the NOMAD setup, I knew I wasn't alone in that battle. I'm proud of what we've achieved.

Thanks to Giovanni Gallerani, who embraced this challenge just like I did, and proved to be—not just a classmate—but a great travel companion and an even better roommate.

A special thanks also goes to our professor, Maddalena Nonato, who, unlike many colleagues, shared this extraordinary opportunity with us so we could fully live it. A truly FAIR-minded gesture!

Abstract

The exponential growth of data generated in nanoscience research demands robust frameworks for data management, sharing, and reusability. This thesis investigates the implementation of a FAIR-by-design approach specifically tailored for process management within nanoscience foundries, emphasizing its application at the CNR-ISMN cleanroom facility.

In this context, the NFFA-DI (Nano Foundries Fine Analysis - Digital Infrastructure) initiative plays a pivotal role by aiming to create a centralized and FAIR-by-design ecosystem for managing experimental data across a network of nanoscience laboratories. Such an infrastructure is crucial to ensure standardized data practices, foster interoperability, and maximize the scientific value of research outputs.

Starting from the foundational FAIR principles, the work addresses the challenges associated with standardizing process documentation in nanofabrication. A comprehensive hierarchical taxonomy was developed, combining international standards and community-defined terminologies to foster semantic clarity and interoperability across laboratories. Furthermore, an existing laboratory management system, CAMS, was adapted to produce FAIR-compliant data outputs through structured JSON reports, facilitating seamless integration with external repositories.

The research culminated in the creation of a specialized plugin for the NOMAD repository, enhancing its capability to handle nanofabrication data effectively. The implemented solution supports detailed metadata extraction and promotes automated, standardized data sharing practices. This integration not only improves data visibility and reusability but also sets a foundation for ongoing refinement and community collaboration, thus significantly advancing the FAIR principles within nanoscience infrastructure.

Contents

1	Introduction	8
1.1	Timeline	8
1.2	Challenges and Considerations	10
1.3	FAIR as <i>Fully AI Ready</i>	12
1.3.1	Data FAIRness	12
1.3.2	Software FAIRness	13
1.4	Overview	14
2	The laboratory: CNR-ISMN	15
2.1	The Cleanroom	15
2.2	CAMS	16
2.2.1	Jam.py	16
2.2.2	What is CAMS	18
2.2.3	Data Organization	20
2.2.4	Reports	22
2.3	Recipes	22
3	The vocabulary	24
3.1	Pre-existent Literature	25
3.1.1	MDMC-NEP Glossary of Terms	25
3.1.2	ISO/TS 80004-1	26
3.1.3	NanoFabNet	27
3.2	Hierarchical Taxonomy	27
3.2.1	Superclasses	28
3.2.2	Classes	29
3.2.3	Technologies	31
3.3	Considerations	31
4	A FAIR Workflow	32
4.1	NOMAD	32
4.1.1	Core Functionalities	33

4.1.2	Role in NFFA-DI	35
4.2	The Plugin	35
4.2.1	The Structure	36
4.2.2	Schemas	38
4.3	CAMS Upgrade Overview	40
4.3.1	APIs	41
4.3.2	Data Transformation	42
	Final Remarks and Prospects	45
	A JSON Report	47
	B Technologies	51
	C NOMAD-ready files	53

Chapter 1

Introduction

“There is an urgent need to improve the infrastructure supporting the reuse of scholarly data.”

So began the seminal paper titled *The FAIR Guiding Principles for scientific data management and stewardship* [1], published in 2016. This landmark work addressed the escalating challenges associated with the exponential growth of data in scientific research, providing foundational guidelines to improve how data is organized, shared, and preserved for future use.

With swiftly evolving methodologies and increasingly sophisticated computational capabilities, the volume and intricacy of generated data have outpaced conventional management and preservation approaches.

The FAIR principles—advocating that data should be Findable, Accessible, Interoperable, and Reusable—were proposed as a structured framework to elevate the quality, effectiveness, and reproducibility of research outputs. Since their inception, these guidelines have substantially reshaped data handling practices across diverse scientific disciplines, inspiring an international movement towards uniform and sustainable data protocols.

This thesis critically examines the adoption and application of the FAIR principles, exploring the technological advancements, cultural shifts, and institutional reforms necessary to fully harness scholarly data’s potential in expediting scientific innovation, with a specific focus on the implementation at the CNR-ISMN Laboratory in Bologna.

1.1 Timeline

Since their introduction in 2016, the FAIR principles have significantly influenced data management and stewardship across various scientific disciplines.

Here is an overview of their evolution and adoption over the years:

2016: Introduction and Immediate Endorsement While the concept of data sharing had been gaining traction, 2016 marked a significant paradigm shift with the formal publication of the principles. They underscored the necessity for data to be not only accessible to humans but also machine-actionable, that is, structured in a way that allows computational systems to automatically find, access, interoperate, and reuse data with minimal human intervention. This shift was driven by the increasing reliance on computational tools to manage the growing volume and complexity of scientific data. The FAIR principles quickly gained international recognition, with endorsements from major bodies such as the G20, advocating for their adoption to enhance the accessibility and utility of research data [1–3].

2017–2018: Establishment of Support Structures In 2017, Germany, the Netherlands, and France collaborated to establish the GO FAIR International Support and Coordination Office, aiming to promote and facilitate the adoption of FAIR principles globally. During this period, organizations like CODATA and the *Research Data Alliance* (RDA) also began supporting FAIR implementations within their communities, further solidifying the principles’ international standing [2].

2019: Expansion to Research Software Recognizing the importance of software in research, the FAIR principles were extended to encompass research software, leading to the development of the *FAIR for Research Software* (FAIR4RS) principles. This extension aimed to ensure that software, like data, is findable, accessible, interoperable, and reusable, thereby enhancing transparency and reproducibility in computational research [4].

2020–2022: Implementation and Case Studies The main focus shifted to practical implementation, with numerous case studies demonstrating the application of these principles across disciplines. For instance, the FAIR-IMPACT project identified practices, tools, and technical specifications to guide various stakeholders toward a FAIR data management cycle, addressing challenges in fields such as social sciences, humanities, life sciences, and environmental sciences [5].

2023: Ongoing Initiatives and Future Directions Efforts to promote FAIR data sharing continue to evolve. Workshops and initiatives have been organized to explore new strategies for supporting FAIR data

sharing, addressing potential obstacles, and identifying innovative approaches to overcome them. These endeavors aim to accelerate the adoption of FAIR principles and deepen their integration into research practices [6].

2025: Standardization in Nanofabrication FAIR principles were implemented within nanofabrication laboratories to enhance data sharing and reproducibility. Collaborative efforts with many consortia, such as CNR and NFFA-DI, contributed to the establishment of standardized vocabularies for nanofabrication processes, addressing the lack of a common terminology in the field.

1.2 Challenges and Considerations

Despite widespread endorsement, challenges remain in fully implementing the FAIR principles. These include ensuring data quality, developing supportive infrastructures, and fostering a cultural shift toward data sharing and openness among researchers. Ongoing efforts are focused on addressing these challenges to realize their full potential in enhancing data-driven research [7].

Applying these guidelines presents several challenges. One significant obstacle is the lack of standardized metadata practices across diverse scientific domains, which hampers the development of universal tools for data discovery and integration. Additionally, ensuring interoperability necessitates the adoption of common vocabularies and ontologies, a process that can be resource-intensive and may encounter resistance due to established disciplinary conventions.

Furthermore, the emphasis on machine-actionability has sometimes overshadowed the need for human-actionability, highlighting the importance of enhancing human understanding of FAIRness criteria to effectively implement these principles. Addressing these challenges requires coordinated efforts to develop community standards, invest in infrastructure that supports both machine and human interactions with data, and promote education on FAIR principles to facilitate widespread adoption [8].

Some of the most common challenges are:

Data Fragmentation Data are often scattered across various platforms, databases, and file formats, making it difficult to locate and access. Non-standardized metadata and inconsistent data organization further hinder effective data discovery and reuse. [9]

Limited Accessibility Access to data may be restricted due to proprietary constraints or privacy concerns, impeding collaboration and reproducibility. Legal and ethical considerations, along with concerns about data security and intellectual property, can pose barriers to implementing FAIR data practices. [9]

Interoperability Issues Incompatibility between different software systems, tools, and formats complicates data integration. The absence of standardized data models, ontologies, and controlled vocabularies impedes the exchange and integration of FAIR data across disciplines and research domains. [9]

Data Quality and Documentation Inadequate documentation, incomplete metadata, and inconsistent data formats affect data quality and reliability. Poor documentation hampers understanding of the data context, making validation and interpretation challenging. [9]

Resource Constraints Implementing FAIR principles requires significant time and resources. Many organizations may lack the necessary infrastructure, funding, or expertise to undertake the complex task of FAIRification. [10]

Cultural Resistance Transitioning to FAIR data practices often necessitates a cultural shift within organizations. Resistance to change and reluctance to share data openly can impede the adoption of FAIR principles. [10]

To address these challenges, organizations can:

Develop Clear Data Governance Policies Establishing comprehensive data governance frameworks ensures clarity on data ownership, accessibility, and management, facilitating the implementation of these new guidelines. [11]

Invest in Standardization Efforts Adopting and promoting standardized data models, metadata schemas, and ontologies enhances interoperability and data integration across diverse systems and disciplines. [12]

Enhance Data Documentation and Metadata Providing detailed documentation and rich metadata improves data quality and usability, enabling better understanding and reuse of data. [13]

Allocate Resources for FAIR Implementations Securing funding and investing in necessary infrastructure and training supports the effective adoption of FAIR data practices. [10]

Foster a Culture of Data Sharing Encouraging collaboration and openness within the research community helps overcome cultural barriers and promotes the benefits of FAIR data practices. [14]

As we will see, all of this solution has been adopted in the CNR-ISMN laboratory. In the next chapters, a detailed description will be provided.

1.3 FAIR as *Fully AI Ready*

In a future where Artificial Intelligence becomes increasingly integral to various aspects of society, adherence to FAIR principles will be crucial for ensuring effective, reliable, and ethically grounded AI systems. Achieving truly trustworthy AI requires applying these principles comprehensively: from initial data management practices through to model development. This section explores the significance of such practices, starting with data considerations and progressing to AI models.

Although the primary focus of this thesis is on FAIR-by-design data management rather than on AI model development, enhancing the machine interoperability and reusability of data is a foundational step toward fully automated, AI-driven research workflows. Moreover, while FAIR principles are discussed here in relation to the datasets that AI models ingest and the models themselves, the same considerations remain equally valid for any software system managing scientific data. By ensuring that experimental data is structured, accessible, and actionable by machines, the work presented here contributes to building the necessary infrastructure for future trustworthy AI systems and datasets.

1.3.1 Data FAIRness

FAIR data practices ensure that datasets are well-organized and richly described, facilitating seamless integration and analysis by AI algorithms. This structured approach allows AI systems to process information more efficiently, leading to accelerated scientific discoveries and innovations across diverse fields (such as medicine, environmental science, and engineering) [15].

Moreover, the promoted accessibility and transparency are crucial in mitigating biases and enhancing the fairness of AI models, which we'll cover later.

By providing clear documentation and open access to datasets, researchers can scrutinize and address potential biases, leading to more equitable AI outcomes. This openness fosters trust and accountability in AI applications, which is essential as these systems play a more significant role in decision-making processes day by day [16].

Finally, high-quality and interoperable datasets are foundational for training *very* robust AI models. FAIR data practices ensure that datasets are not only accessible but also standardized, allowing AI systems to learn from diverse and comprehensive data sources. This diversity enhances the *generalizability* and performance of AI models, making them more effective in real-world applications.

1.3.2 Software FAIRness

Let's explore how software can be FAIR along all the four principles:

Findable: Ensuring AI models are easily discoverable is crucial. Assigning unique identifiers and rich metadata facilitates efficient retrieval and utilization by researchers and practitioners. For instance, publishing models with comprehensive descriptions and searchable keywords in repositories like the *Data and Learning Hub for Science* (DLHub) or GitHub (if coupled with Zenodo, which provides a DOI) enhances their visibility and adoption [15, 17].

Accessible: Providing clear usage licenses and open access to AI models promotes transparency and inclusivity. When models are accompanied by detailed documentation and hosted on platforms supporting open access, they become more readily available for validation and application in diverse research contexts [16, 17].

Interoperable: Adopting standardized formats and protocols ensures AI models can seamlessly integrate with various systems and datasets. Utilizing common data formats like HDF5 or adhering to established ontologies allows models to communicate effectively across different platforms, facilitating collaborative research and the development of complex, multi-faceted AI systems [17].

Reusable: Comprehensive documentation, including details about the model architecture, training data, and performance metrics, enables others to replicate and build upon existing work. Incorporating uncertainty quantification metrics further informs users about the reliability and applicability of model predictions. Such practices foster trust and

accelerate innovation by reducing redundancy and encouraging the refinement of existing models [17].

Implementing these principles in AI development leads to more robust, transparent, and collaborative models. By ensuring that AI resources adhere to these criteria, the scientific community maximizes the potential of AI technologies, advancing discovery and innovation across various fields [15–17].

1.4 Overview

At the end of this introduction, the chapters are structured as follows: Chapter 2 introduces CAMS, the laboratory LIMS, detailing its functionality and providing practical examples. Chapter 3 discusses the work undertaken to develop a common vocabulary, facilitating interoperability. Chapter 4 describes the creation of a plugin designed for integration with the NOMAD online trusted repository.

Chapter 2

The laboratory: CNR-ISMN

Istituto per lo Studio dei Materiali Nanostrutturati (ISMN) is a research institute of the Italian *National Research Council* (CNR) devoted to advanced materials science and nanotechnology. Founded in 2000, ISMN has grown into an internationally recognized center of excellence in the field of nanostructured materials and enabling materials processes. Starting from February 2025, the section of Bologna of the *Institute for Microelectronics and Microsystems* (IMM Bologna) merged with ISMN, enlarging its technological capabilities and expertise [18].

Its multidisciplinary team integrates chemistry, physics, and engineering expertise to address research themes with high scientific and societal impact. ISMN's work spans from fundamental research – such as the synthesis, modeling, and characterization of novel materials – to applied technology development, including prototype fabrication and industrial collaboration. Key application areas range from organic and hybrid electronics, opto-electronics, and sensors, to energy conversion and environmental sustainability.

2.1 The Cleanroom

A cleanroom is a controlled environment with low levels of pollutants such as dust, airborne microbes, and chemical vapors, essential for the fabrication of micro- and nanostructures. In this project, the cleanroom serves as the operational context where process management practices are standardized and aligned with FAIR principles.

The Bologna headquarters of CNR-ISMN has one of the biggest publicly funded facilities in Italy (the cleanroom covers an area of 500 m², with 100 and 1000 classrooms), with a pilot line which enables research and process development for the prototyping of innovative *micro-* and *nano-electro-mechanical-*

systems (MEMS/NEMS) for chemical and physical sensors, and 3rd generation solar cells and photovoltaic devices. [19]. This cleanroom is equipped with all major technologies for top-down device fabrication – including optical lithography, thin-film deposition (CVD/PVD), and advanced etching (reactive ion etching, etc.) – enabling the production of a variety of micro/nanostructured devices. In particular, the facility supports the fabrication of non-VLSI (non very large scale integration) electronic components and sensors, and innovative photovoltaic devices on silicon, silicon carbide (SiC), and quartz substrates.

In addition to, and in synergic integration with the Si technologies, ISMN Bologna has specific expertise on nanomaterial synthesis: catalytic growth of carbon nanotubes (CNT), graphene, 2D materials, and oxide materials; and on the integration of graphene technology into microelectronics processes for the fabrication of hybrid devices and electronic systems.

The institute’s nanofabrication infrastructure thus provides a cutting-edge platform for prototyping advanced electronic, photonic, and sensing devices in support of its research and technology transfer activities.

The full list of available machinery will be detailed in the next chapter, together with the presentation of a common vocabulary used to describe the key parameters associated with each machine-specific process step.

2.2 CAMS

CAMS is the laboratory management software: it is a tailored web-based platform designed to oversee and document the full workflow of fabrication activities. Before delving into its details, it is important to first introduce Jam.py: the Open Source web framework on which CAMS is built. Understanding the structure and capabilities of this framework provides essential context for appreciating the architecture and functionalities of CAMS.

2.2.1 Jam.py

Jam.py is a free and Open Source web application framework for building data-centric applications with minimal coding. It was first released in 2015 by Andrew Yushev and follows a model–view–controller (MVC) architecture, separating data models, user interface views, and control logic.

The framework is object-oriented and event-driven, featuring a hierarchical, modular design with an intimate coupling between the database and the graphical user interface [20]. Jam.py’s server side is written in Python, while the client side uses JavaScript (leveraging jQuery and Bootstrap for a

responsive single-page interface) [21]. This design, along with the *Don't Repeat Yourself* (DRY) principle, enables Jam.py to serve as a robust low-code/full-stack development platform that reduces repetitive coding in database-driven web projects.

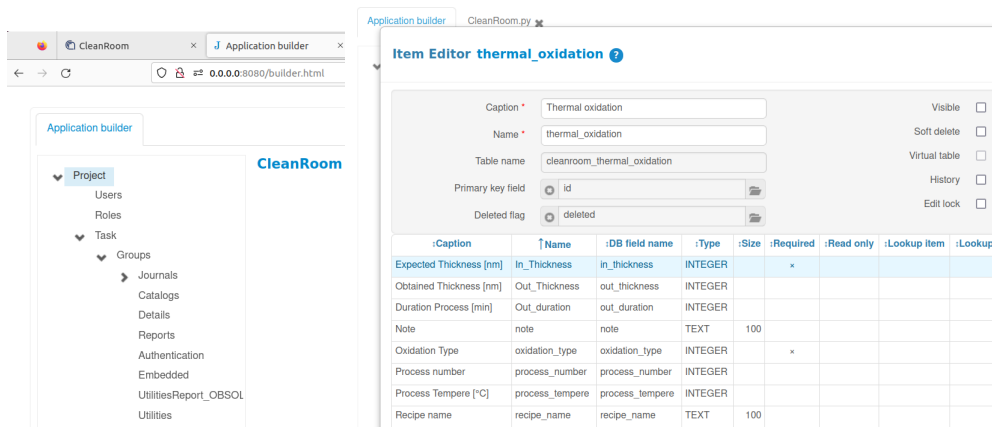


Figure 2.1: Builder interface. Left: the tree structure of a Jam.py project, displaying both administrative modules (e.g., Users, Roles) and data-related sections (Catalogs, Details). Right: an example of a Catalog, the *Thermal Oxidation* step, represented as a table containing the specific parameters associated with this process.

Its most important features adhering to FAIR principles (other than being Open Source) are the following:

- Both the data and the application are reusable assets. Jam.py provides import/export utilities to package the entire application's metadata (data model, configuration, and even custom code) into a single file. This means the lab's *Customer Relationship Management* (CRM) setup can be shared and reused by other research groups or easily replicated in a new environment, preserving the investment in data organization.
- Jam.py's database-agnostic design means the lab is free to use mainstream database systems and can switch to a different backend if needed without altering the application logic. This flexibility facilitates integration with other systems (e.g. linking the CRM database with external analysis tools) since the data resides in standard SQL-based formats. Additionally, Jam.py can exchange data in common formats like JSON for RESTful APIs, allowing the CRM to interoperate with other software [21].

- Data is made accessible through a web interface that any authorized user in the lab can reach via a standard browser. The framework includes role-based access control to manage permissions, ensuring that while the CRM data is readily accessible to the right people, it remains protected from unauthorized access. Moreover, Jam.py’s support for multiple languages and compatibility with mobile devices means the data can be accessed under various usage contexts.

Figure 2.1 provides an overview of the Jam.py graphical backend, showcasing the hierarchical organization of the project modules (such as tasks and users) alongside an example table with its attributes. This illustrates how the framework facilitates the intuitive development of full-stack applications, even for users with limited programming experience.

Overall, Jam.py demonstrates how an Open Source, low-code framework can support FAIR data principles in practice. It allows users to rapidly create database-backed applications (like a lab CRM) that keep data well-structured and accessible, while promoting interoperability and reuse of both the software and the data it manages.

2.2.2 What is CAMS

We may start describing CAMS (Cleanroom Activities Management System) as a lightweight web application designed to organize and document every step in a cleanroom fabrication process, developed by CNR-ISMN to meet typical R&D cleanroom needs—such as managing process flows, recording parameter values, monitoring equipment usage, and producing cost estimates—all in one place. Nonetheless, citing the documentation provided by Stefano Zampolli, the author of this software, conveys a clear and concise idea of what CAMS is (and is not):

- ✓ A straightforward tool to define, record, and coordinate micro/nanofabrication steps and runs.
- ✓ A central point for process documentation, monitoring, and cost reporting.
- ✓ Easily customized for different process flows and cleanroom layouts.
- × A full *LIMS* (Laboratory Information Management System). It does not track consumables or handle detailed equipment maintenance scheduling.

- × A professional security platform: while it can be protected behind credentials and VPNs, anyone needing strict confidentiality for each process step may require additional measures or a different solution.

Several factors support the selection of CAMS, summarized in the five key advantages below.

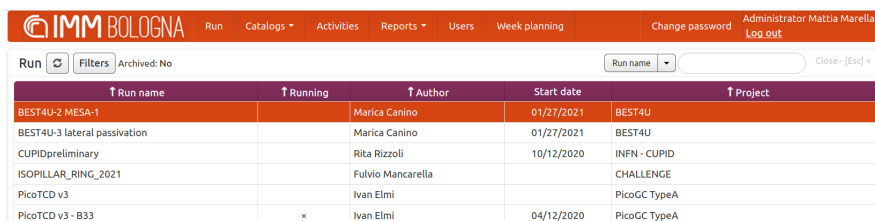
Consistency Mandatory fields keep everyone’s records thorough and uniform.

Searchability All process data, including parameters and results, are stored in a database, allowing users to perform queries to filter, extract, and analyze information, for example, to track how process outcomes evolve over time or across different fabrication recipes.

Efficiency Researchers can clone entire process flows or copy batches of steps from a proven run, rather than recreating them from scratch. This promotes repeatability and saves time.

Reports Built-in cost reporting helps management understand where time and equipment resources go. Data reports provides interoperability, since they use standard file structures.

Collaboration Everyone involved, from research staff to cleanroom technicians, sees the same process flows. This reduces miscommunication and helps ensure a consistent approach.



Run name	Running	Author	Start date	Project
BEST4U-2 MESA-1		Marica Canino	01/27/2021	BEST4U
BEST4U-3 lateral passivation		Marica Canino	01/27/2021	BEST4U
CUPIDpreliminary		Rita Rizzoli	10/12/2020	INFN - CUPID
ISOPILLAR_RING_2021		Fulvio Mancarella		CHALLENGE
PicoTCD v3		Ivan Elmi		PicoGC TypeA
PicoTCD v3 - B33	x	Ivan Elmi	04/12/2020	PicoGC.TypeA

Figure 2.2: The CAMS homepage displays a list of accessible runs along with some of their properties. Additionally, it features a set of tools at the top, such as the ability to generate reports based on the stored data or view a weekly planning to monitor the usage of machines and rooms

2.2.3 Data Organization

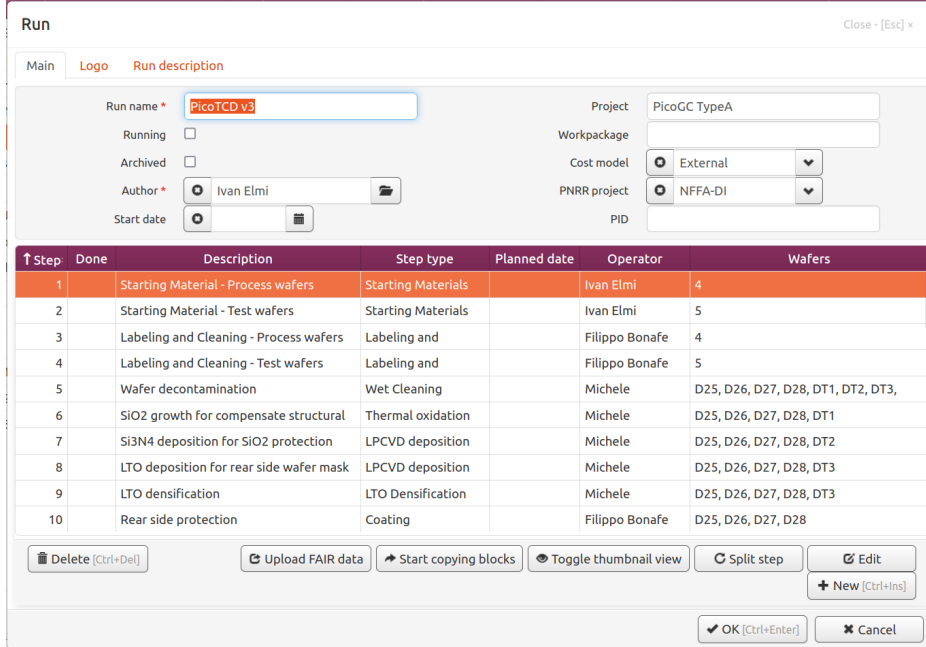
CAMS organizes its data around the concept of a *Run* (see image 2.2), which captures the entire sequence of fabrication *steps* for one or more wafers, from the starting bare material to the final device. Each run may comprise many distinct steps—even hundreds of them—where each one corresponds to a single operation: such as lithography, chemical vapor deposition, or etching.

By structuring information in this manner, CAMS ensures that all critical data is captured consistently, enhancing traceability, enabling seamless integration with other systems, and providing a clear overview of complex workflows in device fabrication.

By examining Figures 2.3 and 2.4, the layout and logic behind CAMS data entry and visualization structure become apparent. CAMS prompts users to record both general and step-specific data throughout the fabrication process. At the run level, this includes information such as the run name, associated project and work package, cost model, author, start date, and a structured workflow listing each process step. Each step entry captures common parameters along with step-specific details. Typically, the initial step defines the nature of the starting wafers, while subsequent steps document the specific actions carried out at each stage. Additional functionalities—such as uploading data to FAIR-compliant repositories, splitting or duplicating steps, and editing entries—enhance flexibility and ensure that all critical process data is consistently captured and easily traceable.

At a more granular level, within a step view:

- The upper section displays parameters common to all step types: description, step type, operator, assigned wafers, planned date, and important notes.
- The top-left corner shows a device image illustrating previously deposited layers and those relevant to the current step.
- The lower section contains step-specific details: deposited material, required thickness, process number, and other fields.
- Additional metadata, such as room details, timing, comments, and file attachments (e.g., process recipes) can be accessed through the corresponding tabs to further enrich the step description.



Run Close - [Esc] x

Main Logo Run description

Run name *

Running

Archived

Author *

Start date

Project

Workpackage

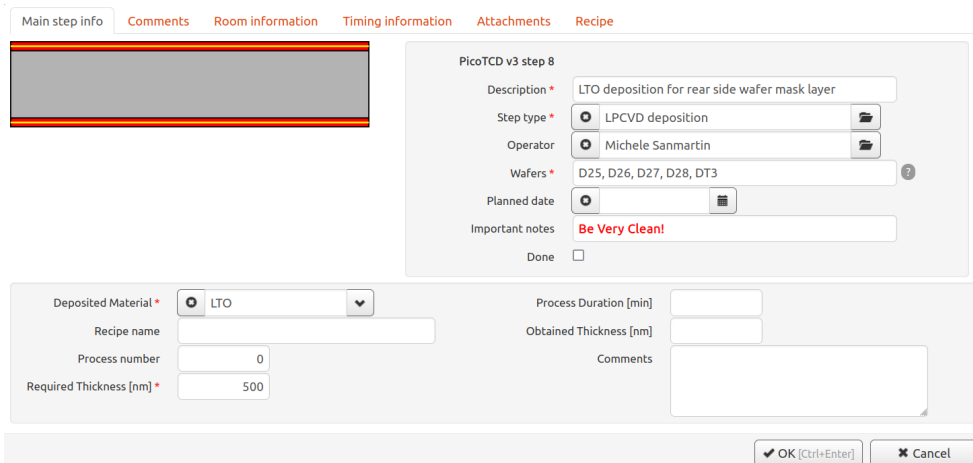
Cost model

PNRR project

PID

Step	Done	Description	Step type	Planned date	Operator	Wafers
1		Starting Material - Process wafers	Starting Materials		Ivan Elmi	4
2		Starting Material - Test wafers	Starting Materials		Ivan Elmi	5
3		Labeling and Cleaning - Process wafers	Labeling and		Filippo Bonafe	4
4		Labeling and Cleaning - Test wafers	Labeling and		Filippo Bonafe	5
5		Wafer decontamination	Wet Cleaning		Michele	D25, D26, D27, D28, DT1, DT2, DT3,
6		SiO2 growth for compensate structural	Thermal oxidation		Michele	D25, D26, D27, D28, DT1
7		Si3N4 deposition for SiO2 protection	LPCVD deposition		Michele	D25, D26, D27, D28, DT2
8		LTO deposition for rear side wafer mask	LPCVD deposition		Michele	D25, D26, D27, D28, DT3
9		LTO densification	LTO Densification		Michele	D25, D26, D27, D28, DT3
10		Rear side protection	Coating		Filippo Bonafe	D25, D26, D27, D28

Figure 2.3: The main run view, where each step in the workflow is listed with relevant properties.



Main step info Comments Room information Timing information Attachments Recipe

PicoTCD v3 step 8

Description *

Step type *

Operator

Wafers *

Planned date

Important notes

Done

Deposited Material *

Recipe name

Process number

Required Thickness [nm] *

Process Duration [min]

Obtained Thickness [nm]

Comments

Figure 2.4: A closer look at a single step, showcasing both general and step-specific fields, along with a visual representation of the device layers involved in the process

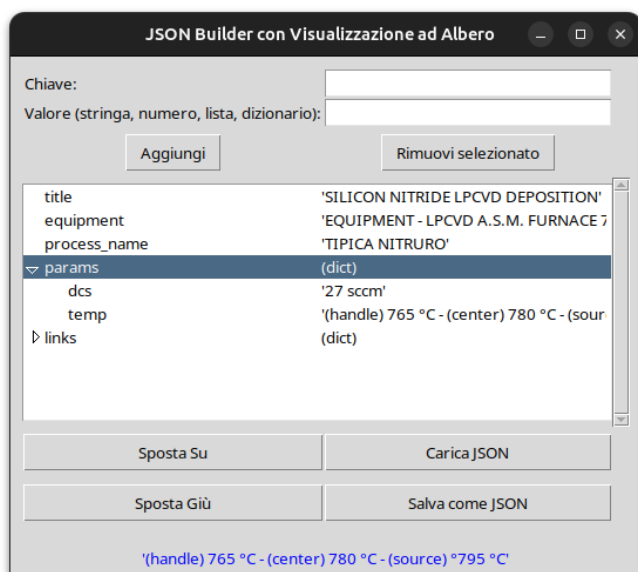


Figure 2.5: The primary (and only) screen of the GUI provides a simple and user-friendly way to create and edit JSON files. Users can add or remove key-value pairs, load existing JSON files, save their work, and even reorder items as needed. Hovering on an entry displays its value at the bottom of the GUI.

2.2.4 Reports

One of the key functionalities supporting FAIR data management in CAMS is the ability to generate comprehensive reports in standardized, interoperable formats. Initially, CAMS only allowed reports to be exported as Microsoft proprietary Word (.docx) files. While convenient for human readability, this format lacks the structured properties necessary for seamless machine processing and data exchange across platforms. To address this limitation and foster greater interoperability, a function was developed to export all recorded information in a JSON file. JSON, as an open and machine-readable format, enables automated data analysis and integration with external systems far more effectively than proprietary or minimally structured formats. In the appendix A, an illustrative example compares the Word export with its JSON counterpart, highlighting the enhanced potential for data discovery, transformation, and reuse that the JSON version offers.

2.3 Recipes

Recipes are critical documents specifying how machines must be configured to execute specific fabrication steps. Each recipe comprises a detailed set of parameters, such as temperature ramp-up and ramp-down profiles, duration intervals, solution proportions, and other process-specific settings that will be thoroughly examined in the following chapter.

Initially, the CAMS platform supported only the recording of recipe

names associated with each process step, without storing or managing the recipe content itself. Recognizing the need to align the workflow with FAIR principles, a decision was made to extend CAMS functionalities. Specifically, an update was implemented allowing users to attach actual recipe documents directly to each step within CAMS, facilitating comprehensive documentation and enhancing data traceability.

However, since the original recipe documents were available exclusively in .docx format, a challenge emerged related to their limited machine-readability and integration potential. To address this limitation and foster automated data extraction and interoperability, the decision was made to transition recipe documents into a structured and open JSON format.

A preliminary step toward this transition involved developing a Python-based application equipped with a graphical user interface (GUI). This software was designed to assist laboratory personnel in creating and validating JSON-formatted recipes effortlessly, even without prior familiarity with JSON syntax. Although the Python GUI tool proved effective and user-friendly, an online alternative eventually replaced it, offering equivalent functionality with the added advantage of greater accessibility and convenience. This online tool streamlined the creation of standardized, interoperable, and machine-readable JSON recipe files, ultimately simplifying workflow integration and eliminating the necessity for local software installations.

An image of this GUI is provided in Figure 2.5, and the corresponding code is available on GitHub at the following link: <https://github.com/mat-tiamare9/json-writer.git> [22] or at the MDMC code repository, here [23].

Chapter 3

The vocabulary

Nanofabrication laboratories are at the forefront of creating silicon-based devices, sensors, and actuators, employing a wide range of growth and synthesis processes. These processes encompass a broad variety of fabrication steps. However, a critical challenge has emerged: there is currently no standardized vocabulary or terminology to consistently describe these procedures across different labs, not only within the CNR or NFFA-DI networks but also around the world.

Laboratories within the CNR network and participants in the NFFA-DI (Nanoscience Foundries and Fine Analysis – Digital Infrastructure) initiative use identical or similar technological processes, but often employ local or ad-hoc terms. This fragmentation of terminology leads to miscommunication and inefficiencies; data and process metadata recorded in one lab may be unclear or misinterpreted in another.

Analogous issues have been documented in other scientific domains, such as Batteries, Chemistry, Tribology, Biomaterials, where a “lack of consistency in the terminology” and the use of free-form natural language to describe experiments made it time-consuming to access and reuse knowledge, and rendered results from different groups hardly comparable [24]. These examples underscore the need for a common language in nanofabrication: a shared vocabulary that can eliminate ambiguities and improve clarity.

In response to this challenge, this chapter focuses on the development of a shared vocabulary for growth and synthesis processes in nanofabrication. The aim is to establish a controlled set of terms (an ontology or taxonomy) that researchers are invited to use to describe fabrication steps in a consistent, machine-readable manner. This effort is firmly grounded in the principles of FAIR data management: in particular, it addresses the *Interoperability* pillar [1]. In practice, this means replacing idiosyncratic naming conventions

with standardized terms defined in a common framework. By doing so, we enable nanofabrication process data to be understood not just by humans in different labs, but also by computers. Indeed, the FAIR principles put specific emphasis on enhancing machine actionability.

Establishing a common vocabulary is not merely a linguistic exercise, but a step toward a semantic framework for nanofabrication research. In the context of data science, shared ontologies and vocabularies have proven invaluable for aligning understanding across different teams and tools. Ontologies provide a precisely defined set of terms – effectively, a dictionary equipped with formal definitions and relationships – that captures a conceptualization of a domain. By formalizing the concepts of growth and synthesis steps (e.g., naming the main parameters for a *CVD deposition* or *thermal oxidation* in unambiguous terms), we build a foundation for common understanding. This approach yields multiple benefits for the community: first, it enables consistent annotation of experimental data and protocols across projects and laboratories.

When researchers describe their fabrication workflows using the same controlled vocabulary, the resulting metadata become semantically rich and comparable. Such data are searchable and exchangeable: one can query a database for all datasets involving a certain process and retrieve results across all labs because the process was labeled in a standard way. In essence, well-defined vocabulary terms turn free-form procedural descriptions into machine-interpretable metadata. This makes the data more reusable: scientists can more easily understand and build upon results from other labs, and software systems can integrate data without needing brittle, case-by-case mappings.

3.1 Pre-existent Literature

Rather than reinventing the wheel, we began by surveying existing work to draw inspiration and build on proven approaches. Below are some of the key sources we consulted.

3.1.1 MDMC-NEP Glossary of Terms

One existing vocabulary considered was a glossary of nanoscience terms published via Zenodo in February 2024 [25]: the *MDMC-NEP Glossary of Terms* is a high-level framework to describe experimental and computational workflows in nanoscience; it's a result of the Metadata Working Group, which involves members of the NFFA EUROPE Pilot (NEP) and of the Joint

Lab “Integrated Model and Data Driven Materials Characterization” (JL-MDMC) of the Helmholtz Association. The glossary contains a set of about 45 terms covering the lifecycle of a research project from material fabrication to publication, focusing on broad process stages and provenance information.

Upon evaluation, however, we found that this vocabulary operates at too high a level of abstraction for our purposes: it provides only generalized descriptions of processes and lacks the granularity needed to document individual fabrication steps in detail. Indeed, the authors themselves note that the current glossary is “idealized and simplified”, with plans to extend it to a more fine-grained level in the future. Due to this insufficient level of detail for step-specific process documentation, a more detailed terminology was needed for the project.

3.1.2 ISO/TS 80004-1

For a more granular and standardized terminology, we turned to the *ISO/TS 80004-1* (Nanotechnologies – Vocabulary – Part 1: Core terms) vocabulary. Developed by the International Organization for Standardization’s Technical Committee 229 (ISO/TC 229) on Nanotechnologies [26], this standard provides an authoritative list of core nanotechnology terms and their definitions, designed to facilitate consistent communication in the field.

Each term entry in ISO/TS 80004-1 is structured with a preferred term (the principal one), a clear definition of that term, any synonyms or alternative expressions, and notes that give additional context or clarification. As an example, the standard defines “*nanomaterial*” as a “material with any external dimension in the nanoscale or having internal structure or surface structure in the nanoscale”. The entry for *nanomaterial* also includes notes to elucidate the term’s scope, stating that this generic term encompasses both nano-objects and nanostructured materials, and references related terms.



This level of detail and formal structure makes the ISO vocabulary a robust reference for precisely describing materials and processes. Hence, it was found to be much more suitable for documenting our lab’s growth and synthesis processes. Many of the concepts involved in nanofabrication are directly covered or can be described unambiguously using the ISO standard’s terms and definitions. By adopting the ISO/TS 80004-1 terminology, we ensure that each fabrication step is described with a well-defined, consis-

tent, and unambiguous technical term, improving clarity and aligning the documentation with international standards.

This approach also supports the FAIR principles by using a common, interoperable vocabulary for process metadata. In summary, ISO/TS 80004-1 was used as the basis for our process documentation, and the specific fabrication steps described in the following sections of this thesis are documented using the standardized terms and definitions from this ISO vocabulary where available; in cases where no suitable ISO term existed, custom terms were defined, following the same descriptive structure and level of granularity.

3.1.3 NanoFabNet

During the development of the shared vocabulary, the resources provided by NanoFabNet were consulted too [27]. These included a community-curated taxonomy of processes and equipment, based on ISO/TS 80004-8:2020 and other sources [28]. The taxonomy offered five main categories with 21 sub-categories and around 55 generic terms, covering the most common nanofabrication methods. Several terms and structural elements from NanoFabNet were adopted to ensure alignment with community standards. However, the vocabulary was not fully integrated, as it lacked the level of detail needed to describe certain specific growth and synthesis steps. Instead, NanoFabNet’s framework was used as a foundation and complemented with terms from ISO/TS 80004-1, which provided standardized definitions. This combined approach ensured both domain relevance and consistency with international standards.

3.2 Hierarchical Taxonomy

Since a taxonomy provides an organized framework, each fabrication technology (also referred to here as a “step type”) is defined and structured at multiple levels of detail. In this taxonomy, each step type is hierarchically classified into:

1. Superclasses – The most general categories capturing broad functional or conceptual groupings of technological processes.
2. Classes – Macro ensemble that groups a set of specific fabrication processes.
3. Class Instances – The specific, detailed list of processes in a given class.

This hierarchy facilitates comparative analysis of fabrication steps and supports robust documentation for large-scale manufacturing or research undertakings. A brief overview of each of these levels is now provided.

3.2.1 Superclasses

At the highest level of the hierarchy, superclasses represent broad, functionally defined categories that group more specialized approaches based on their primary objective or physical mechanism. These superclasses serve as conceptual *umbrellas*, encompassing all techniques having the same final effect through broadly related methods. Designed to capture the principal action or physical mechanism distinguishing top-level processes, superclasses form the foundation for further subdivision. As such, they must be sufficiently distinct, comprehensive, and logically inclusive of any subclasses placed under them. The topmost level of this taxonomy, reflecting the most general grouping of fabrication actions, can take one of the following four categories:

Add superclass encompasses all the processes that act adding a layer of material onto a substrate, using chemical or physical actions. One example is chemical vapor deposition (CVD), where gases react to form a solid material that is deposited as a thin film, commonly used in microelectronics and materials fabrication.

Transform superclass encompasses processes used to define the surface structure of a material or a layer, or to modify its chemical, physical, or electrical properties, without adding or removing significant amounts of material. One example is thermal treatments like annealing, which are used to induce crystallization or control dopant diffusion, altering the material's properties. These processes focus on changing the composition or structure of the material to influence its behavior or interaction with subsequent steps.

Remove superclass includes processes designed to selectively remove material, shaping or patterning existing layers. One example is wet etching, where chemical etchants are used to dissolve material in localized regions, defining critical dimensions and shaping device layers. These processes are essential for creating precise patterns and ensuring that only the targeted areas of the material remain, ultimately influencing the surface properties and structure of the item.

Characterization superclass includes inspection and metrology procedures used to evaluate key parameters during or after fabrication. These

characterization methods enable feedback on process outcomes, ensuring that fabrication steps meet the required specifications and adjustments can be made as needed. However, characterization is not officially considered in our taxonomy, as our work focuses on nanofabrication technologies impacting device development rather than measurement or characterization.

Table 3.1: Overview of the eight classes, their respective superclasses, and any applicable ISO references.

Class	Superclass	ISO/TS 80004-1	
		Ref.	Definition
Lithography	Transform	8.1	Nanopatterning litho.
Etching	Remove	8.3	Etching processes
Synthesis	Add	8.2	Deposition processes
Integration	Add	–	–
Doping	Transform	7.5.8	Ion implantation only
Thermal Processing	Transform	–	–
Dicing	Transform	–	–
Characterization	Characterization	–	–

3.2.2 Classes

Classes form the second level of the taxonomy: while superclasses capture a broad action, classes identify the techniques used to implement that action. The table 3.1 clarifies these relationships, listing which class belongs to which superclass, along with relevant ISO references where applicable. Here a detailed description of each class is provided:

Lithography is the process of transferring a designed pattern onto a substrate at microscale or nanoscale dimensions. While many methods rely on a photo- or radiation-sensitive resist to define patterned regions for subsequent etching or deposition, direct-write techniques can imprint a pattern directly into the substrate without a separate resist layer.

Etching is the selective removal of material from a substrate to shape and define features. It can be performed using wet chemical solutions, which dissolve targeted regions, or dry plasma-based processes, which bombard the substrate with reactive ions or radicals. By precisely controlling

which areas are etched, often using masks, scientists create the intricate patterns and geometries essential for modern electronic and photonic devices.

Synthesis involves depositing or assembling material onto a substrate. Techniques can range from chemical processes—like chemical vapor deposition and solution-based growth—to physical methods such as evaporation or sputtering. By precisely controlling factors like precursor chemistry, temperature, and pressure, thin films or nanostructures are formed with specific thicknesses, compositions, or crystalline properties. This ability to add new layers or materials is critical for building complex device stacks and functional surfaces.

Integration focuses on joining or assembling materials or components into a cohesive structure. In micro- and nanofabrication, this can involve wafer bonding, flip-chip attachment, or other packaging techniques to ensure both mechanical integrity and reliable electrical connections.

Doping modifies the electrical properties of a semiconductor or other material by introducing impurity atoms or ions. Common methods include ion implantation or diffusion, where dopant species are driven into the substrate to achieve desired conductivity, carrier density, or junction profiles. This crucial step in device fabrication underlies the operation of most transistors, sensors, and other nanoelectronic components.

Thermal processing uses elevated temperatures, often in controlled atmospheres, to modify material structure or properties. Typical examples include annealing to relieve stress or activate dopants, oxidation to grow dielectric layers, or crystal growth to improve film quality. Through precise temperature profiles and gas flows, thermal processes enable critical changes in a substrate's composition and morphology.

Dicing is the process of cutting a substrate into individual dies or chips. This can be done mechanically, using a thin saw blade, or by laser-based methods. Once separated, these individual pieces can be packaged or integrated into larger systems, marking a critical transition from wafer-scale fabrication to device-level deployment.

Characterization involves examining a substrate's surface or internal structure. These insights help optimize fabrication processes and validate design outcomes by confirming whether devices meet intended specifications. One example is scanning electron microscopy (SEM), which provides high-resolution images of surface topography or cross-sections,

offering valuable insights into the material structure. This class is considered only if the related superclass is.

3.2.3 Technologies

Only the processes actually implemented in the CNR-ISMN cleanroom in Bologna are discussed here—out of the many steps defined in the full taxonomy—so as to keep this presentation focused and practically relevant (see Appendix B for a complete listing). The comprehensive taxonomy, encompassing all identified processes and tools, can be consulted online at [29].

3.3 Considerations

The taxonomy presented in this chapter stems from a collaborative effort between CNR-ISMN in Bologna and CNR-IFN (Institute for Photonics and Nanotechnologies) at the FBK (Fondazione Bruno Kessler) laboratories in Trento, marking a unique instance of two European cleanrooms joining forces to develop a unified nanofabrication vocabulary. Such a partnership underscores the growing need for shared reference frameworks within the scientific community—particularly for advanced, interdisciplinary research. It is hoped that other laboratories will adopt and refine this vocabulary, ultimately facilitating more coherent, reproducible, and scalable fabrication processes across Europe and beyond.

Additionally, applying this taxonomy in both facilities enabled the identification of overlapping fabrication steps and those unique to each cleanroom. By mapping process commonalities and differences, the groundwork has been laid for the detailed analysis in the following chapter, which explores the specific parameters that must be documented in a trusted repository. Such documentation is vital for ensuring consistent, reproducible results across different research environments.

Chapter 4

A FAIR Workflow

Materials science has entered a data-rich era in which high-throughput simulations and advanced experiments produce vast volumes of complex data. Effective data infrastructure is required to handle this information deluge, ensuring that datasets are well organized, preserved, and broadly accessible. The Novel Materials Discovery (NOMAD) repository is a leading solution addressing these challenges, and the main character of this section.

In this chapter, we present all the stages of the newly developed FAIR-compliant workflow. The discussion follows a reverse-order approach, beginning with an introduction to the final data destination: NOMAD. We then describe how the new NOMAD plugin, developed by the cleanrooms of Bologna and Trento, integrates with the platform and enables the deposition of data. Subsequently, we move backward along the workflow to analyze the modifications introduced in the ISMN’s *Laboratory Information Management System*: CAMS, which have been implemented to ensure that the generated files are compatible with the plugin and suitable for publication on NOMAD.

4.1 NOMAD

NOMAD is a free and Open Source platform developed by the FAIRmat consortium to enable scientists to manage, share, and publish materials science data on a global scale: indeed, the mission of the consortium is to maintain a federated FAIR data infrastructure for materials data while supporting the scientific community to introduce and maintain high standards of reproducibility, research integrity, and compliance with ethical and legal requirements. Since its public launch in 2014–2015, NOMAD has grown into the largest repository of its kind, encompassing more than 19 million simulations and more than 4 million unique materials.

4.1.1 Core Functionalities

NOMAD’s design goes beyond simple file storage; it provides an integrated environment to ingest data, enrich it with metadata, enable discovery, and facilitate analysis. Its core functionalities can be summarized as follows:

Data Ingestion and Publication Researchers can upload raw research data (e.g. simulation input/output files) in any of 60 (and more) supported formats. Upon upload, NOMAD automatically processes files to extract structured data and rich metadata, converting heterogeneous inputs into a common data schema (the *NOMAD Archive*). Users can organize data into projects or datasets incrementally and even integrate electronic lab notebooks (ELNs) for capturing experimental workflows. NOMAD supports publishing datasets with Digital Object Identifiers (DOIs), ensuring that data can be cited in publications and permanently referenced. Publishing is optional and under user control: data can remain private until the contributor decides to release it. Once published, the data are formally archived (preserved long-term) and assigned a DOI for persistent identification.

Unified Discovery and Access All content in NOMAD is made available through a unified interface and format, regardless of the original data source or simulation code. Users can search and explore the repository based on rich metadata rather than just raw filenames. Crucially, all published data in NOMAD are made openly accessible under a permissive license (CC BY 4.0), meaning anyone can download the full data or processed archives for reuse.

Programmatic Access and Analysis To further encourage data reuse, NOMAD offers extensive APIs and computational tools for accessing and analyzing data. All repository functions – from querying datasets to retrieving processed results – are exposed via a public REST API, enabling integration with external applications and workflows. The data served by NOMAD are machine-actionable, provided in a unified JSON-based format (the *Archive schema*) that is independent of any specific software code. Moreover, users can launch notebook servers on this infrastructure to perform interactive analysis on the data without needing to download large files, and then publish the notebooks or their results.

Through these capabilities, NOMAD creates an ecosystem in which managing data (uploading, curating metadata, obtaining DOIs), discovering data (searching and visualizing), and utilizing data (downloading or computing on

it) are all streamlined. The platform's free and open availability lowers the barrier for individual researchers or large collaborations to adopt good data management practices. As a cloud-based service hosted at a high-performance data center, NOMAD also spares users from needing to maintain their own storage or servers, while still giving options (see the next section) for groups that require local control.

NOMAD Oasis

NOMAD Oasis is an on-premises deployment of the NOMAD platform, enabling research laboratories to manage and preserve their data within local infrastructure. It replicates the core functionalities of the central NOMAD service in a private environment, offering the same web interface, tools, and workflows. However, unlike the public NOMAD repository, OASIS does not currently mint DOIs and is not a certified repository; rather, it serves as a local digital platform for data sharing and discovery.

Another practical aspect is its ease of deployment and maintenance: the platform is distributed via containerization technologies (such as Docker), which enables straightforward installation on local servers. In typical scenarios, an entire Oasis instance can be set up within minutes using pre-configured Docker images and compose files, making it feasible for research groups to deploy a professional data management system with minimal IT overhead. Remarkably, this container-based approach also ensures that the software environment is consistent with that of the central NOMAD, reducing configuration complexities and simplifying updates or scaling when needed.

The key purpose of NOMAD Oasis is to meet local data management needs by allowing groups to organize, annotate, and share research data internally, using their own compute and storage resources rather than relying on a remote server. Furthermore, to facilitate development and customization, NOMAD offers the *nomad-distro-dev* environment [30], which allows developers to experiment with new features and plugins in a streamlined setup. This development environment supports editable installations, enabling immediate reflection of code changes without the need for reinstallation. It also provides centralized codebase management, improved editor support, consistent tooling, and flexible plugin management, making it easier for developers to contribute to and extend the NOMAD platform.

The *nomad-distro-dev* environment was instrumental in our project, serving as the foundational platform for developing and testing the plugin essential for our FAIR workflow. This development setup allowed us to integrate custom functionalities into NOMAD efficiently, ensuring that our extensions aligned

seamlessly with the existing infrastructure. We could rapidly prototype and validate our plugin, facilitating a more agile and effective development process.

By supporting both production and development environments, NOMAD Oasis ensures that research groups can maintain FAIR data management practices locally, while also contributing to the broader ecosystem through customized development and experimentation.

4.1.2 Role in NFFA-DI

NOMAD’s capabilities have been recognized and adopted by broader research infrastructure initiatives. NFFA-DI aims to integrate data from nanofabrication and characterization foundries and make them FAIR across dozens of labs: notably, participants in the NFFA-DI project have selected NOMAD as the repository of choice for hosting lab process data from its network of experimental facilities. In this context, NOMAD provides the backend for storing and organizing the experimental data generated in NFFA-DI.

While this repository was originally designed to manage data from characterization experiments, its architecture has been extended to accommodate nanofabrication workflows. This adaptation is particularly relevant for the NFFA-DI project, which requires the integration of synthesis-related data. To facilitate this, NOMAD’s flexible data model and plugin system have been leveraged to support the unique requirements of nanofabrication processes.

A significant challenge in this integration was the standardization of data formats, since NOMAD supports NeXus files, an open and legally recognized standard for experimental data. However, extending NeXus to encompass the specific outputs of nanofabrication facilities would have been time-consuming. To address this, we developed a custom plugin capable of parsing data directly from our cleanroom instruments.

These enhancements enable NOMAD to serve as a comprehensive repository for both characterization and synthesis data, aligning with the FAIR principles of data management. The system now supports a broader range of experimental workflows.

4.2 The Plugin

The development of the plugin marks the commencement of our workflow journey. Utilizing the NOMAD Oasis environment, specifically the *nomad-distro-dev* setup, the subsequent step involves creating a dedicated plugin

which ensures that all data generated within a cleanroom is accurately parsed and integrated into NOMAD’s infrastructure.

A noteworthy aspect of this endeavor is the collaborative effort between CNR-ISMN in Bologna and CNR-IFN in Trento in developing both the taxonomy and the plugin. The primary objective was to design a plugin with a highly generalizable structure. Adopting a modular approach allows the plugin to accommodate various fabrication processes, while the taxonomy provides a systematic organization of these processes.

Once the taxonomy reached a preliminary stage—acknowledging that it’s an ongoing project—the teams from both laboratories convened to identify the specific parameters collected for each fabrication process (referred to as *step types*) in their respective labs. This comparative analysis aimed to determine whether each lab could achieve complete reproducibility of the processes.

Considering two critical factors: (1) this initiative represents the initial approach towards establishing a FAIR workflow, and (2) the laboratories operate under different paradigms (for instance, one lab might focus on fabrication from the perspective of a single wafer, while another might consider the entire workflow encompassing multiple items), we concluded that the most inclusive strategy would be to implement the *union* of parameters collected by both labs into the plugin. This approach ensures comprehensive coverage and facilitates interoperability across different fabrication methodologies.

4.2.1 The Structure

While the *Fabrication Utilities* plugin repository [31] contains various standard files—such as `pyproject.toml`, `requirements.txt`, and `LICENSE`—the core functionality resides in the `src/fabrication_facilities` directory.

This directory houses three pivotal components: *apps*, *normalizers*, and *schema packages*.

Schemas: Defining Data Structure Schemas are fundamental to the plugin’s architecture. They define the structure and organization of data within NOMAD by specifying sections and quantities. Schemas can be defined in Python or YAML and are organized into schema packages, facilitating modular and reusable data definitions. They dictate how data ingested by the plugin is structured and presented to the user. By defining clear data models, schemas enable reliable data validation, storage, and retrieval [32].

Apps: Enhancing Data Findability Apps provide customized user interfaces within the NOMAD GUI, tailored to specific domains or data types. They enhance data findability by offering intuitive menus and dashboards that highlight relevant properties and facilitate efficient data exploration.

Normalizers: Automating Data Processing Normalizers are methods (of Python classes) that process and augment the data after parsing. They take the parsed data (as instances of the `EntryArchive` class) as input and can add additional sections or quantities based on existing information. Normalizers are executed in a defined order and can be customized to perform code-independent processing steps, ensuring that the data conform to the desired structure and semantics.

In our implementation, we did not rely on normalizers or parsers, as we directly provide NOMAD with a well-structured JSON file that is already compliant with the expected schema (more about this feature in the next sections, see also appendix C).

Regarding the Apps, we ensured that every *Quantity* (i.e., parameter) associated with each process step is indexed. This enables users to search for any specific value that a step might take, thereby maximizing findability. While no code examples are included here—since the implementation is relatively straightforward—the key task was to define, for each quantity, the unit of measurement (when applicable) to be displayed in the app’s Menu (see [31] for the full code and Figure 4.1). An important detail is that Menus follow a hierarchical structure, mirroring the taxonomy defined in the schema. This design ensures intuitive navigation and semantic clarity when exploring data through the NOMAD interface.

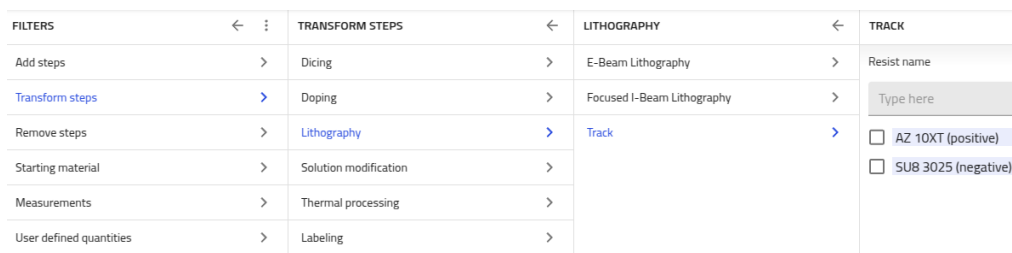


Figure 4.1: A glance at the App interface. Menus follow a hierarchical structure, from superclasses (see 3.2.1) to step types, and finally (on the rightmost side) to the searchable parameters of each step. Only two steps feature a separate menu: one for the starting material undergoing the process, and one for the characterization and measurements performed on the sample.

4.2.2 Schemas

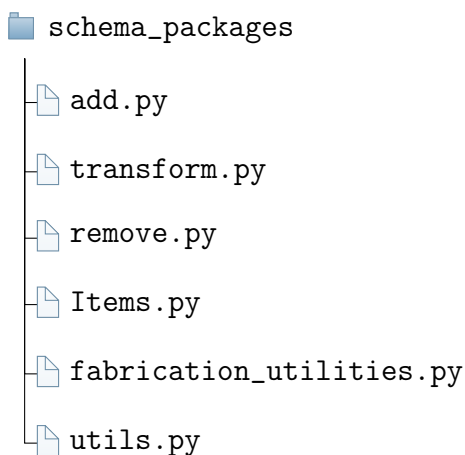


Figure 4.2: Directory tree illustrating the organization of the schemas in the NOMAD plugin.

Schemas as well follow the taxonomy, since three files reflect the official superclasses (add, transform, remove) and contain all the step types belonging to them, as illustrated in the adjacent figure (4.2).

The `Items.py` file defines data models for items involved in nanofabrication processes. It includes classes that specify attributes such as material type, dimensions, and initial conditions of substrates or wafers. These classes standardize how items are represented and tracked throughout the fabrication workflow.

`fabrication_utilities.py`, which allows the initial material to be treated as a fabrication step. The same file also contains several utility classes such as `FabricationProcessStep` and `FabricationProcess`, which define the core structure of a complete process and its individual steps, including shared parameters applicable across all steps. Each class corresponding to a specific step type inherits from `FabricationProcessStep` to specialize it by introducing step-specific parameters.

A similar role is likewise played by `StartingMaterial`: a straightforward but fundamental class in the file

The file also includes the association of equipment with specific techniques. Equipment is described using the homonymous class, which not only stores general metadata (e.g. model, manufacturer) but also includes technical capabilities (`EquipmentParameterData`), compatible item types, and techniques supported (`EquipmentTechnique`). Finally, `SampleParenting` allows modeling how a given item results from multiple steps and materials, effectively mapping fabrication lineages.

`utils.py` defines two main components reused across the package: a utility function to parse chemical formulas, and the `Massflow_controller` class, which represents a gas source in fabrication processes. This class stores the flow rate and automatically computes the elemental and mass composition from the chemical formula.

Figure 4.3 and 4.4 show an example of a class from `transform.py`. All

```

601 class Stripping(Cheical, FabricationProcessStep, ArchiveSection):
602     m_def = Section(
603         a_eln={
604             'hide': [
605                 'description',
606                 'lab_id',
607                 'datetime',
608                 'comment',
609                 'duration',
610                 'end_time',
611                 'start_time',
612             ],
613             'properties': {
614                 'order': [
615                     'job_number',
616                     'name',
617                     'description',
618                     'affiliation',
619                     'location',
620                     'operator',
621                     'room',
622                     'id_item_processed',
623                     'starting_date',
624                     'ending_date',
625                     'step_type',
626                     'definition_of_process_step',
627                     'recipe_name',
628                     'recipe_file',
629                     'stripping_type',
630                     'short_name',
631                     'chemical_formula',
632                     'duration_target',
633                     'removing_temperature',
634                     'ultrasound_required',
635                     'notes',
636                 ]
637             },
638         },
639     )

```

(a) Header of the class

```

601 class Stripping(Cheical, FabricationProcessStep, ArchiveSection):
602     stripping_type = Quantity(
603         type=str,
604         a_eln={
605             'component': 'StringEditQuantity',
606         },
607     )
608     short_name = Quantity(
609         type=str,
610         description='Material to remove',
611         a_eln={'component': 'StringEditQuantity', 'label': 'target material'},
612     )
613     chemical_formula = Quantity(
614         type=str,
615         description='Inserted only if known',
616         a_eln={'component': 'StringEditQuantity'},
617     )
618     removing_temperature = Quantity(
619         type=np.float64,
620         a_eln={
621             'component': 'NumberEditQuantity',
622             'defaultDisplayUnit': 'celsius',
623         },
624         unit='celsius',
625     )

```

(b) Quantities

```

629 def normalize(self, archive: 'EntryArchive', logger: 'BoundLogger') -> None:
630     super().normalize(archive, logger)
631     if self.chemical_formula:
632         elements, counts = parse_chemical_formula(self.chemical_formula)
633         total = 0
634         for token in counts:
635             total += int(token)
636         if total != 0:
637             elemental_fraction = np.array(counts) / total
638             elementality = []
639             i = 0
640             for entry in elements:
641                 elemental_try = ElementalComposition()
642                 elemental_try.element = entry
643                 elemental_try.atomic_fraction = elemental_fraction[i]
644                 i += 1
645                 elementality.append(elemental_try)
646             else:
647                 print('No elements provided')
648             self.material_elemental_composition = elementality

```

(c) Normalize function

Figure 4.3: The left column displays the class header, including its inheritance hierarchy and both inherited and class-specific attributes. The top-right section presents examples of quantities and how they are constructed, while the bottom-right illustrates the `normalize` function, which identifies and parses a chemical formula to distinguish its constituent elements.

Figure 4.4: The interface of a step in NOMAD when used as an electronic laboratory notebook.

classes follow a similar structure: they inherit from relevant superclasses, define the display order of parameters in the GUI (using the *order* property), and may hide certain fields using the *hide* keyword. They then specify a set of *Quantities*, which are the actual parameters whose values will be stored in the database. Each quantity requires a type, may include a description and, when required, specifies a unit of measurement. Additionally, quantities can be displayed in the GUI with a custom label. The figure also shows an example of the `normalize` function, which processes the chemical formula—if provided—to extract and register the elements involved in the step, using the `parse_chemical_formula` function described above.

A noteworthy and convenient feature of NOMAD is its flexibility regarding input data completeness. When submitting data, it is not strictly necessary to provide values for every defined quantity. In the associated JSON file, if a key corresponding to a particular parameter is absent, NOMAD will either ignore that field altogether or substitute a default value—provided that a default has been specified in the quantity’s definition.

Now that we understand how NOMAD reads and stores the data we send to it, we can move on to explain how the raw data are prepared prior to submission.

4.3 CAMS Upgrade Overview

As discussed in Section 2.2.4, an initial effort was made to generate a machine-readable file, enabling CAMS to export all data—both from the main process and from each individual step—in a single JSON file. However, this approach proved insufficient. Uploading a single, consolidated file undermined the goal of creating a user-friendly interface, as Apps were unable to isolate the parameters for individual steps when they were all mixed together.

The solution was straightforward and easy to implement: for each *Run* and its associated *steps*, a separate file is created for each step, containing all relevant information. A main or *parent* file, corresponding to the Run, stores general information and includes a dedicated list that references all the step files, preserving their order.

To achieve this, the logic for file generation and uploading first handles the upload of each step file, retrieves the corresponding *upload_id* and *entry_id* for each (see Figure 4.4 on the left side), and then populates the reference list in the parent file accordingly, as illustrated in Appendix C.

4.3.1 APIs

The NOMAD platform provides a well-structured and comprehensive RESTful API, enabling seamless integration with Python-based environments. This is particularly advantageous for applications like CAMS that utilize Python backends. The API facilitates the entire data management workflow, from authentication and file uploads (including compressed formats like `.zip`) to metadata editing, dataset creation, and final publication.

NOMAD offers example Python functions to streamline these processes, allowing for efficient automation and integration into existing Python workflows [33]. After some adjustments primarily focused on exception handling, these functions were consolidated into a single file (available at the link in these references [23, 34]), which now serves as a plugin for CAMS. This file is simply imported into the backend, so once the data are extracted, a straightforward function call referencing this plugin handles the upload and publication process. This approach keeps the backend code clean and well-organized.

Here's a concise explanation of the functions and their roles:

`get_authentication_token` Authenticates with the NOMAD API using a username and password, returning an access token required for all other operations. Includes error handling for HTTP and connection issues.

`upload_to_NOMAD` Uploads a file to NOMAD. Accepts compressed archives and automatically triggers NOMAD's internal decompression and processing.

`check_upload_status` Retrieves the current status of a specific upload using its `upload_id`. Useful for monitoring whether the upload has completed or encountered errors.

`wait_for_upload_completion` Continuously polls the upload status at fixed intervals until the upload is successfully processed or a timeout is reached. Helps automate the workflow without manual checks.

`get_upload_entries` After a successful upload, this function fetches the list of processed entries, including useful metadata such as filenames and entry IDs. Returns a dictionary of results.

`delete_upload` Deletes an upload from the NOMAD repository, typically used when the upload fails or is no longer needed. Handles all request-related exceptions gracefully.

`upload_file` A high-level function that coordinates the full upload pipeline: authenticates the user, uploads the file, waits for completion, retrieves entry data, and optionally deletes failed uploads. This function acts as the main callable interface from the CAMS backend.

`publish_upload` Publishes an upload to make it publicly accessible via the NOMAD repository.

4.3.2 Data Transformation

One key distinction between CAMS and NOMAD data is their vocabulary. As detailed in Sections 3.3 and 4.2, once an initial taxonomy had been drafted, we reached an agreement on which parameters from each processing step needed to be exported. After defining the classes for each step type in the plugin [31], we finalized those parameters endeavoring, where possible, to reuse consistent names across different classes.

Since the plugin was designed to support generalization and interoperability, many of the identifiers used in CAMS do not align with the plugin's naming conventions. Consequently, once the data are exported into JSON dictionaries (sometimes spread across multiple files) the dictionary keys may not match NOMAD's schema. The final challenge, therefore, was to implement a reliable mapping mechanism to translate these keys.

Mapping the run-specific parameters and those common to all steps proved straightforward: we simply constructed a dictionary using the appropriate NOMAD-compliant keys and assigned each its corresponding value, as shown in the frame 4.1.

Source Code 4.1: Function that prepare the dictionary with the correct NOMAD keys associated to some parameters of a *Run*.

```
1 def get_process_dict(run):
2     """
3     Given an open run, the function returns a standardized dict that
4     ↪ gathers all process params.
5     The trailing 'steps' list will be filled in another function
6     """
7     return {
8         "m_def": "fabrication_facilities.schema_packages.fabricatio_
9         ↪ n_utilities.FabricationProcess",
```

```
8     "name": run.run_name.display_text,  
9     "id_proposal": run.pid.display_text,  
10    "project": run.project.display_text,  
11    "description": run.run_description.display_text,  
12    "author": run.author.display_text,  
13    "cost_model": run.cost_model.display_text,  
14    "affiliation": run.pnrr_project.display_text,  
15    "start_date": serialize_value(run.start_date.value),  
16    # "generated_on": serialize_value(datetime.now()),  
17    "steps": [],  
18 }
```

The step-specific parameters required additional care because there are over twenty distinct step types, and hardcoding every possible parameter combination would clutter the code. To maintain clarity, we separated the conversion logic from the data definitions. We created a configuration file named `cams2nomad.py` that houses a dictionary entry for each CAMS step type [23], each containing three fields:

1. `m_def` – This unambiguously links a CAMS step type to its corresponding NOMAD class. Several CAMS step types can map to the same NOMAD class thanks to their shared generalization.
2. `mapping` – A sub-dictionary pairing NOMAD field names with their CAMS counterparts. During conversion, for each pair, the code retrieves the value under the CAMS key and assigns it to the appropriate NOMAD key.
3. `transformations` – Another sub-dictionary that assigns a transformation function to specific NOMAD keys. When the raw CAMS value requires unit conversion or type casting to meet NOMAD standards, the function is applied before assignment.

By clearly separating mapping definitions from transformation logic, this setup keeps the conversion code both concise and flexible as new step types are introduced. The code environment 4.2 demonstrates how the above bullet-point configuration is applied in practice.

This concludes the trip along the workflow of uploading and publishing data in a FAIR repository, here NOMAD. In summary, when the *Upload*

FAIR button is triggered in CAMS, the associated function collects all data related to the process and its individual steps. If any of this data needs to be exported, it is first converted, when necessary, and then mapped to the appropriate keys. Once this is done, the data is uploaded to the local OASIS platform. However, the workflow remains fully compatible with the central NOMAD repository, simply by setting the correct endpoint and authentication parameters.

Source Code 4.2: Example entry from the CAMS-to-NOMAD conversion file. Note that the values in `transformations` represent functions (e.g. `min2sec`, which converts minutes in seconds) and should not be enclosed in double quotes—they're quoted here purely for typographical rendering aesthetics.

```
1 {
2   "Stripping" :{
3     "m_def" : "fabrication_facilities.schema_packages.remove.St_
4     ↪ ripping",
5     "mapping" : {
6       "stripping_type":"stripping_type",
7       "short_name":"material_to_remove",
8       "duration_target":"duration",
9       "removing_temperature":"temperature",
10      "ultrasound_required":"ultrasaund",
11      "recipe_name":"",
12    },
13    "transformations" : {
14      "ultrasound_required": "choice",
15      "duration_target": "min2sec",
16    }
17  },
18 }
```

Final Remarks and Prospects

In this thesis, we have laid the groundwork for a truly FAIR-compliant laboratory data ecosystem: one that not only meets today’s needs but can evolve in step with future research demands.

First, although it wasn’t detailed in the previous chapters, a comprehensive Data Management Plan was in fact conceived at the outset of this project and has been actively maintained and iteratively updated throughout the internship. It codifies how all data—raw, intermediate, and processed—are collected, documented, stored, and shared. As an “alive” document, it will continue to evolve, reflecting new processes, tools, and community requirements.

Second, our investigation into a standardized hierarchical taxonomy for nanofabrication processes represents only the beginning of what must be an ongoing effort. At its current preliminary stage, the taxonomy provides a consistent framework for naming and describing each process step, but it will need to be enriched, corrected, and aligned with the latest community standards as further research and inter-laboratory collaborations unfold.

Third, the Fabrication Utilities plugin for the NOMAD repository exemplifies the “living” nature of our tools. Designed with maximum generality in mind, the plugin can accommodate future modifications to the taxonomy, integrate new process parameters, and adapt to evolving dependencies. However, it will require regular maintenance: both to incorporate advances in the taxonomy and to add new features, ensuring continued compatibility and performance.

Finally, the embryonic FAIR workflow presented here, spanning CAMS exports, JSON report generation, and seamless deposition into NOMAD, demonstrates the feasibility of implementing FAIR principles end-to-end. Yet this workflow is by no means complete. It, too, must be iterated, perfected, and possibly extended with additional workflows (e.g., direct integration of electronic lab notebooks or automated metadata harvesting) to make our data as FAIR and as resilient as possible.

Taken together, these four strands—from living DMPs to evolving taxonomies, from adaptable plugins to extendable workflows—outline a dynamic roadmap for embedding FAIRness at the heart of nanofabrication research. By treating every component as a “living” entity that grows and adapts alongside our science, we position ourselves to meet the ever-changing challenges of tomorrow’s data-driven discoveries.

Appendix A

JSON Report

This appendix provides examples of the report export functionalities implemented in CAMS, which are essential for achieving FAIR-compliant data management.

The first set of images (see [A.1](#) and [A.2](#)) illustrates the original report format, which was generated as a Microsoft proprietary Word (.docx) document. Although human-readable and convenient for manual inspection, this format is not well-suited for automated processing or integration with external systems due to its unstructured nature and reliance on proprietary standards.

To enhance machine-readability and interoperability, a new export functionality was developed to output the same report data in JSON format. The second part of this appendix contains a representative JSON excerpt demonstrating the structure and richness of the exported data.

Although the Word and JSON examples refer to different experimental runs, this does not affect the comparison, as both formats share a set of common parameters and structure relevant to the CAMS data model. These examples serve to highlight the advantages of adopting structured, open formats for research data reporting and reuse.

Source Code A.1: The new JSON report.

```
1 {  
2   "run_name": "CUPIDpreliminary",  
3   "project": "INFN - CUPID",  
4   "workpackage": "",  
5   "run_description": "",
```

Run dump: Testrun TCD-like

Project: Site development tests

Workpackage:

Run description:

Author: Mattia Marella

Cost model: External

Start date: 03/23/2020

This dump was produced on 2025-01-09 10:28:16

Figure A.1: Header section of the Word-based report export from CAMS, showing metadata such as project name, author, and start date.

Step number: 1 Step type: Thermal oxidation

Description: Thermal oxidation 200nm

	Step Parameters	Value
	Description	Thermal oxidation 200nm
	Operator	Mattia Marella
	Wafers	W1, W2, W3, W4, W5
	Planned date	2020-03-23
	Done	True
	Run	

Thermal oxidation parameter	Value
Oxidation Type	Clean Dry Oxidation (F1)
Recipe name	
Process number	0
Process Tempere [°C]	0
Expected Thickness [nm]	100
Duration Process [min]	0
Obtained Thickness [nm]	0
Note	

Comments	Value
Important notes	Proteggere retro dei wafer
Comments before process	Attenzione a...
Comments after process	

Room information	Value
Room	Furnaces
Room comments	

Timing information	Value
Operator start time	2020-03-23 10:05:00
Operator end time	2020-03-23 12:45:00
Equipment start time	2020-03-23 10:10:00
Equipment end time	2020-03-23 12:05:00

Figure A.2: Detailed view of a single process step in the Word report, including general parameters, timing information, and step-specific data for thermal oxidation.

```
6     "author": "Mattia Marella",
7     "cost_model": "",
8     "start_date": "2020-10-12",
9     "generated_on": "2025-01-12 22:55:30",
10    "steps": {
11      "0": {
12        "Header": {
13          "Step#": 1,
14          "Step type": "Electron Gun",
15          "Description": "Test deposition rate evaporazione Pd
16          ↪ "
17        },
18        "Step General Parameters": {
19          "Description": "Test deposition rate evaporazione Pd
20          ↪ ",
21          "Operator": "Mattia Marella",
22          "Wafers": "T1",
23          "Planned date": "2020-10-12",
24          "Done": false,
25          "Run": "CUPIDpreliminary"
26        },
27        "Comments": {
28          "Important notes": "evaporazione su campione test ",
29          "Comments before process": "Misura deposition rate
30          ↪ Pd\nsu wafer di Si di test",
31          "Comments after process": "definito il dep rate "
32        },
33        "Room information": {
34          "Room": "Metallizations",
35          "Room comments": ""
36        },
37        "Timing information": {
38          "Operator start time": "2020-10-12 09:20:15",
39          "Operator end time": null,
40          "Equipment start time": null,
41          "Equipment end time": null
```

```
39         },
40         "step_specific_info": {
41             "Deposited Material": "Palladium (Pd)",
42             "Process Number": 1,
43             "Required Thickness [nm]": 100,
44             "Expected Duration [s]": 0,
45             "Chamber Vacuum [mTorr]": "",
46             "Obtained Thickness [nm]": 0,
47             "Note": ""
48         }
49     },
50     "1": {
51         // ...
52     }
53 }
54 }
```

Appendix B

Technologies

Table B.1: List of the available technologies at CNR-ISMN.

Superclass	Class	Technology
Add	Synthesis	Chemical Vapour Deposition Dip coating Electro-deposition & -plating Evaporation Physical Vapour Deposition Sputter deposition Spin coating Spray deposition Surface polymerization
	Bonding	Direct bonding Anodic bonding Thermalcompression bonding
Remove	Etching	DRIE Bosch etching DRIE Cryogenic etching Dry ashing FIB assisted etching/milling ICP RIE RIE Wet etching

Continued on the next page

Continued from the previous page

Superclass	Class	Technology
Transform	Lithography	Electron beam lithography
		Photo-/Opto-lithography
		Nano-imprint lithography
		Focused Ionbeam lithography
		Micro-contact printing
Transform	Dicing	Soft lithography
		Diamond blade dicing
		Silicon dry oxidation
		Silicon wet oxidation
		Sintering
Transform	Thermal processing	Annealing
		Drive in
		Baking
		Ion implantation
		Diffusion from solid source
Transform	Solution modification	Resist development
Characterization	Characterization	Optical microscopy

Concluded here, from the previous page

Appendix C

NOMAD-ready files

This appendix presents example code for files prepared for submission to NOMAD. Since each step must be uploaded individually, we demonstrate the structure of these files and how the Run file references each of the step files.

Source Code C.1: Example of a file associated to a *Run*.

```
1 {
2   "data": {
3     "m_def": "fabrication_facilities.schema_packages.fabricatio_
4     ↪ n_steps.FabricationProcess",
5     "name": "ISOPILLAR_RING",
6     "id_proposal": "ABC",
7     "project": "CHALLENGE",
8     "workpackage": "XYZ",
9     "description": "My description",
10    "author": "Mattia Marella",
11    "cost_model": "Partner",
12    "affiliation": "NFFA-DI",
13    "start_date": null,
14    "steps": [
15      "1.archive.json",
16      "2.archive.json",
17      "3.archive.json",
18      "4.archive.json",
19      "5.archive.json",
```

```
19         "6.archive.json",
20         "7.archive.json",
21         "8.archive.json"
22     ]
23 }
24 }
```

Source Code C.2: Example of a file associated to a single step.

```
1 {
2   "data": {
3     "name": "LABELING",
4     "operator": "Mattia Marella",
5     "id_items_processed": "HR1, HR2, HR3, HR4, HR5, HR6",
6     "room": "Lithography",
7     "step_type": "Starting Materials",
8     "notes": "<ul> <li>Step notes</li> </ul> <p></p> <ul>
   ↳ <li>Important notes</li> </ul> <p></p> <ul> <li>Comments
   ↳ before process</li> </ul> <p></p> <ul> <li>Comments
   ↳ after process</li> </ul> <p></p> <ul> <li>Room
   ↳ comments</li> </ul> <p></p> ",
9     "m_def": "fabrication_facilities.schema_packages.fabricatio_
   ↳ n_utilities.StartingMaterial",
10    "wafer_quantity": 6,
11    "wafer_resistivity": -1,
12    "wafer_orientation": "<111>",
13    "wafer_thickness": 500,
14    "wafer_surface_finish": "DSP",
15    "short_name": "n (Ph) ",
16    "manufacturer_name": "SIEGERT"
17  }
18 }
```

Bibliography

- [1] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, *et al.*, “The fair guiding principles for scientific data management and stewardship,” *Scientific Data* **3** (2016) 160018.
- [2] W. Contributors, “Fair data,” 2023. https://en.wikipedia.org/wiki/FAIR_data. Accessed: 2025-03-22.
- [3] Dutch Techcentre for Life Sciences, “G20 endorse the fair principles,” 2016. <https://www.dtls.nl/2016/09/13/g20-endorse-fair-principles/>. Accessed: 2025-03-22.
- [4] M. Barker and et al., “Introducing the fair principles for research software,” *Nature* **3** (2019) 1314.
- [5] F.-I. Project, “Use cases,” 2023. <https://fair-impact.eu/use-cases>. Accessed: 2025-03-22.
- [6] N. C. for Biotechnology Information (NCBI), “Fair data sharing workshops,” 2023. <https://www.ncbi.nlm.nih.gov/books/NBK603623/>. Accessed: 2025-03-22.
- [7] CODATA, “Implementing fair principles: Challenges and considerations,” *Data Science Journal* **19** (2020) 32.
- [8] L. Vogt, “The fairer guiding principles: Organizing data and metadata into semantically meaningful types of fair digital objects to increase their human explorability and cognitive interoperability,” 2024. <https://arxiv.org/abs/2301.04202>.
- [9] S. Solutions, “Applying fair principles to lab data,” 2023. <https://bioit.semaphoresolutions.com/applying-fair-principles-to-lab-data/>. Accessed: 2025-03-27.

- [10] Benchling, “Developing a fair data strategy: Key considerations,” 2023. <https://www.benchling.com/blog/fair-data-strategy>. Accessed: 2025-03-27.
- [11] F. P. Framework, “5.0 developing a fair data strategy,” 2025. <https://www.fairprocessframework.org/steps/step-5/>. Accessed: 2025-03-27.
- [12] OpenAIRE, “How to make your data fair,” 2025. <https://www.openaire.eu/how-to-make-your-data-fair>. Accessed: 2025-03-27.
- [13] A. Unknown, “Fair data: What it is and how we can support its principles.” <https://www.csescienceeditor.org/article/fair-data-what-it-is/>, 2025. Accessed: 2025-03-27.
- [14] N. I. of Health, “Getting practical with the fair principles,” 2025. <https://datascience.nih.gov/getting-practical-with-the-FAIR-principles>. Accessed: 2025-03-27.
- [15] S. Hill, “Ai-ready fair data: Accelerating science through responsible ai and data stewardship.” https://medium.com/@sean_hill/ai-ready-fair-data-accelerating-science-through-responsible-ai-and-data-stewardship-3b4f21c804fd, July, 2024. Medium.
- [16] H. Marmanis, “Unlocking the power of fair data: Building trust and success in the ai era.” <https://www.copyright.com/blog/unlocking-the-power-of-fair-data-building-trust-and-success-in-the-ai-era/>, October, 2023.
- [17] N. Ravi, P. Chaturvedi, E. A. Huerta, Z. Liu, R. Chard, A. Scourtas, K. J. Schmidt, K. Chard, B. Blaiszik, and I. Foster, “Fair principles for ai models with a practical application for accelerated high energy diffraction microscopy,” November, 2022. <https://www.nature.com/articles/s41597-022-01712-9>.
- [18] Istituto per lo Studio dei Materiali Nanostrutturati (ISMN), “Institute for the science and technology of materials - cnr,” 2025. <https://www.ismn.cnr.it/>. Accessed: 2025-03-30.

- [19] Nano Foundries and Fine Analysis Digital Infrastructure (NFFA-DI), “Nffa-di: The advanced infrastructure for nanosciences,” 2025. <https://nffa-di.it/en/about-us/project/>. Accessed: 2025-03-30.
- [20] K. Powe, “Jam.py testing.” <https://testrigor.com/jam-py-testing/>, 2023. Accessed: 2024-03-30.
- [21] A. Yushev, “Jam.py github repository.” <https://github.com/jam-py/jam-py>, 2023. Accessed: 2024-03-30.
- [22] M. Marella, “Json writer,” 2024. <https://github.com/mattiamare9/json-writer>. Accessed: 2025-04-04.
- [23] M. D. Management and Curation, “Thesis code.” https://github.com/Master-Data-Management-and-Curation/Thesis_Code, 2025. Accessed: 2025-04-27.
- [24] E. Norouzi, J. Waitelonis, and H. Sack, “The landscape of ontologies in materials science and engineering: A survey and evaluation,” *CEUR Workshop Proceedings* **3760** (2024) 78–100. <https://publikationen.bibliothek.kit.edu/1000175445>.
- [25] N. E. Pilot and M. J. Lab, “Mdmc-nep glossary of terms.” <https://zenodo.org/record/10663833>, 2024.
- [26] I. O. for Standardization, “Nanotechnologies — vocabulary — part 1: Core terms,” 2015. <https://www.iso.org/standard/68058.html>.
- [27] NanoFabNet, “Nanofabrication competence map: Infrastructures, knowledge & skills – proposal for a new nanofabrication taxonomy,” 2021. <https://nanocommons.github.io/user-handbook/NanoFab/>.
- [28] “Nanotechnologies — vocabulary — part 8: Nanomanufacturing processes,” 2020. <https://www.iso.org/standard/74666.html>.
- [29] L. Ferrario, P. Maccagnani, I. Elmi, C. Gionco, G. Aprile, M. Bontorno, M. Marella, and P. Conci, “Nada taxonomy: a common vocabulary for fair nanofabrication data management,” tech. rep., Zenodo, Apr., 2025. <https://doi.org/10.5281/zenodo.15262085>. Technical note, Version v1.

- [30] FAIRmat-NFDI, “Nomad development distribution.”
<https://github.com/FAIRmat-NFDI/nomad-distro-dev>, 2025.
Accessed: 2025-04-12.
- [31] M. Bontorno and M. Marella, “Fabrication utilities.”
<https://github.com/Trog-404/Fabrication-utilities>, 2025.
Accessed: 2025-04-14.
- [32] N. Laboratory, “Get started with plugins.” <https://nomad-lab.eu/prod/v1/docs/howto/plugins/plugins.html>, 2025.
Accessed: 2025-04-14.
- [33] N. Laboratory, “How to publish data programmatically with python.”
https://nomad-lab.eu/prod/v1/docs/howto/programmatic/publish_python.html, 2025. Accessed: 2025-04-18.
- [34] M. Marella, “Nomad workflow.”
<https://github.com/mattiamare9/nomad-workflow>, 2025. Accessed:
2025-04-18.