ing.grid preprints

RESEARCH ARTICLE

How to Make Bespoke Experiments FAIR: Modular Dynamic Semantic Digital Twin and Open Source Information Infrastructure

Manuel Rexer [1]
Nils Preuß [1]
Sebastian Neumeier [1]
Peter F. Pelz [1]

1. Chair of Fluid Systems, Technische Universität Darmstadt, Darmstadt.

Date Submitted:
2024-05-28

Keywords:
FAIR, linked data, modular test environment, information model, experimental data, information infrastructure

Data availability:
Developed information models can be found here: `https://git.rwth-aachen.de/fst-tuda/public/metadata`

Software availability:
The developed software and their sources are listed in table 3

Abstract.
In this study, we apply the FAIR principles to enhance data management within a modular test environment. By focusing on experimental data collected with various measuring equipment, we develop and implement tailored information models of physical objects used in the experiments. These models are based on the Resource Description Framework (RDF) and ontologies. Our objectives are to improve data searchability and usability, ensure data traceability, and facilitate comparisons across studies. The practical application of these models results in semantically enriched, detailed digital representations of physical objects, demonstrating significant advancements in data processing efficiency and metadata management reliability. By integrating persistent identifiers to link real-world and digital descriptions, along with standardized vocabularies, we address challenges related to data interoperability and reusability in scientific research.
This paper highlights the benefits of adopting FAIR principles and RDF for linked data proposing potential expansions for broader experimental applications., Our approach aims to accelerate innovation and enhance the scientific community's ability to manage complex datasets effectively.

1 Introduction

In scientific research, effective data management is key, especially when dealing with experimental data. The increasing volume and complexity of data collected in experimental settings demand rigorous methodologies to ensure that such data remains findable, accessible, interoperable, and reusable (FAIR). These principles, established by Wilkinson et al. [1], are crucial for enhancing the transparency, reproducibility, and utilisation of research data across various scientific disciplines.

The primary aim of this research is twofold: to develop methodologies that make extensive datasets not only searchable and uniformly usable but also traceable and comparable across different studies. This is essential for building upon existing research without redundant experiments, thereby accelerating scientific discovery and innovation. Our approach involves a detailed examination of the test environment, which includes a wide array of measuring equipment and units under test. The reliability of data processing and the precision in uncertainty quantification heavily rely on our ability to thoroughly document and manage both raw data and its metadata.

The challenge in this context lies in effectively mapping all relevant information about the experiment, including its components and physical objects, while linking this information to the experimentally obtained data. To address this, it is essential to generate a digital representation of the objects used and ensure its availability for the measurements.

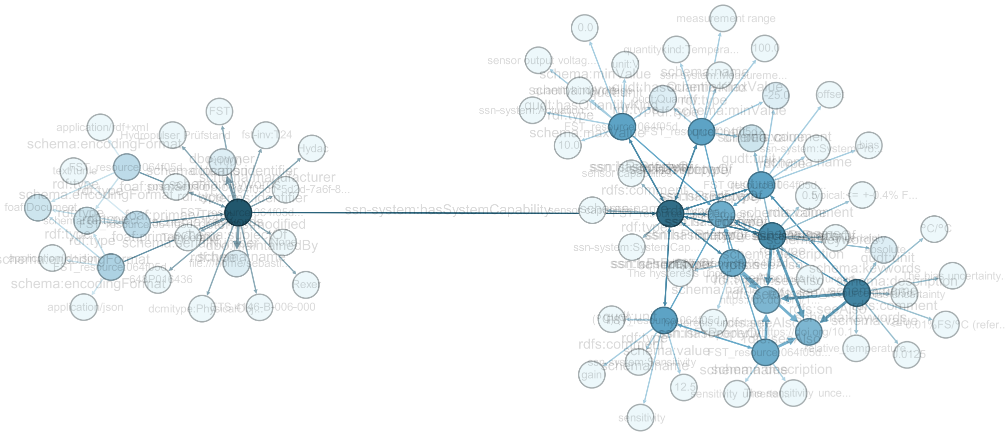


Figure 1: Graph of a digital data sheet of a sensor based on the developed information model. The colouring represents the connectivity (number of connected edges) of the different nodes where darker colours represent a higher, and lighter colours a lower connectivity. The data inside the graph is vaguely indicated by the faint text labels.

Using three key use cases from our modular test environment, we outline specific requirements for effective data and metadata management. This paper presents an overview of current advancements, technologies, and methods for making metadata FAIR. To meet these requirements, we develop information models and implement a robust working environment to provide and easily access the necessary information. Figure 1 shows an example for a populated information model, that results in a semantically enhanced, detailed digital data set of a real-world object. Since the data set closely describes the traits of the physical object it can be seen as a digital depiction or a digital twin. The infrastructure for providing this information is based on open source resources. We demonstrate practical benefits and improved efficiencies in data management through the utilization of FAIR principles and our implementation.

Ultimately, this research exemplifies the broader applicability and significant advantages of adopting FAIR principles within experimental research frameworks, potentially guiding future utilization in similar settings.

2 Application Use Case and Requirements

In engineering researchers are involved with experimental test setups consisting of a large number of sensors and actuators connected and interfaced with digital data acquisition hardware and software. Depending on the research method and the research topic, these experiments can either be highly individualised and thus designed to answer exactly one question, or they can be universal test environments characterised by the fact that different questions and setups can be answered in a short time.

This paper deals with the latter type. It focuses on two particular challenges related to (meta)data management. Firstly, it is essential that the metadata can be captured as easily and quickly as possible, since the test bench is frequently reconfigured. Secondly, it is particularly important to correctly record the setup and the components used, as it is no longer possible to manually check the setup and compare it with the measured data once the test bench has been reconfigured.

The following is a description of such a test environment, where various dynamic and quasi-static tests are carried out on mechanical components that are mainly chassis components. From this, requirements on the metadata management are derived.

2.1 Test Environment

The considered uni-axial servo hydraulic test rig[1] is a modular test rig, where several different units under test with a high variety in sensor and application setups are investigated.

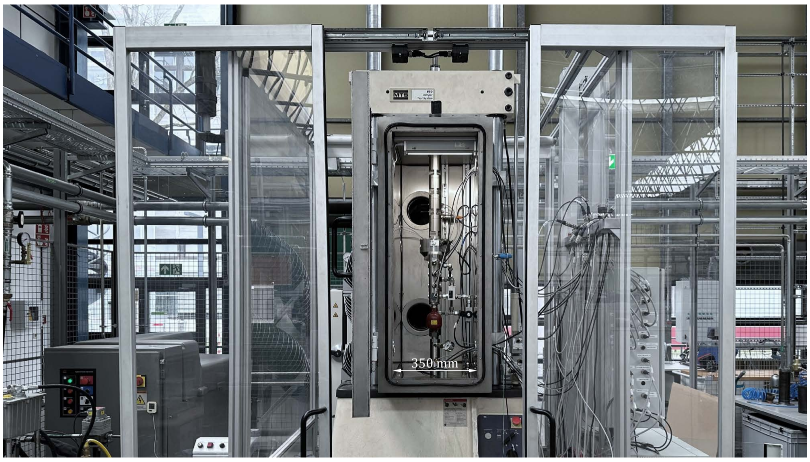


Figure 2: Uni-axial servo hydraulic test rig MTS 850 test damper at the chair of Fluid Systems at Technische Universität Darmstadt

Figure 2 shows the test rig providing dynamic testing in a temperature controlled environment in a range of $T = -40\,^\circ\mathrm{C} \ldots 100\,^\circ\mathrm{C}$. Dynamic forces of $F = \pm 50\,\mathrm{kN}$ at a cylinder stroke of up to $\Delta z = 300\,\mathrm{mm}$ are possible [2]. This allows for a large number of static and dynamic experiments to be performed. For example materials are tested for their strength, while dynamic transfer functions of springs or dampers may also be determined. The used measurement hardware is a dSPACE MicroLabBox with 32 analog in- and 16 outputs and all common digital interfaces. The box is able to simulate in real time and is therefore suited to apply Hardware-in-the-Loop investigations [3]. All this illustrates that very heterogeneous test setups with a large number of different sensors can be examined on this modular test rig.

---

[1] IRI: `https://w3id.org/fst/resource/018beaa3-8fe6-7ab5-83f7-81468a8a8784`

Figure 3 shows two very different test objects as examples. Both are chassis components for passenger cars with different complexity. The steel spring (left)e is characterized simply with a deflection and a force sensor. The active air spring (right) [3]–[6], on the other hand, has more degrees of freedom. The pressure and temperature in the spring must also be known, and additional displacement sensors are required to control the active system. This experiment also requires additional components, such as an external energy supply. In addition, the properties of the active air spring depend on the gas used, in this case dry air.

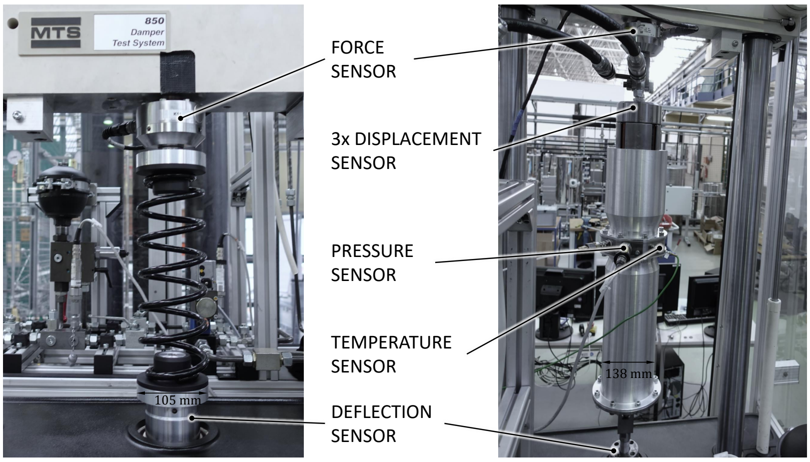


Figure 3: Two examples of different test objects whose dynamic properties are investigated. The left coil spring is measured with a deflection sensor and a force sensor to determine the transfer function. The air spring on the right is in addition also equipped with temperature, pressure and other displacement sensors.

There is a wide variety of sensor and component suppliers and most of the investigated hardware are new developments. Therefore, there are not always data sheets, let alone data sheets in a standardized or machine actionable form, available. This shows the need of a universal, efficient and easy metadata handling for this test environment.

From this modular setup test environment, three types of used objects can be identified, which are described by similar information. For each of which information models are developed:

M1 Components (in-house developed as well as purchased)

M2 Substances (e.g. dry air which is used in air springs)

M3 Sensors (varying suppliers)

2.2 Research and User Objectives

Following we give the main objective for our way towards FAIR measurement data for the specific modular test rig as well as general requirements. The requirements are to be established on the basis of the following three specific but also generalizing examples or tasks which are typical during the described experiments. Furthermore, the aforementioned information models are to be considered in conjunction with the specified requirements. Additionally, there are requirements pertaining to experimental and measurement data.

Although the goal is to align as closely as possible with the FAIR criteria, including the sub-criteria mentioned in [1], the approach taken here is to derive the requirements from the presented practical examples. While there is overlap between the identified requirements and the FAIR criteria, this overlap is not addressed further.

1. Using basic sensor information during data acquisition A typical task known by nearly every experimenter is to add the sensor characteristics to the data acquisition environment. Each sensor has an individual characteristic. In our case, these are exclusively linear characteristics, which does not necessarily apply to all sensors. These characteristics can change over time, which is why the sensors should be calibrated regularly. The sensor characteristics information must be clearly assigned to the input channels of the data acquisition.

As already mentioned, there are plenty of sensor manufacturers all of whom provide sensor information for their sensors, but there is no standard on how to provide this information. Therefore, a universal sensor information model should meet the following requirements R_i.

R1 Information must be associated with a unique ID. (M1, M2, M3)

R2 The ID must be persistent. (M1, M2, M3)

R3 The information must be retrievable via ID (either known source or via protocol). (M1, M2, M3)

R4 The information must be machine actionable[2]. (M1, M2, M3)

R5 The sensor information model must include sensor characteristics (e.g. sensitivity and bias). (M3)

R6 The information models should allow for changing information (and or redundant but non-conflicting information). (M1, M2, M3)

2. Tracing of data results back to component and substance information Experimental data is always subject to further processing and analysis, which can lead to new questions. It is important to note that the experimenter and the scientist who conducts further analysis may not be the same person. It is also crucial to identify the components and their properties used in the experiment as their properties are dependent on the environmental conditions, particularly when fluids are involved. Therefore, the ambient conditions should be recorded in the experiment.

The following requirements for the different information models to be developed are derived from this problem.

R7 The component information model must allow representation of relevant quantities. (M1)

R8 Measurement results must be linked to measured data as well as component information, which is used for model parameterization or as input in some other computation.

R9 Relevant physical properties should be able to depend on other variables. (M1, M2, M3)

R10 Measurement results must specify which information to use when redundant data is provided (e.g., multiple measurement ranges of a sensor).

---

[2] Machine actionable means that the data can be processed automatically without human interaction.[7]

3. Using sensor information for uncertainty quantification When evaluating experimental data, it is essential to determine both systematic and stochastic uncertainties. This process involves interpreting the uncertainty information provided by sensor manufacturers in data sheets or calibration certificates and assigning it to the corresponding measured variables. However, since sensor manufacturers do not adhere to standardized formats for reporting uncertainty —such as those outlined by the Joint Committee for Guides in Metrology [8]— this interpretation is laborious and time-consuming. Once completed, the extracted uncertainty information should be available in the sensor information model to ensure accessibility and consistency.

R11 The sensor information model should include and differentiate typical uncertainty information. (M3)

R12 The sensor information model should also specify the uncertainty information and the source of them. (M3)

In general Hardware, substances, and sensors can be used at various test benches with different data recording environments. This affects the reusability of the information and also consecutive the choice of frameworks used to model the information. The usage at different test environments also suggests that other aspects of a sensor or component may be significant, necessitating a flexible information model.

Also this flexible standard information model must be flexible enough to be able to be reused and adapted to describe different objects, where reoccurring descriptions for similar objects (e.g. sensors) could therefore be made a information model again. This assures a recursive and modular workflow. Also the whole process should be easy to use to get the users up and running fast.

R13 The information models must be compatible to various measurement setups. (M1, M2, M3)

R14 The information models must be hardware independent and, therefore, be deployable in various experimental setups. (M1, M2, M3)

R15 The information models must be programming language independent. (M1, M2, M3)

R16 The information model[s] must be flexible and easily expandable.(M1, M2, M3)

R17 Version control of the information models is needed. (M1, M2, M3)

R18 Access control to the information models must be provided. (M1, M2, M3)

Experience has shown that test bench modifications require simple processes for connecting and adding information. The more manual effort is required, the greater the likelihood of neglect or bypassing the process by the experimenter. This can call the reliability of measurement metadata into question and may even require repeating measurements.

R19 All information that is already available digitally should be collected automatically.

3 State of the Art and Relevant Standards

Numerous works and projects have focused on making research data FAIR, as seen in [9]–[11].

The FAIR principles serve as guidelines. However, the specifics of how to achieve FAIR data are still developing. Although future technologies may enhance these processes [12], current efforts involve technologies and ideas from the semantic web, linked data and knowledge management [13]–[15]. Since the early days of the Internet, technologies have been developed to facilitate the interoperable availability of knowledge. Best practices and guidelines are provided in resources like the FAIR CookBook [16].

Persistent Identifiers (pID) A crucial element towards achieving FAIR data is the use of unique identifiers for specific objects [12]. A well-known example is the International Standard Book Number (ISBN), which identifies books. However, it is not a web link and thus information about the entity cannot be directly retrieved automatically. Consequently, the Digital Object Identifier (DOI) has become established for identifying books and other digital objects, such as published software code. It is important to note that the same object, such as a book, can have multiple identifiers, e.g., an ISBN and a DOI.

Semantic Web and Linked Data The web contains an overwhelming amount of data, prepared for human consumption, not standardized for machines. Humans can derive information from context, a capability machines currently lack without assistance. The Semantic Web aims to provide this assistance by making information available in a format that also machines can process [17]. Promoted by the World Wide Web Consortium (W3C) [18], this initiative offers a range of technologies and standards to facilitate this provision.

A core concept is the semantic presentation of data, contextualizing it through directed graphs where objects (nodes) relate via directed edges. The standard framework for this is the Resource Description Framework (RDF), also developed by W3C. For RDF-described information to be interoperable, standardized vocabularies for nodes and edges are necessary, facilitated by ontologies that store terminological knowledge.

The ultimate vision is a vast graph connecting knowledge across disciplines via standardized and formalized terms, forming Web 3.0 [19]. Hitzler et al. [17] provide a comprehensive overview of the Semantic Web. Relevant aspects for FAIR experimental data are summarized below.

Resource Description Framework (RDF) RDF represents relationships as subject-predicate-object triples, a standard developed since the 1990s and continually refined [20]–[22]. Multiple triples build a bigger directed graph. Various serializations of the graphs exist, such as Turtle or JSON-LD.

In RDF, Internationalised Resource Identifiers (IRIs) [23] are used.[3] These unique IRIs denote nodes and edges, serving as unique identifiers of information.[17]

Objects can be connected or assigned properties, and data values in RDF are represented as literals, which are character strings that can also have assigned data types. However, objects can also be labeled as instances of a class.

---

[3] Another option is the Uniform Resource Identifier (URI) [24], from which the IRI originated. The IRI expands the permissible character set. IRIs are used in this paper.

Ontologies As ontologies may not be familiar to every scientist, especially in engineering disciplines where experiments are common, four basic questions are answered below.

What is an ontology?

The term ”ontology” originates from philosophy and was characterized by Aristotle [25]. Since the late 20th century, the term has been adopted in computer science, where it refers to ”a formal, explicit specification of a shared conceptualization” [26].

Staab et al. [25] provide a detailed overview of what exactly is meant by this term. Noy et al. [27] offer a more practical definition: ”An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them” [27].

One of the well-known ontologies is the Friend of a Friend (FOAF) ontology[4], which was developed to describe relationships between people, i.e., social networks.

The main components of an ontology are:

- Classes and subclasses, which describe concepts via common properties. These are analogous to classes in object-oriented programming to implement the concept of generalization in contrast to individualization [28]. An example class that describes documents is foaf:Document.
- Attributes or properties that can be assigned to a class and, in turn, point to another class, e.g., foaf:maker, or contain a value, e.g., foaf:name.

This small example is shown as a graph in the following Figure 4. The relationship (predicate) between a document (subject) and a person (object) is created via the object property maker.

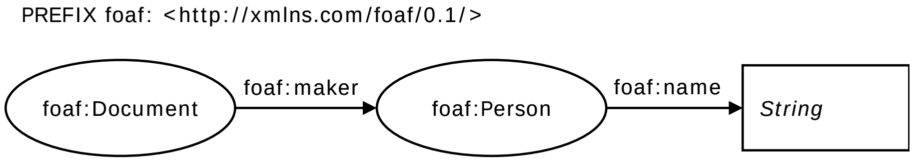


Figure 4: Simple example from the FOAF ontology, which represents the relationship between a document and a person with a name. Classes are oval and start with a capital letter by convention [17]. Properties that contain literal data are square.

Additional concepts such as owl:restriction allow the modeling of more complex concepts. The Web Ontology Language (OWL) [29], developed by the W3C, is often used to formulate complex ontologies.

Why develop and use ontologies?

For scientists in fields where ontologies are not the norm, the effort involved in creating and using ontologies might seem substantial with little perceived benefit for the individual researcher. However, data and information in the disciplines are often generated at significant expense in terms of time and money. They should therefore also be prepared in such a way that they can

---

[4] `http://xmlns.com/foaf/0.1/`

be reused. This is particularly evident in major initiatives for the FAIRification of data [30]. Considering the continuously increasing data-supported research, it is worthwhile to explicitly create knowledge and prepare it in a way that it can be used by others (programs). Ontologies offer this possibility and are already being successfully used in areas that rely on the research of others, e.g., in life sciences [31].

How to develop an ontology?

Ontology engineering is a science in its own right. There are several methods for developing ontologies, yet no single method has been established as the standard that must be strictly followed. Femi Aminu et al. [32] provide an overview of ontology development methods and categorize their advantages and disadvantages. Allemang and Hendler [28] provide practical guidelines for developing ontologies and also give an overview of existing ones.

A common learning from all methods is: If possible, use existing, well-maintained vocabularies [27], [33]. OBO Semantic Engineering Training also provides a guide on when not to develop an ontology [34].

A second lesson is that development should be focused on a defined domain or application [27]. In the first draft, it is acceptable not to cover every possible application. Any given information is better than none.

Thus, we develop information models for our application and utilize existing ontologies for this purpose.

What is the difference between an ontology and an information model?

The distinction between an information model and an ontology is not completely straightforward. In general, every ontology is an information model, but not every information model must be an ontology [35]. An information model therefore does not have to contain all the information that an ontology does. It is an application of the concept of an ontology to a specific problem.

Schulz et al. [36] provide an overview of the characteristics of both, shown in Table 1. However, the authors themselves state that in reality, there is no sharp distinction in the use of the two terms.

Ontologies	Information Models
Contain classes that have really existing domain entities (particulars) as members	Classes have information entities as members
Represent real-world particulars in terms of their inherent properties	Represent artifacts that are built to collect or annotate information
Can exist independently of information models as long as only the existence of particular things is recorded	Are required to record beliefs or states of knowledge about real things or types of things (as represented by ontologies)
Context-independent	Context-dependent

Table 1: Comparison of ontologies and information models from [36]

In our application, three information models are developed in RDF, which are based on existing RDF vocabularies and ontologies.

Since only existing RDF vocabularies and ontologies are used, the extended modeling capabilities of RDFS, or even more so that of OWL are not required and therefore RDFS or OWL are not used.

4 Modeling Approach and Implementation

Three information models for 1. sensors, 2. components, and 3. substances were developed from the requirements of the work on the test bench and the known methods of knowledge representation. These information models were instantiated for the objects, used on the test environment presented, and made available online for use.

4.1 Information Model

In the following sections, we present the three developed information models, M1, M2, M3. While these models share fundamental properties, such as a common method for assigning IRIs and the use of the same ontologies, they differ in the information they contain and their structure.

IRIs Each object is assigned a unique identifier, for which we use a Universal Unique Identifier (UUID), version 7 [37]. This is a 128-bit code that can be generated automatically using a Python library[5].

As persistent identifiers for the instances of our information model we use the *w3id.org* redirect service in combination with GitLab Webpages and GitLab repositories for each.

Used Vocabulary Various classes and properties are then linked to this object in a RDF graph. When linking, only known semantic vocabulary from established ontologies are used. The most important ontologies are summarised in the following table with their reference and their application domain.

abbreviation	URI prefix	application
rdf	`http://www.w3.org/2000/01/rdf-schema#`	RDF schema
dcTerms	`http://purl.org/dc/terms/`	general metadata terms
dcType	`http://purl.org/dc/dcmitype/`	general types
schema	`http://schema.org/`	general vocabulary
foaf	`http://xmlns.com/foaf/0.1/`	social networks
qudt	`http://qudt.org/schema/qudt/`	quantities and units
ssn	`http://www.w3.org/ns/ssn/`	sensor networks
ssn-system	`http://www.w3.org/ns/ssn/systems/`	systems for measurements
sosa	`http://www.w3.org/ns/sosa/`	sensors and actuators (based on ssn)

Table 2: Ontologies used with the abbreviation in the first column, the full link in the second column and the application area in the last column

---

[5] `https://github.com/oittaa/uuid6-python`

M1 Components The model of a component is the most basic model and consists of three levels of information, metadata, further documentation and physical properties. Figure 5 presents a simplified version of the model, with nodes printed in bold, and edge descriptions in thin font. Any parent nodes are outlined with a box, and descriptive properties are arranged in tabular form below.

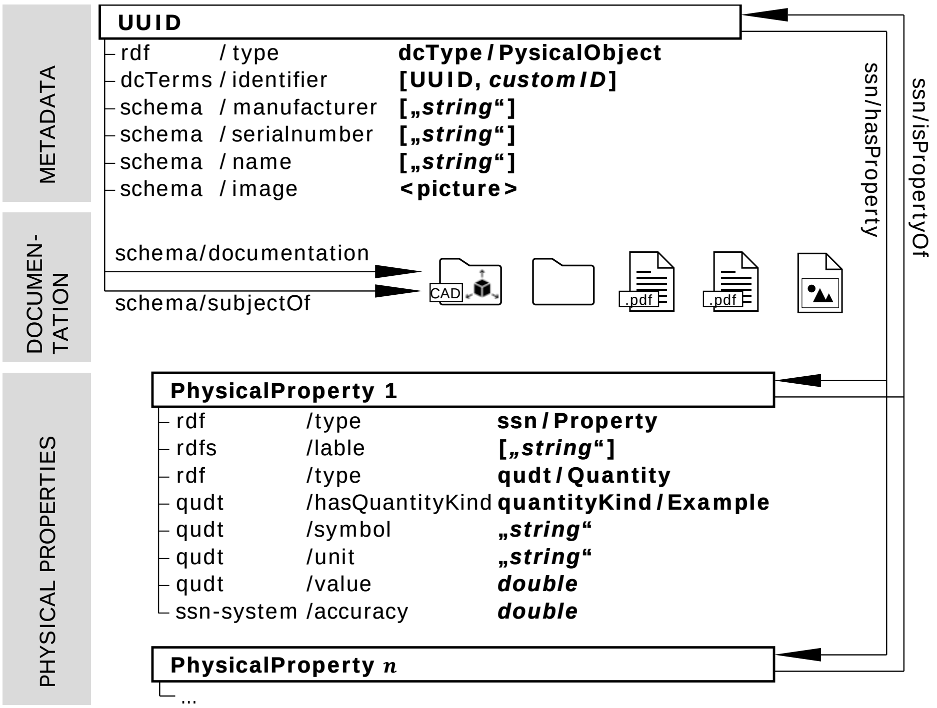


Figure 5: Simplified structure of the RDF graph of an information model of a component consisting of metadata, additional documentation and physical properties.

The metadata for the component is linked at the top level, defining it as a physical object with a name, serial number, and manufacturer, etc.. The UUID serves as the primary ID, but other IDs can also be assigned individually.

The second level provides additional information that cannot yet be processed by machines, such as images, CAD data, reports, and data sheets.

The third level contains relevant physical properties. It is important to note that not all properties of a component are specified. The user can specify the properties that are important for further processing during or after an experiment. The RDF graph is expandable, allowing for additional properties to be added at a later date if they are relevant for further investigations. The properties displayed here consist of a fixed value, such as the volume of an air spring, cf. Section 2.1. However, it is also possible to add characteristic fields, as shown in the next section on substances.

One example component developed at the chair of fluid systems is a gas spring which is experimentally investigated in the test rig presented in section 2.1. The information model of this component containing all its metadata and properties can be found at `https://w3id.org/fst/resource/018bb4b1-db48-73b8-9d82-8a8ffb6ee225.ttl`.

M2 Substances *Substances* in this context refers to `https://schema.org/ChemicalSubstance`, namely ”*a portion of matter of constant composition, composed of molecular entities of the same type or of different types*”. Only fluids, such as nitrogen or hydraulic oil, have been described in the context of the particular test environment.

The information model of a substance, as shown in Figure 6, is based on that of the component. The origin node **UUID** is described by metadata. Further documentation, such as safety data sheets, can also be referenced, or physical properties analogous to those of the component can be attached. However, these are not displayed in Figure 6 to provide a clearer overview.

However, the fluids used in the experiments whose information is shown here have physical properties that depend on the ambient conditions. This is particularly evident in the properties of gases, such as the density of nitrogen. The density ϱ depends on the temperature T and the pressure p. At pressures $p < 10$ bar the state can be described analytically with sufficient accuracy using the ideal gas law $p = \varrho RT$ and the specific gas constant of nitrogen $R_{N_2} = 297$ J/kgK [38]. At higher pressures the behaviour deviates from that of an ideal gas. Therefore, in standard works such as the CRC Handbook of Chemistry and Physics [39], the VDI Wärmeatlas [40] or the NIST Chemistry WebBook [41] density is given as a lookup table. The goal now is to adequately represent this property in the information model.

A very direct approach would be to include all values or combinations of values in the graph. However, this would lead to the graph becoming very large and confusing, and the actual coherent information of the table is lost. It is much easier to store the values in a suitable data format and refer to them in the information model, as well as describing the information stored in the file. This means that the values can also be quickly read into a suitable data processing system and used as a lookup table.

This is achieved in the model by using the open Hierarchical Data Format *.hdf5* [42]. The file contains the lookup tables for the thermophysical property as well as the column and row vectors that describe the table. The file is also described in the RDF graph, see Figure 6, where the source of the data is given. The thermophysical property of the substance is linked to the dataset.

The complete information model of nitrogen as an example can be found at `https://w3id.org/fst/resource/018dba9b-f067-7d3e-8a4d-d60cebd70a8a.ttl`. The stored data[6] originate from the NIST Chemistry WebBook [41].

M3 Sensors The last but not least information model represents sensors. Their Properties are basically the same as those of the **Component** and **Substance**, which are directly linked to the origin node **UUID**. Figure 7 summarizes them in the **Properties** category. Sensors can also process signals, and these capabilities are grouped together under the **SensorCapability** node in Figure 7.

This capability is further specified in the second level. The test environment only uses sensors with a linear characteristic, so the characteristic properties of the characteristic are described

---

[6] The data is stored in the file *nitrogen.h5* at `https://w3id.org/fst/resource/018dba9b-f067-7d3e-8a4d-d60cebd70a8a`

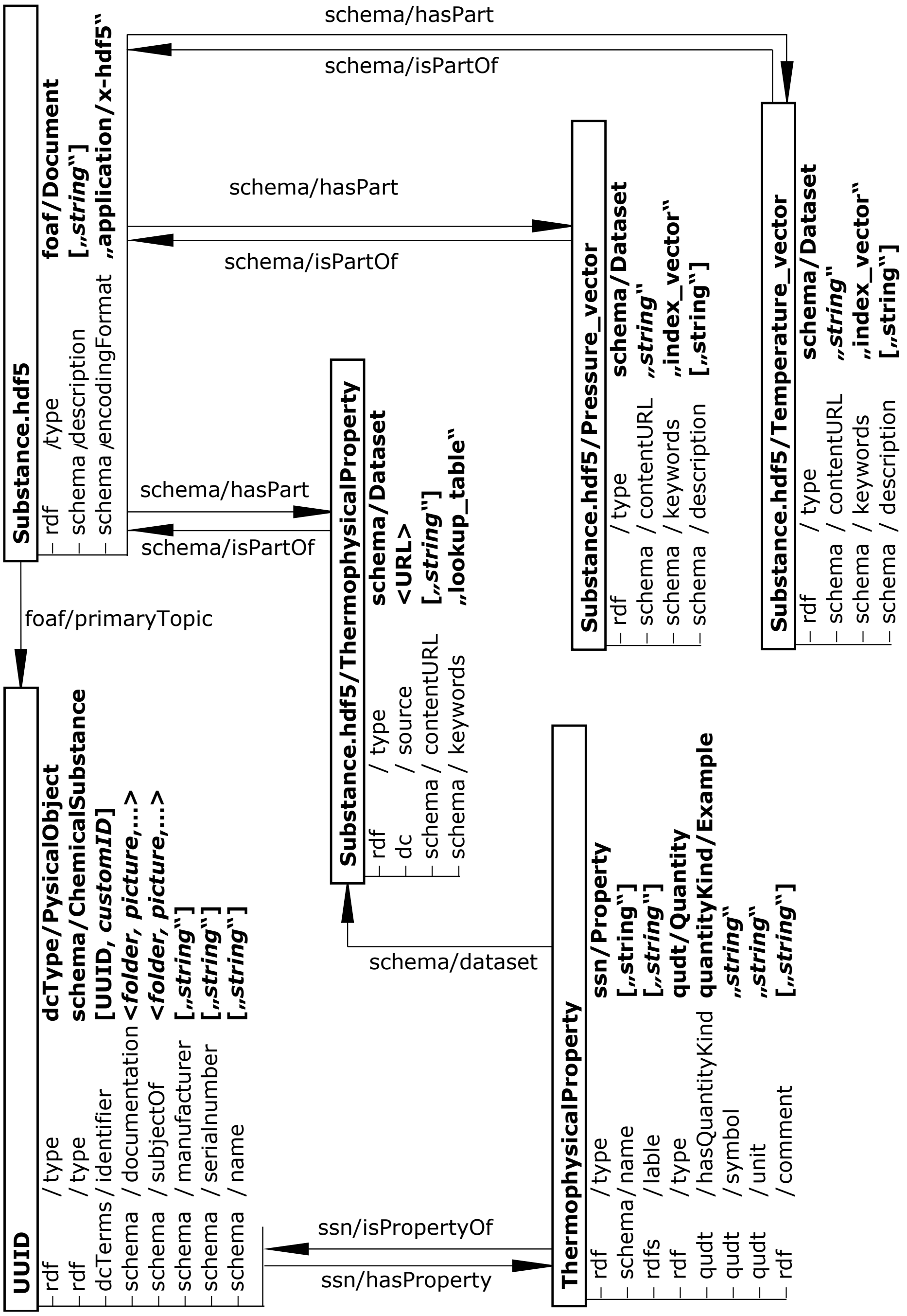


Figure 6: Simplified structure of the RDF graph of an information model of a substance. In addition to metadata of the substance, dependent thermophysical properties are also represented, the data of which is available as a lookup table.

by **Sensitivity** and **Bias**. In addition, the measuring range and the analogue output signal in the modelled case are specified under **SensorAcutationRange**.

Finally, the uncertainty properties are described by four classes **SensitivityUncertainty**, **BiasUncertainty**, **LinearityUncertainty**, and **HysteresisUncertainty**. This distinction complies to the publications [43], [44] and originates from the book by Tränkler [45]. The first two

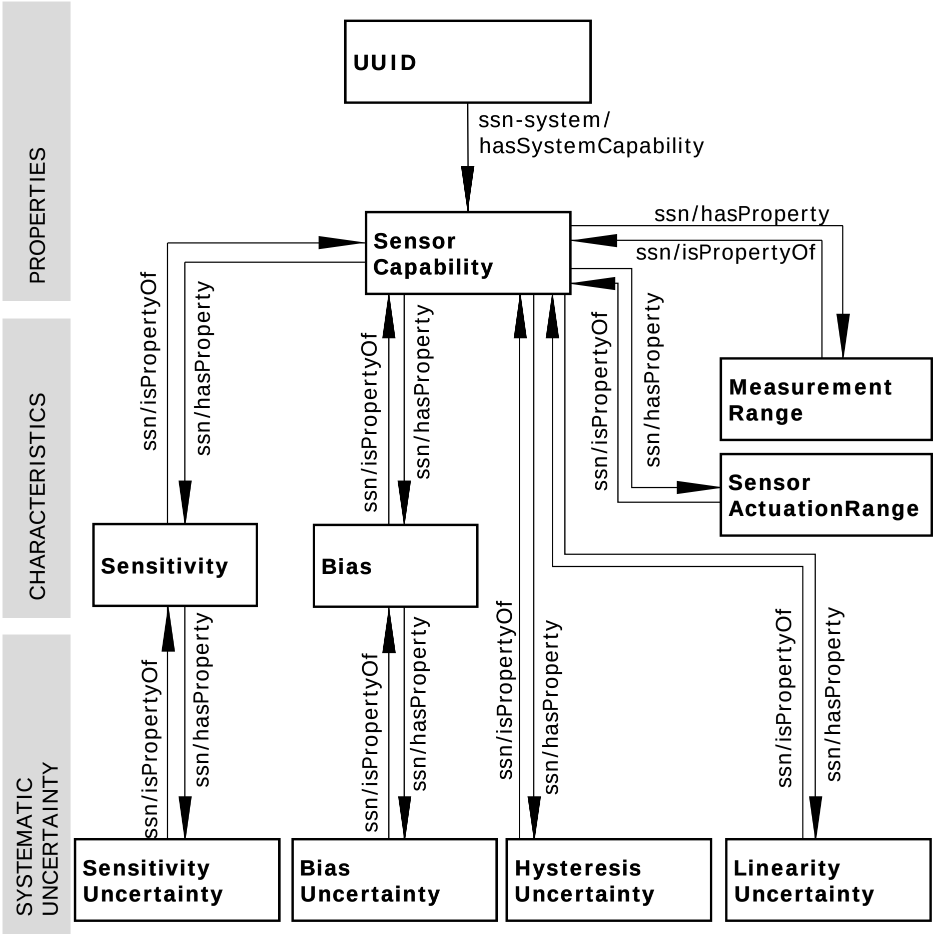


Figure 7: Overview of the main classes of the sensor information model.

uncertainty classes directly refer to the uncertainty of the sensitivity and bias parameters and are therefore linked to them. The last two uncertainties describe the entire characterisation and are therefore related to the sensor capability.

The complete model can be found in Figure 12 appendix 7. An example sensor can also be found under the following link `https://w3id.org/fst/resource/064f05d1-5d2d-7a6f-8000-a3da10f5a1a3`.

4.2 Implementation and Usage

In the following, we will briefly show how the specific models are generated, stored and made available for further use.

Instantiating the Models The Python RDFlib package [46] is utilised to generate the information models of specific entities, which can be serialised in various formats, including Turtle and JSON-LD.

For unique components, the model can be created manually, while for a larger number of objects, such as sensors, with the same description, they can be generated automatically from a table or a database. An example code is available in the following repository `https://github.com/test123-all/hydropulser-database-scripts`.

Storing Information Once the information is generated it has to be stored. As an online repository service we use GitLab [7] [47]. The generated information models and other descriptive files are stored there in repositories. This has the following advantages:

i. It allows the upload of files, folders and subfolders of any format.

ii. It has an integrated version control with git [48] (R17). This allows the user to continuously expand the information models (R16). In addition, a reference to the corresponding version (commit hash) can be used to reference and access a corresponding status.

iii. It offers integrated access control (R18). Repositories can be made public or private at any time. This can also be changed at a later date. Therefore, also data that is not to be publicly accessible can be uploaded and used.

iv. The web interface is able to render Markdown files making it possible to display human-readable information in a well formatted and easily digestible way.

One disadvantage is that no persistent identifiers are created. The URLs with which a repository, a folder or a file is called up depend on the paths within the repositories and their storage location. Therefore, a redirect service described below is required.

Providing Information - Redirect Service The long-term accessibility of information on the web depends on several critical factors. Pointers to resources must be unique per resource, even for multiple similar resources, which can be achieved using a unique ID (R1). Unique IDs must remain unchanged once established for a resource to ensure persistence (R2). The content associated with these IDs should also be easily retrievable using established web standards (R3) Choosing a URI namespace that incorporates one unique UUID per item, like “https://w3id.org/fst/resource/UUID” and using that string as ID should be sufficient to achieve uniqueness on the world wide web for every item (R1, R2). The “https://” part also indicates that these persistent IDs can function as a direct mechanism to retrieve the information. For information hosted on platforms like GitLab, like in this example, where URLs may change over time, a persistent entry point is required. This entry point must allow for a flexible redirection to the current location of the information and must be modifiable as needed to ensure continuous accessibility.

The w3id.org web service, provided and maintained by the W3C Permanent Identifier Community Group, offers an open and permanent URL redirection service [49]. This initiative is also backed by organizations with the joint goals to keep the longevity of the domain and also the webserver functioning [49]. The deployed web server relies on .htaccess files to define redirection rules [49], which are typical for the Apache HTTP Server open-source project [50]. The service is managed through a GitHub repository that is linked on their website, with the earliest pull requests dating back to mid 2013 [49]. This indicates that the service should have been active for over a decade and also continues to show significant activity.

---

[7] The instance is provided by RWTH Aachen University: `https://git.rwth-aachen.de/`.

Therefore the persistence of w3id.org URLs can be assumed, making the w3id.org web service a reliable choice as the entry point for persistent IDs. Since both the web server and the entire w3id.org repository are open source, the entire ecosystem could be reconstructed by a third party if necessary.

The following sections provide a more detailed description of the complete redirection service.

The information is stored in a directory in a GitLab repository and is therefore accessible online. The directory name is the UUID. By providing the information via an online platform, it is independent of the specific test environment (R14) and can be used on multiple test benches (R13).

The directory contains three representations of the information model RDF graph: a turtle-file (ttl), a JSON-LD-file, and an XML-file. This redundancy allows users to choose the representation that suits them best without the need for conversion. Additionally, a markdown-file is used to store a human-readable representation of the information, which GitLab is able to render in the browser. Any further documentation, such as data sheets or images, are stored in simple directories, as previously described in the information models.

An example directory structure is shown on the right-hand side in Figure 8. The figure also demonstrates how the information can be retrieved from any requesting program.

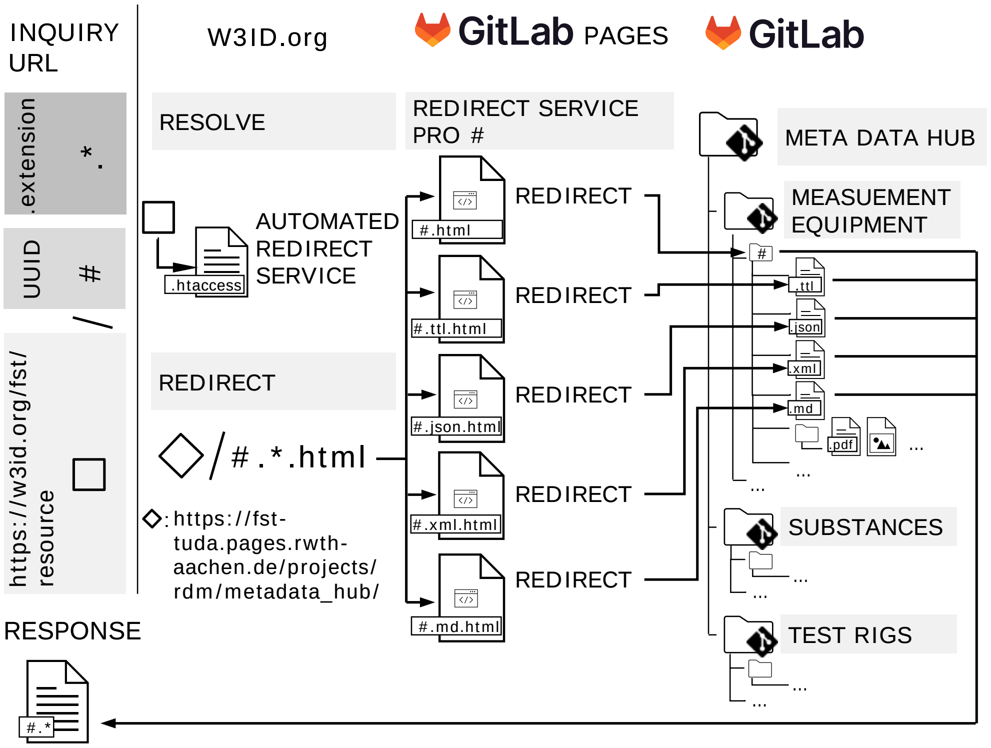


Figure 8: Two stage redirect service to get data from the META DATA HUB repository on GitLab using W3ID.org and GitLab pages.

A http(s) GET request is used to access information. It consists of a prefix symbolized with a square □, the UUID, #, and the desired file extension, .* . The extension specifies which representation of the information is required. If no type is specified, the request is forwarded to the repository.

First the request is sent to the w3id.org web server defined by the prefix □ that functions as a redirect service. There the URL is automatically reassembled using rules stored in an .htaccess-file. For a given UUID # and extension .* the assembled URL redirects to an automatically generated HTML-file stored in GitLab Pages, which in turn points and redirects to the requested file in the repository through a html meta refresh tag.

This allows the file to be returned as a response, provided that a corresponding HTML-file exists in GitLab Pages for the requested file in the repository.

This two stage redirect has three advantages:

i The W3ID service is maintained and hosted by a large community and is therefore likely to be available for a very long time (persistence).

ii The first stage of the automated redirect at w3id does not need to be updated even if names and paths in the GitLab repository change.

iii The redirect service at GitLab Pages can be created using an automatic program, therefore maintaining the redirects requires very little effort overall (R19).

CI/CD Pipeline for Automated Update of the Second Redirect Stage The functionality of the developed software[8] that automatically generates the HTML-files is described in more detail in the following. Additionaly, the software is embedded into a CI/CD Pipeline combined with GitLab Pages to further automate the process.

The primary objective of the second stage redirect is to establish a single, primary base URL that does not require frequent updates and resolves all UUIDs to their corresponding repository and directory. Both the repository and directory path may undergo changes. For instance, the repository location could shift within the GitLab instance, or the data set directory location could alter in relation to the repository. Both actions result in alterations to the URL, which would necessitate updates to the w3id service. Updating the URLs within the w3id.org service would be an unfeasible amount of work for the w3id service team. Therefore, it is necessary to implement a second stage in the redirect process, whereby the URLs are managed automatically by the user.

The CI/CD Pipeline of the META DATA HUB repository is depicted in Figure 9. Initially, the user updates or creates new data set directories within one of the sub-module repositories. Subsequently, the user must also update and commit the modified sub-modules within the META DATA HUB repository. Every commit uploaded to the main branch of the META DATA HUB repository initiates the CI/CD pipeline. The GitLab CI/CD pipeline[9] configuration is stored as the *.gitlab-ci.yml*-file within the root directory of the META DATA HUB repository.

The initial step undertaken by the pipeline is to establish the requisite environment. This involves the download of requisite software and the recursive cloning of the META DATA HUB repository. The recursive clone ensures the replication of the META DATA HUB repository and all its sub-module repositories, which contain the data set directories.

---

8. The software can be found at `https://github.com/test123-all/html-redirect-file-generator`

9. Further information on how to configure GitLab CI/CD pipelines can be found at `https://docs.gitlab.com/ee/ci/pipelines/`.

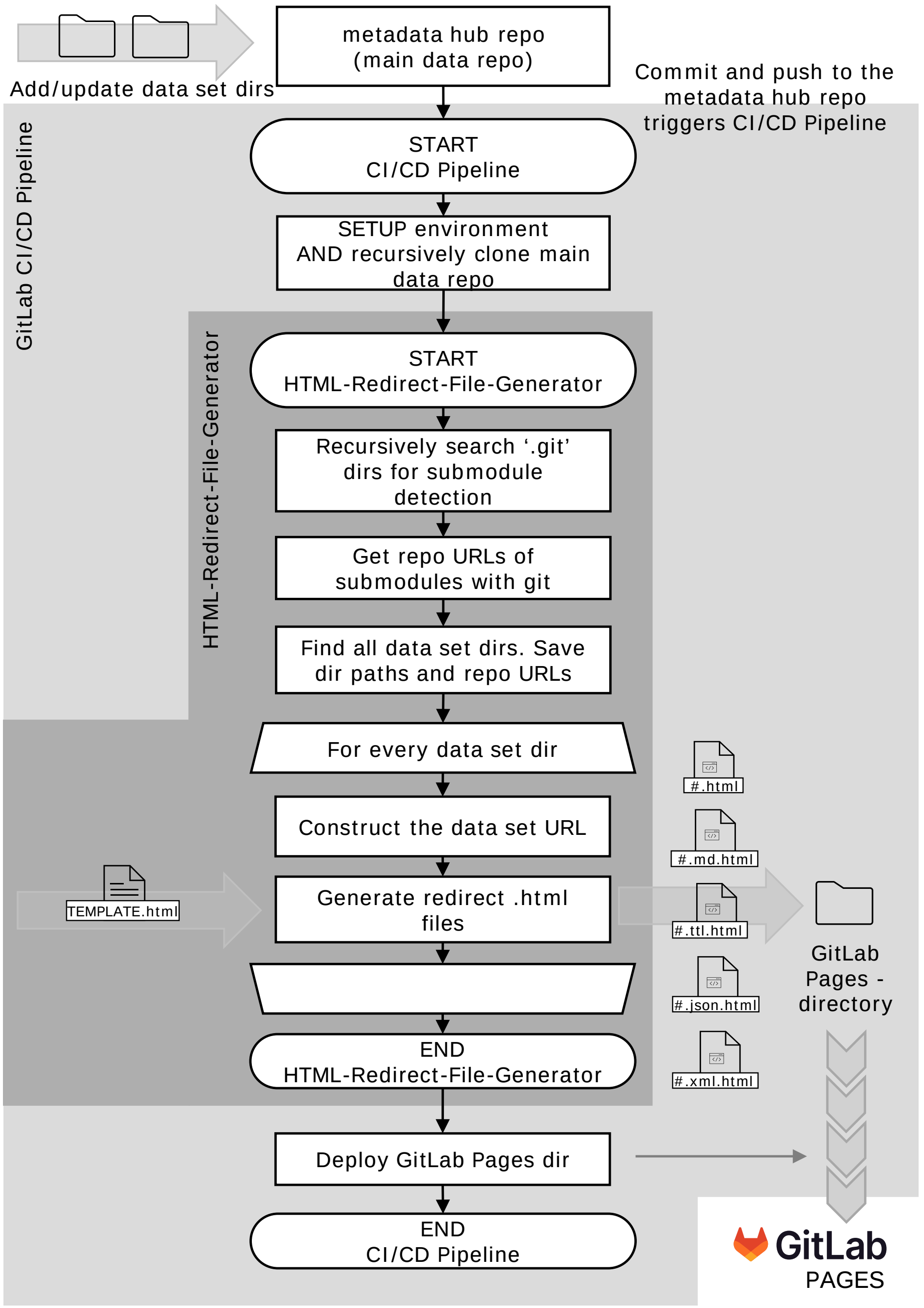


Figure 9: CI/CD Pipeline of the META DATA HUB repository on GitLab using the HTML-File-Redirect-Generator and GitLab Pages to generate and host the HTML-redirect-files of the second redirect stage.

The next stage is the initiation of the HTML-File-Redirect-Generator. The following input arguments are required: the path of the cloned META DATA HUB repository and the path where the HTML-redirect-files will be saved to. The HTML-redirect-files path is set to the GitLab Pages directory. The GitLab Pages directory is a special directory path within the pipeline where the HTML-files must be saved in order for them to be accessible and deployable by the GitLab Pages web service.

The HTML File Redirect Generator employs a recursive search of the META DATA HUB data directory and its subdirectories to identify directories with the extension ”.git.” Each cloned repository, including its submodules, contains a ”.git” directory at the root level, which enables the distinction of these directories and the retrieval of their respective directory paths in relation to the META DATA HUB data directory.

In the subsequent step, the URLs of the distinct repositories can be retrieved by executing a git command at the location of the different root paths of the submodules. The URLs are also stored for later retrieval. Subsequently, all data set directories for the distinct submodules are identified by a location convention within the different submodule directories. The data set directory paths are also saved to a list for later retrieval.

For each data set directory, a URL must be constructed that includes the repository URL of the data set directory in question, as well as the directory path of the data set relative to the repository directory. Subsequently, for each data set directory URL and a HTML redirect template, the main redirect URL for the dataset and the different file type redirects (.ttl, .json, .xml, .md) are constructed, parsed within the template and saved as their own file. The HTML files are saved to the GitLab Pages directory. This results in five redirect files as shown in Figure 9 on the right-hand side.

Once the HTML redirect files have been created, the HTML File Redirect Generator terminates and the process is handed back to the CI/CD pipeline. In the subsequent step, the pipeline deploys the GitLab Pages directory to the GitLab Pages web service, which is also shown on the right-hand side of the Figure 9 at the bottom.

Subsequently, the pipeline reaches its final state and successfully terminates, ultimately providing the automatically generated HTML files for the second stage of the redirect, as illustrated in Figure 8.

5 Application

Now that the implementation is known and the data is accessible, the question remains as to how the data can be integrated into the environment that is being used. A measurement recording program was created using the MATLAB software [51] and the Simulink simulation environment [52] due to proprietary restrictions of the test environment. In order to be able to use the RDF data in MATLAB it is necessary to develop a MATLAB package that downloads the RDF information, extracts it from a turtle file and converts it into a MATLAB *struct* data structure.[10] If the information is not publicly accessible, a personal access token is used to obtain access authorisation. With the help of the IRI, the information can be used both for recording and analysing measurement data. The information is therefore traceable to a single source, without the need to keep values inside different scripts updated. It is also possible to extend the recorded data afterwards through loaded information, that were not explicitly recorded during the experiment, to generate new knowledge or easily check for anomalies that were not obvious before. Ultimately this leads to processes and workflows that do not need the rerecording of time- and cost-ineffective measurements that do not provide much added value.

[10] The software is available at:`https://github.com/test123-all/fst-rdf-utilities`

21 components, 6 substances and 162 sensors have already been created.[11] Experiments were conducted in the transfer project T12 of the CRC805 based on the created metadata. No reference can be made to publications that use this data as the experimental data is still being analysed. For validation purposes, the three example tasks and their workflow are presented below.

1. Using basic sensor information during data acquisition The sensor information is used in a Simulink model that specifies the measurement data acquisition on the software side using blocks provided by dSpace.

The IRIs are represented by a QR code to provide easy accessibility and are respectively permanently assigned to one unique entity of a physical sensor. This is combined with other human readable information on a label and attached to the sensor, cf. Figure 10. With the help of QR code readers, the IRI can easily be inserted anywhere in a computer program.

In Simulink, the IRI is used in a custom block to automatically retrieve curve information and metadata about the sensor from the repository. This is shown on the right hand side of Figure 10.

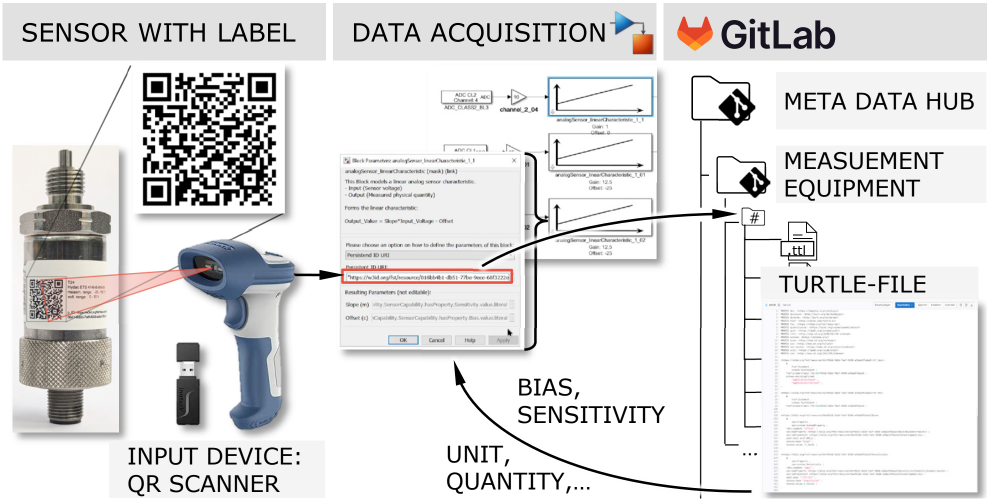


Figure 10: Interaction between IRI on the label of the sensor, the data acquisition software and the sensor information in the GitLab repository.

Overall, this approach meets all requirements R1-R6. Additionally, it offers the benefits of saving time when setting up new measurement environments and ensures that all relevant data is stored.

2. Tracing provenance of results to component and substance information To demonstrate how the information can be traced, let us examine the structure of a measurement in Figure 11. The data represents a measurement of hydraulic accumulators [53] and is stored as a MATLAB struct, which is a hierarchical data structure.

Each sensor provides a time series as measurement data. These series are highlighted in yellow. When examining the time series, it is important to note that in addition to the actual values, there is also metadata stored that pertains to the measurement itself, such as the unit of measurement and

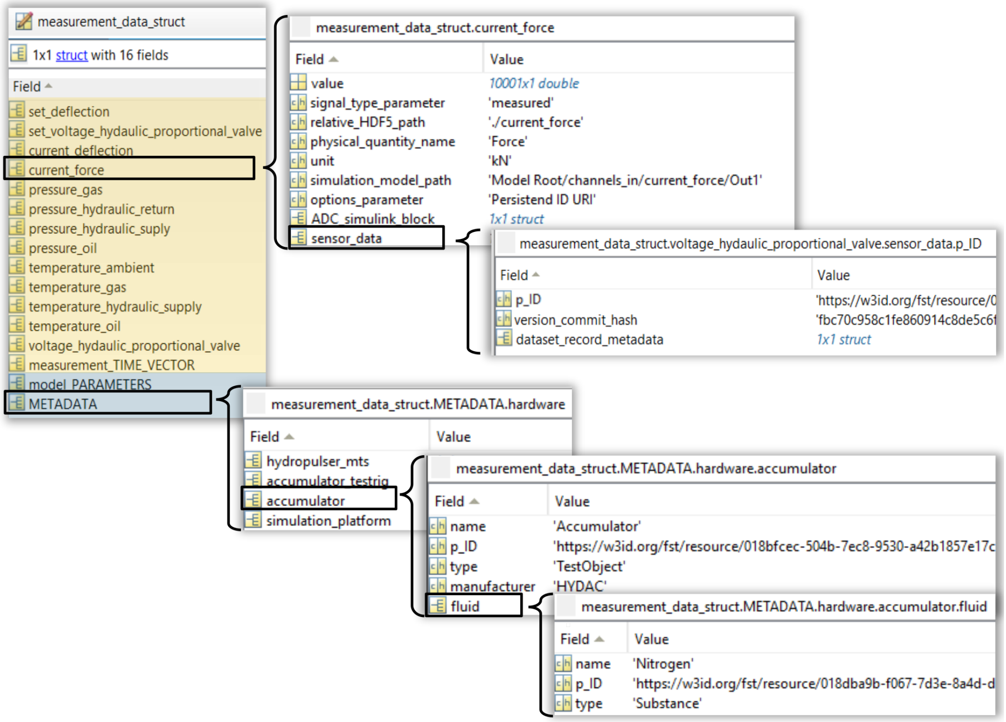
---

[11] Available at `https://git.rwth-aachen.de/fst-tuda/public/metadata`, although some of the data is not publicly accessible.

physical quantity. It is worth noting that additional information is not stored in each measurement file, but rather the link to the sensor is given under *sensor_data*. This allows for additional information to be obtained afterwards.

Two types of measurement metadata are also stored and highlighted in blue in Figure 11. Firstly, there are model parameters that can be set and read out, such as the excitation frequency. On the other hand, there is metadata which contains more general information, including details about the experimenter, software setup, and hardware setup.

The setup information is initially presented as a list to ensure all components used are recorded accurately. The information provided always includes the IRI to enable retrieval of all relevant information at any time. Objects can also be specified in a nested form. In this example, the accumulator is filled with nitrogen, as shown in Figure 11 at the bottom right. Additionally, a type is specified for all components, with the label **TestObject** used to identify the analysed component. It is important to note that the used label is not part of any standardized vocabulary. However, sosa:FeatureOfInterest could be a first suitable option.



measurement_data_struct

1x1 struct with 16 fields

Field
set_deflection
set_voltage_hydaulic_proportional_valve
current_deflection
current_force
pressure_gas
pressure_hydraulic_return
pressure_hydraulic_suply
pressure_oil
temperature_ambient
temperature_gas
temperature_hydraulic_supply
temperature_oil
voltage_hydaulic_proportional_valve
measurement_TIME_VECTOR
model_PARAMETERS
METADATA

measurement_data_struct.current_force

Field	Value
value	*10001x1 double*
signal_type_parameter	'measured'
relative_HDF5_path	'./current_force'
physical_quantity_name	'Force'
unit	'kN'
simulation_model_path	'Model Root/channels_in/current_force/Out1'
options_parameter	'Persistend ID URI'
ADC_simulink_block	*1x1 struct*
sensor_data	

measurement_data_struct.voltage_hydaulic_proportional_valve.sensor_data.p_ID

Field	Value
p_ID	'https://w3id.org/fst/resource/0
version_commit_hash	'fbc70c958c1fe860914c8de5c6f
dataset_record_metadata	*1x1 struct*

measurement_data_struct.METADATA.hardware

Field	Value
hydropulser_mts	
accumulator_testrig	
accumulator	
simulation_platform	

measurement_data_struct.METADATA.hardware.accumulator

Field	Value
name	'Accumulator'
p_ID	'https://w3id.org/fst/resource/018bfcec-504b-7ec8-9530-a42b1857e17c
type	'TestObject'
manufacturer	'HYDAC'
fluid	

measurement_data_struct.METADATA.hardware.accumulator.fluid

Field	Value
name	'Nitrogen'
p_ID	'https://w3id.org/fst/resource/018dba9b-f067-7d3e-8a4d-d
type	'Substance'

Figure 11: Excerpt from the data structure of a measurement in MATLAB. The time series are highlighted in yellow and metadata of the measurement are marked in blue.

As no data analysis has been published yet, it is not possible to demonstrate how the results can be traced back to the components used. R8 and R10 are therefore only partly fulfilled. There are also plans to automatically create a graph of the experiment itself to enable searches for specific measurements.

3. Using Sensor information for uncertainty quantification Section 4.1 details how uncertainty information is integrated into the sensor model. This model supports storing multiple types of uncertainty data directly connected to the sensor characteristics as well as overall uncertainty information provided by the sensor manufacturer. We implemented this approach for various sensor types, including among others pressure, force, and displacement sensors, across different manufacturers. The provided uncertainty data can be easily and directly applied for quantifying measurement uncertainty. Additionally, it serves as a foundation for advanced uncertainty propagation methods, such as those implemented in a MATLAB framework [44], [54], enabling the transformation of sensor systematic uncertainty from the time domain to the frequency domain [43].

The IRI provides access to both general R11 and specific R12 uncertainty information.

In general The description of the information using RDF graphs ensures that it is independent of the programming language R15 and hardware R14 used. In order for the models to be used on any test environment, it is only necessary to program the appropriate interfaces to retrieve the information from the repositories and insert it in the measurement program (R13).

6 Conclusion

This research has highlighted the importance of FAIR principles in managing experimental data, demonstrating significant improvements in the accessibility, interoperability, and reusability of data through tailored information models and linked data technologies. By integrating persistent identifiers and standardized vocabularies within a dynamic test environment, we have streamlined data acquisition and analysis processes, enhancing both efficiency and reliability.

The application of these models within our test environment has not only reduced manual effort but has also increased the adaptability and scalability of our data management systems. This approach promises substantial benefits for future experimental research.

Moving forward, the focus will be on broadening the application of these models to include a wider range of experimental setups and to improve the usability and efficiency of the tools to build ontologies and information-models. These efforts will continually result in further supporting of the scientific community in achieving more systematic and effective research data management.

7 Appendix

Support If you are interested in using the proposed framework, please do not hesitate to contact the authors for further support under info@fst.tu-darmstadt.de. The required software code is already referenced in the text and summarised in the following table 3. The software code that is not explicitly mentioned in the text, but which is relevant for this paper, is summarised in table 4

Name	Description	URL
hydropulser-database-scripts	This repository contains the python scripts to create the RDF dataset files (.ttl, .json, .xml, .md) of the different information models. Some of the scripts have a template character, for example the one for the sensors, other ones are purely hard-coded. This software repository is mentioned in 4.2.	`https://github.com/test123-all/hydropulser-database-scripts`
HTML-Redirect-File-Generator	Software to generate the HTML-Redirect-Files for the second redirect stage of the persistend ID URI redirect service explained in more detail in 4.2.	`https://github.com/test123-all/html-redirect-file-generator`
FST RDF utilities	Software that is able to load graphs given by a main node into python dictionaries and matlab structs to be able to load and use a subset of RDF data more easily and efficiently without the need to break long established and intuitive data usage habits in Python and Matlab. The program needs to start the mapping of the graph into a hierarchical data structure at one main node and will traverse and load all sub nodes, their sub-nodes and so on, that are connected and directed away from them until there are no nodes left or got already used in the graph. This software gets mentioned in section 5.	`https://github.com/test123-all/fst-rdf-utilities`

Table 3: Software referenced in this paper

Name	Description	URL
FST Label Creator	Software (Python package without CLI or GUI yet) with which custom labels on PDF label sites (only DIN A4 for now) with chosen preconfigured label templates and digital spreadsheets (like from Microsoft Excel, Apache OpenOffice Calc or LibreOffice Calc) could be generated. The generated PDF label site[s] can be printed later on store-bought label sheets using a laser printer. The previously chosen preconfigured label template inside the software should match the label template of the label sheet. The software is in an initial and very raw state and therefore might be subject to significant changes in the future.	https://github.com/test123-all/fst-label-creator
NIST Scripts	The scripts used to download the data from the NIST Chemistry WebBook - Thermophysical Properties of Fluid Systems - website and to generate the HDF5 files from the downloaded data and the corresponding RDF files. The scripts have not been published to avoid potential conflicts with German or American law (since NIST is a U.S. government institute and agency). Additionally, NIST may have an interest in ensuring that we do not share scripts that download their data. For this reason, we have only used the scripts internally to improve our data workflow and provide feedback.	/ not available /

Table 4: Additional software used in this paper

Information Model of a Sensor The complete information model is shown in the following figure. As there are a relatively large number of nodes, they are grouped together. An RDF graph of an example sensor is available at https://w3id.org/fst/resource/064f05d1-5d2d-7a6f-8000-a3da10f5a1a3.ttl. This can be displayed, for example, with an online RDF visualiser https://issemantic.net/rdf-visualizer.

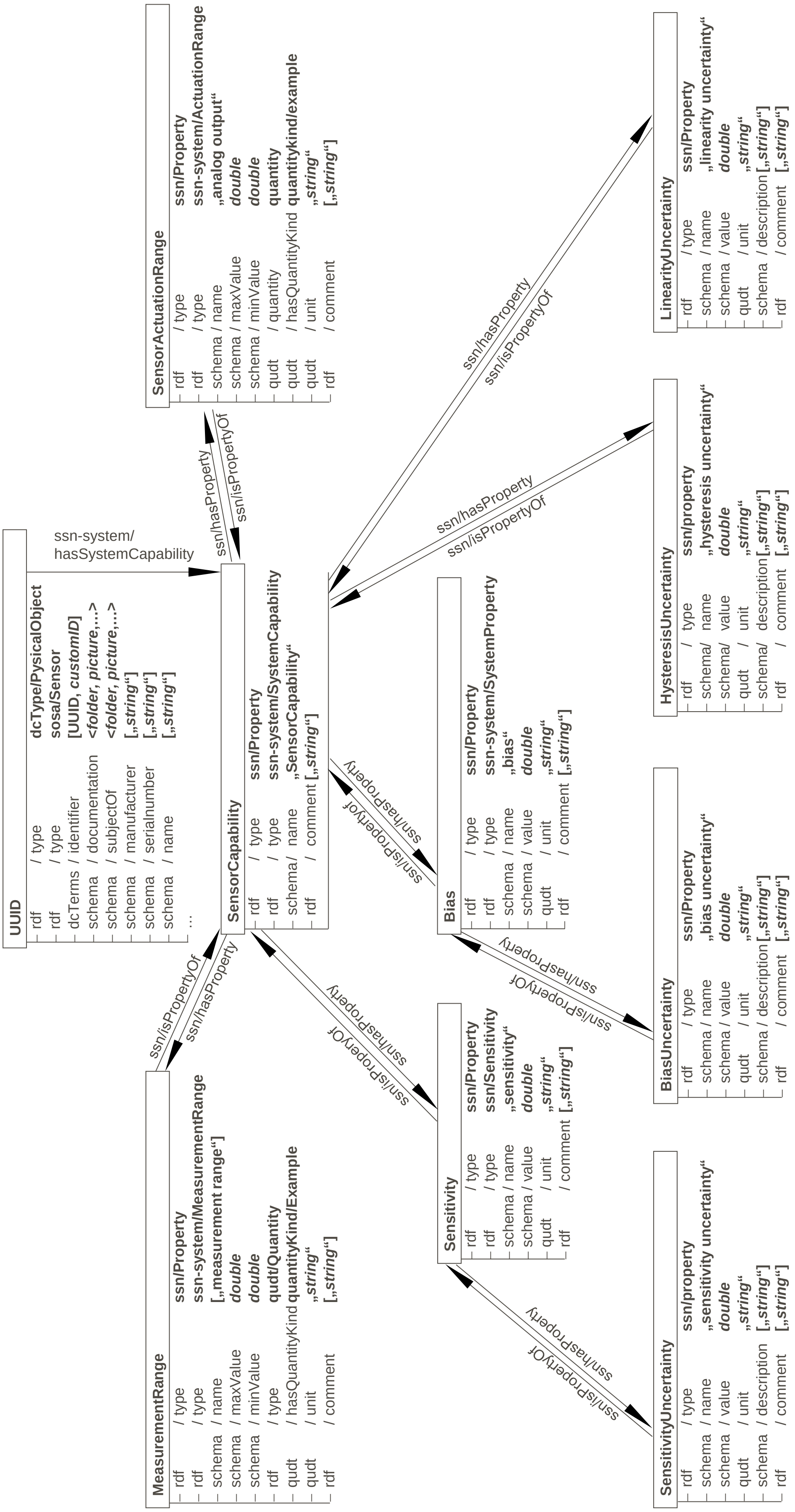


Figure 12: Complete sensor information model

8 Acknowledgements

Special thanks to the funding projects:

Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 57157498 – SFB805,
Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) –Project number 432233186 - AIMS,
Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) –Project number 442146713 - NFDI4Ing.

9 Roles and contributions

Manuel Rexer: Hardware setup, Conceptualization, Writing – original draft

Nils Preuß: Conceptualization, Writing – original draft

Sebastian Neumeier: Implementation, Documentation, Writing – review & editing

Peter F. Pelz: Supervision

References

[1] M. D. Wilkinson, M. Dumontier, I. J. J. Aalbersberg, *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, p. 160 018, 2016. DOI: 10.1038/sdata.2016.18.

[2] M. Puff and P. F. Pelz, *Entwicklung einer Prüfspezifikation zur Charakterisierung von Luftfedern* (FAT-Schriftenreihe). Berlin: Verband der Automobilindustrie (VDA), 2009.

[3] E. Lenz, P. Hedrich, and P. F. Pelz, "Aktive luftfederung – modellierung, regelung und hardware-in-the-loop-experimente," *Forschung in Ingenieurwesen*, pp. 1–15, 2018, ISSN: 0015-7899. DOI: 10.1007/s10010-018-0272-2.

[4] P. F. Pelz, P. Groche, M. E. Pfetsch, and M. Schaeffner, Eds., *Mastering Uncertainty in Mechanical Engineering* (Springer Tracts in Mechanical Engineering), 1st ed. 2021. Cham: Springer International Publishing and Imprint Springer, 2021, ISBN: 978-3-030-78353-2. DOI: 10.1007/978-3-030-78354-9.

[5] P. Hedrich, E. Lenz, and P. F. Pelz, "Minimizing of kinetosis during autonomous driving," *ATZ Woldwide*, vol. 120, no. 7-8, pp. 68–75, 2018.

[6] P. Hedrich, "Konzeptvalidierung einer aktiven luftfederung im kontext autonomer fahrzeuge," Dissertation, Technische Universität Darmstadt, Darmstadt, 2018.

[7] S. Islam, "Fair digital objects, persistent identifiers and machine actionability," *FAIR Connect*, vol. 1, pp. 29–34, 2023, 1, ISSN: 2949-799X. DOI: 10.3233/FC-230001. [Online]. Available: https://doi.org/10.3233/FC-230001.

[8] Joint Committee for Guides in Metrology, *Evaluation of Measurement Data—Guide to the Expression of Uncertainty in Measurement*. 2008. [Online]. Available: `https://www.bipm.org/en/publications/guides` (visited on 10/15/2021).

[9] European Commission, Directorate-General for Research, and Innovation, *Turning FAIR into reality – Final report and action plan from the European Commission expert group on FAIR data // Turning FAIR into reality : final report and action plan from the European Commission expert group on FAIR data*. Luxemburg and Hannover: Publications Office, Amt für Veröffentlichungen, and Technische Informationsbibliothek, 2018, ISBN: 978-92-79-96546-3. DOI: `10.2777/1524`.

[10] D. Hornung, F. Spreckelsen, and T. Weiß, "Agile research data management with open source: Linkahead: Ing.grid volume 1 issue 1 2023," 2024. DOI: `10.48694/INGGRID.3866`.

[11] S. Ferenz and A. Nieße, "Towards improved findability of energy research software by introducing a metadata-based registry: Ing.grid volume 1 issue 2 2023," 2023. DOI: `10.48694/INGGRID.3837`.

[12] B. Mons, C. Neylon, J. Velterop, M. Dumontier, L. O. B. Da Silva Santos, and M. D. Wilkinson, "Cloudy, increasingly fair; revisiting the fair data guiding principles for the european open science cloud," *Information Services & Use*, vol. 37, no. 1, pp. 49–56, 2017, ISSN: 01675265. DOI: `10.3233/ISU-170824`.

[13] M. D. Wilkinson, R. Verborgh, L. O. Da Bonino Silva Santos, *et al.*, "Interoperability and fairness through a novel combination of web technologies," *PeerJ Computer Science*, vol. 3, e110, 2017. DOI: `10.7717/peerj-cs.110`.

[14] A. Mazimwe, I. Hammouda, and A. Gidudu, "Implementation of fair principles for ontologies in the disaster domain: A systematic literature review," *ISPRS International Journal of Geo-Information*, vol. 10, no. 5, p. 324, 2021. DOI: `10.3390/ijgi10050324`.

[15] N. Jeliazkova, M. D. Apostolova, C. Andreoli, *et al.*, "Towards fair nanosafety data," *Nature nanotechnology*, vol. 16, no. 6, pp. 644–654, 2021. DOI: `10.1038/s41565-021-00911-6`.

[16] P. Rocca-Serra, W. Gu, V. Ioannidis, *et al.*, "The fair cookbook - the essential resource for and by fair doers," *Scientific data*, vol. 10, no. 1, p. 292, 2023. DOI: `10.1038/s41597-023-02166-3`.

[17] P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure, *Semantic Web: Grundlagen* (eXamen.press). Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ISBN: 9783540339946. [Online]. Available: `http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1582600`.

[18] W3C, *About us*, 12.04.2024. [Online]. Available: `https://www.w3.org/about/` (visited on 04/12/2024).

[19] T. Berners-Lee, *Giant global graph*, 2007. [Online]. Available: `https://web.archive.org/web/20160713021037/http://dig.csail.mit.edu/breadcrumbs/node/215` (visited on 04/12/2024).

[20] D. Brickley and R. V. Guha, *Resource description framework (rdf) schema specification*, W3C Recommendation, Ed., 1999. [Online]. Available: `https://www.w3.org/TR/PR-rdf-schema/` (visited on 04/08/2024).

[21] D. Brickley and R. V. Guha, *Rdf schema 1.1*, W3C Recommendation, Ed., 2014. [Online]. Available: `https://www.w3.org/TR/rdf-schema/`.

[22] RDF Working Group, Ed., *Rdf 1.1 concepts and abstract syntax*, 2014. [Online]. Available: `https://www.w3.org/TR/rdf11-concepts/`.

[23] M. J. Dürst and M. Suignard, *Internationalized resource identifiers (iris)*, 2005. DOI: `10.17487/RFC3987`. [Online]. Available: `https://www.rfc-editor.org/info/rfc3987`.

[24] T. Berners-Lee, R. T. Fielding, and L. M. Masinter, *Uniform resource identifier (uri): Generic syntax*, 2005. DOI: `10.17487/RFC3986`. [Online]. Available: `https://www.rfc-editor.org/info/rfc3986`.

[25] S. Staab and R. Studer, Eds., *Handbook on ontologies* (International handbooks on information systems), Second Edition. Berlin and Heidelberg: Springer, 2009, ISBN: 9783662499955. [Online]. Available: `http://www.loc.gov/catdir/enhancements/fy1312/2008943971-d.html`.

[26] R. Studer, V. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data & Knowledge Engineering*, vol. 25, no. 1-2, pp. 161–197, 1998, ISSN: 0169023X. DOI: `10.1016/S0169-023X(97)00056-6`.

[27] N. F. Noy and D. l. McGuinness, *Ontology development 101: A guide to creating your first ontology*, 2001.

[28] D. Allemang and J. A. Hendler, *Semantic Web for the working ontologist: Effective modeling in RDFS and OWL*, 2nd ed. Waltham, MA: Morgan Kaufmann Publishers/Elsevier, 2011, ISBN: 9780123859655. DOI: `10.1016/C2010-0-68657-3`.

[29] W3C, Ed., *Owl 2 web ontology language document overview (second edition)*, 2017. [Online]. Available: `https://www.w3.org/TR/owl2-overview/` (visited on 04/14/2024).

[30] S. Stall, L. Yarmey, J. Cutcher-Gershenfeld, *et al.*, "Make scientific data fair," *Nature*, vol. 570, no. 7759, pp. 27–29, 2019. DOI: `10.1038/d41586-019-01720-7`.

[31] OBO Foundry, Ed., *Obo foundry*, 2024. [Online]. Available: `https://obofoundry.org/` (visited on 04/14/2024).

[32] E. Femi Aminu, I. O. Oyefolahan, M. Bashir Abdullahi, and M. T. Salaudeen, "A review on ontology development methodologies for developing ontological knowledge representation systems for various domains," *International Journal of Information Engineering and Electronic Business*, vol. 12, no. 2, pp. 28–39, 2020, ISSN: 20749023. DOI: `10.5815/ijieeb.2020.02.05`.

[33] Z. Aloulen, K. Belhajjame, D. Grigori, and R. Acker, “A domain-independent ontology for capturing scientific experiments,” in *Information Search, Integration, and Personalization*, ser. Communications in Computer and Information Science, D. Kotzinos, D. Laurent, N. Spyratos, Y. Tanaka, and R.-i. Taniguchi, Eds., vol. 1040, Cham: Springer International Publishing, 2019, pp. 53–68, ISBN: 978-3-030-30283-2. DOI: 10.1007/978-3-030-30284-9{\textunderscore}4.

[34] N. Matentzoglu and S. Toro, *Creating an ontology from scratch - obo semantic engineering training*, 2024. [Online]. Available: https://oboacademy.github.io/obook/howto/create-ontology-from-scratch/ (visited on 04/12/2024).

[35] PedroD, *What is the difference between an information model and an ontology?* 2015. [Online]. Available: https://stackoverflow.com/questions/30562062/what-is-the-difference-between-an-information-model-and-an-ontology (visited on 04/12/2024).

[36] S. Schulz, D. Schober, C. Daniel, and M.-C. Jaulent, “Bridging the semantics gap between terminologies, ontologies, and information models,” in *Proceedings of the 13th World Congress on Medical Informatics*, ser. Studies in health technology and informatics, C. Safran, Ed., Amsterdam: IOS Press, 2010, pp. 1000–1004, ISBN: 9781607505877. DOI: 10.3233/978-1-60750-588-4-1000.

[37] K. R. Davis, B. Peabody, and P. Leach, *Universally unique identifiers (uuid): Internet-draft*, 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-uuidrev-rfc4122bis/14/.

[38] P. Stephan, K. Schaber, K. Stephan, and F. Mayinger, *Thermodynamik: Grundlagen und technische Anwendungen Band 1: Einstoffsysteme* (Springer-Lehrbuch), 19., ergänzte Aufl. 2013. Berlin, Heidelberg and s.l.: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-30098-1. DOI: 10.1007/978-3-642-30098-1.

[39] W. M. Haynes and D. R. Lide, Eds., *CRC handbook of chemistry and physics: A ready-reference book of chemical and physical data*, 91. ed., 2010 - 2011. Boca Raton, Fla.: CRC Press, 2010, ISBN: 9781439820773.

[40] VDI-Gesellschaft Verfahrenstechnik und Chemieingenieurwesen, Ed., *VDI-Wärmeatlas: Mit 320 Tabellen* (VDI/-Buch]), 11., bearb. und erw. Aufl. Berlin and Heidelberg: Springer Vieweg, 2013, ISBN: 978-3-642-19982-0.

[41] P. Linstrom, *Nist chemistry webbook, nist standard reference database 69*, 1997. DOI: 10.18434/T4D303.

[42] The HDF Group, *Hierarchical data format, version 5*, 2023. [Online]. Available: https://github.com/HDFGroup/hdf5.

[43] M. Rexer, P. F. Pelz, and M. M. Kuhr, “Uncertainty propagation of systematic sensor errors into the frequency domain,” [preprint] 2023. DOI: 10.2139/ssrn.4452038. [Online]. Available: https://ssrn.com/abstract=4452038 (visited on 11/29/2023).

[44] M. Rexer, P. F. Pelz, and M. M. Kuhr, *Propagation of systematic sensor errors into the frequency domain - a matlab software framework*, USA, [preprint] 2024.

[45] H.-R. Tränkler and G. Fischerauer, *Das Ingenieurwissen: Messtechnik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, ISBN: 978-3-662-44029-2. DOI: `10.1007/978-3-662-44030-8`.

[46] D. Krech, G. A. Grimnes, G. Higgins, *et al.*, *Rdflib*, 2023. DOI: `10.5281/zenodo.6845245`. [Online]. Available: `https://github.com/RDFLib/rdflib`.

[47] GitLab Inc., *Gitlab*, 2024. [Online]. Available: `https://gitlab.com`.

[48] J. Hamano, S. Pearce, and L. Torvalds, *Git*, 2024. [Online]. Available: `https://git-scm.com/`.

[49] W3C Permanent Identifier Community Group, *W3id.org - permanent identifiers for the web*, 12.02.2024. [Online]. Available: `https://w3id.org/`.

[50] Apache HTTP Server Project, *Apache http server "httpd"*, 2024. [Online]. Available: `https://httpd.apache.org/`.

[51] The MathWorks Inc., *Matlab*, Natick, Massachusetts, United States, 2019. [Online]. Available: `https://www.mathworks.com`.

[52] The MathWorks Inc., *Simulink*, Natick, Massachusetts, United States, 2019. [Online]. Available: `https://www.mathworks.com`.

[53] M. Rexer, P. Kloft, F. Bauer, J. Hartig, and P. F. Pelz, "Foam accumulators: Packaging and weight reduction for mobile applications," in *12th International Fluid Power Conference (12. IFK)*, Dresden: Technische Universität Dresden, 2020, pp. 181–188. DOI: `10.25368/2020.26`.

[54] M. Rexer, S. Neumeier, and M. M. G. Kuhr, *Propagation of systematic sensor errors into the frequency domain - a matlab software framework*, 2024. DOI: `10.5281/ZENODO.10532137`.