Deliverable D5.5 – Standards and Best Practices Report

Responsible Partner:	SINTEF	2022-11-16
Contributor(s):	Bjørn Tore Løvfall (SINTEF), Kristine Wiik (SINTEF), Casper Welzel Andersen (SINTEF),Treesa Rose Joseph (SINTEF), Stijn Donders (SISW), Noel Vizcaino (UKRI)	2022-11-16
Reviewer(s):	Kontantinos Sipsas (INTRA), Jesper Friis (SINTEF)	2022-11-25
Coordinator:	CMCL Innovations	2022-11-30
Dissemination Level:	Public	
Due Date:	M24	
Submission Date:	30. Nov 2022	

Project Profile

Programme	**Horizon 2020**
Call	**H2020-NMBP-TO-IND-2020-twostage**
Topic	**DT-NMBP-40-2020** **Creating an open marketplace for industrial data (RIA)**
Project number	**953163**
Acronym	**DOME 4.0**
Title	**Digital Open Marketplace Ecosystem 4.0**
Start Date	**December 1st, 2020**
Duration	**48 months**

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953163. It is the property of the DOME 4.0 consortium and do not necessarily reflect the views of the European Commission.

Document History

Version	Date	Author	Remarks
V0.1	19.10.2022	Bjørn Tore Løvfall, Kristine Wiik, Casper Welzel Andersen (SINTEF)	Initial structure and content.
V0.2	07.11.2022	Stijn Donders (SISW)	Added industrial relevance and comments
V0.3	11.11.2022	Treesa Rose Joseph, Kristine Wiik, Bjørn Tore Løvfall (SINTEF)	Content for Sections 4-9 added
V0.4	14.11.2022	Noel Vizcaino (UKRI)	Extended FAIR section
V0.5	15.11.2022	Bjørn Tore Løvfall (SINTEF)	Updated list of abbreviations, executive summary and lessons learnt
V0.6	16.11.2022	Bjørn Tore Løvfall, Casper Welzel Andersen (SINTEF)	Update of section 3 and general cleanup of the document
V0.7	25.11.2022	Kontantinos Sipsas (INTRA)	Review
V0.8	28.11.2022	Casper Welzel Andersen (SINTEF)	General cleanup of the document
V0.9	30.11.2022	Amit Bhave and Willem van Dorp	Finalization, approval of final version

Executive Summary

The current deliverable (D5.5) is a report that summarizes the standards and best practices collected from the Industry Commons Cooperation work package in the DOME 4.0 project.

It is clear that relying on standards and best practices is a way to make sure the burden on the data providers and consumers is as small as possible. In making the DOME 4.0 platform, we want to listen to the communities and reuse as much as possible from what is already created. By adhering to standards instead of creating new alternative solutions, and to listen to the best practices and try to follow them. For DOME 4.0 this relates to how data is documented and handled, but also to how the ecosystem is built and how it is used. This is a vast set of topics and in this report, we try to cover what we think is the most relevant for DOME 4.0.

We try to cover in particular data documentation through ontologies/taxonomies/vocabularies in addition to the FAIR principles and the separation of data and metadata. We also cover software development and software tools and technologies, but limited to what we think is relevant for DOME 4.0.

Table of Contents

List of Figures

1 Introduction

The purpose of this document is to collect all the standards and best practices that are encountered in the interaction and cooperation with other initiatives in other tasks in WP5, Industry Commons Cooperation. This will in turn be used as input to the technical tasks to ease their burden in planning and execution of their tasks.

1.1 Industrial Challenge

Already today, data providers and owners suffer from inefficiencies and tedious manual steps, required to populate material data fields in their databases, maintain/update this information, and correlate it with other sources. This underlines the need for efficient and user-friendly data population, which will also enable providing data services to consumers and users on the basis of secured and trusted data transactions.

In today's ever more digital world, such data population must adhere to established data governance practices. Accordingly, DOME 4.0 will adopt the FAIR Guiding Principles [1] for scientific data management and stewardship providing guidelines to improve the Findability, Accessibility, Interoperability, and Reuse of digital assets. This also includes guidelines for data provenance and sovereignty.

In line with the objectives of Open Science and Open Innovation, the challenge is to make data FAIR through an effective common information system that allows in particular business-to-business data sharing and enables new or improved products, processes and services. Such a system should take the form of a user-friendly, state-of-the-art marketplace that is open to all providers and users of data to maximise the spill over of knowledge across all economic sectors.

1.2 Objective

It is not the objective of this report to give the solution to all the industrial challenges mentioned in the above section, but the objective is to collect standards and best practices to make the job easier for other tasks in the DOME 4.0 project to address the industrial challenge while making the DOME 4.0 ecosystem. The results collected in this report has been continuously fed to the technical work packages to ensure as good a foundation as possible was available to make technical decisions.

2 Standardization and Data Documentation

2.1 Top/Mid level Ontologies

The OntoCommons project [2] aims at developing an Ontology Commons EcoSystem (OCES) [3] where they try to harmonize data documentation using ontologies and taxonomies. Enabling Findable, Accessible[1], Interoperable, Reusable (FAIR) data, (see Section 3.2 for more information) and intra- and cross-domain interoperability. This fits with the vision of the DOME 4.0 project as DOME 4.0 aims to create a Digital Open Marketplace Ecosystem enabling sharing of business-to-business (B2B) data across markets. At the heart of this lies semantic interoperability, and a key enabler for semantic interoperability is ontologies.

In DOME 4.0 we are making a semantic data exchange ontology and an ecosystem information model ontology. According to the landscape analysis [3] done by the OntoCommons project there are a handful of actively developed top level ontologies, with the most important being Basic Formal Ontology (BFO) [4], [5], Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [6] and Elementary Multiperspective Material Ontology (EMMO) [7]. EMMO has a strong focus on the material science domain, and has a strong holding within the European Materials Modelling Council (EMMC) [8] community, and it is then natural for DOME 4.0 to base its ontologies on EMMO. An important lesson learnt in OntoCommons is that it is better to map axioms from existing ontologies than to create new ones.

2.2 Domain Ontologies

There are a number of existing standardized ontologies that are relevant for the work in DOME 4.0. The W3C DCAT(2) [9] vocabulary is an OWL2 ontology. It includes elements from other external vocabularies particularly from The Dublin Core terms (DCTERMS) [10], Friend of a Friend (FOAF) [11] and the provenance ontology (PROV-O) [12]. These are all relevant for the semantic data exchange ontology, and has already been described in D3.1 [13, p. 1]. The description is not repeated here, but the text is also added to Annex 1 for convenience to the reader.

There are other domain ontologies that are relevant for DOME 4.0, but they are not as standardized as the ones mentioned above. The OntoCommons project collected a list of relevant ontologies [14] as part of their landscape analysis [15, p. 2], [16, p. 3]

2.3 Taxonomies/Vocabularies

DOME 4.0 aims at being a Marketplace of Marketplaces. It is then very important to use a vocabulary that is as familiar to the targeted industrial end users as possible. Most notable are the European Science Vocabulary [17] and the Review of Materials Modelling (RoMM) [18].

The European Science Vocabulary is based on CORDIS and is developed as a reference vocabulary. This will be relevant for categorising data and platforms connected to DOME 4.0.

[1] For accessibility ontologies plays a smaller role. It mostly depends on technical solutions, like retrievable data and metadata by identifiers and access to metadata even when the data is no longer available.

The Review of Materials Modelling is a book that aims at amongst other things define the vocabulary for materials modelling within the European materials modelling community. This vocabulary has been adopted and further refined by EMMO.

3 Best practices for connecting to data

This section deals with the best practices for connecting data. This includes the Findable, Accessible, Interoperable and Reusable (FAIR) principles, but also how the data and metadata should be treated, and what could be done to help make the data more FAIR.

3.1 Separation of data and metadata

Data can be small and concise, big and specific, ordered or unstructured, but mostly it is all those things and more. To better understand what a datum or a set of data contains and represents, information about the data is useful.

The description of the information about the data is called metadata and is essentially another piece of data [1] However, the metadata is usually well-structured, maybe even schematized and condensed in comparison to the data. Metadata should not contain any actual data values, but rather, describe what those values represent. For a simple table of data, the metadata can be a list of the table headers associated with a small description for each as well as other useful information, like unit, multiplicity or other.

Since the point of separation between "actual" data values is an important one, this will be exemplified. Let's assume that you have measured a Raman spectrum[2] for a given material. What is the data and metadata? There are no clear rules, but it is easy to agree on the numbers representing the spectrum to be considered as "actual" data. What about the units along the wavelength and energy axis? Since they provide context to the data, it makes sense to consider them as metadata. What about the temperature the experiment was performed at? One could consider that as metadata. But considering the temperature value as another datum in the dataset, and its unit as metadata, will still respect the point that metadata should not contain any "actual" data values.

When adding this contextual information, the metadata is describing the data, and by collecting sets of metadata together, it is possible to search for and handle the data that is useful for a given purpose more easily.

To achieve this searchability there is a need of separation of metadata from the data it describes. Using the previous data table example, this is equivalent to creating a separate datum that lists and describes the table headers, which is kept separate from the data.

Describing data with metadata proves to be extremely useful for the wide spectrum of challenging use cases that are targeted and used by global industry. The following sections explore usage and description of metadata, specifically naming them as data models. How they can be expanded upon for

[2] Raman Spectroscopy is a non-destructive chemical analysis technique, which provides detailed information about chemical structure, phase and polymorphy, crystallinity and molecular interactions.

sharing, achieving wider interoperability. Eventually resulting in fully semantic data models supporting a FAIR semantically- and data model-driven framework of data sharing.

3.2 FAIR

The FAIR Data Principles (Findable, Accessible, Interoperable, and Reusable) are a set of guiding principles proposed by a consortium of scientists from various organizations and companies to support the reusability of digital assets. They were published in 2016 and have since been adopted by research institutions worldwide [1], [19].

For data to be FAIR, the consortium defines the criteria in Table 1 taken from [1].

Table 1 FAIR data criteria

Findable	The data must be described with rich metadata, assigned a globally unique and persistent identifier, and registered or indexed in a searchable resource.
Accessible	The data and its metadata must be retrievable by their identifier using a standardized communications protocol which is open, free, and universally implementable, and which allows for an authentication and authorization procedure where necessary. Furthermore, the metadata must be accessible, even when the data is no longer available.
Interoperable	The data and metadata must use a formal, accessible, shared, and broadly applicable language for knowledge representation, use vocabularies that follow FAIR principles, and include qualified references to other data and metadata.
Reusable	The data and metadata must be richly described with a plurality of accurate and relevant attributes. In particular, they must be released with a clear and accessible data usage license, be associated with detailed provenance, and meet domain-relevant community standards.

International standards play a key role in making the data FAIR. The FAIR principles are broken down into several (technology independent) high-level concerns [20]. The consortium devised FAIR codes, which are listed with comments related to standards and the relevance for DOME 4.0 in Annex 2.

DOME 4.0 will develop a FAIR Data Monitoring and Auditing Service, which will make the adherence of data provided by participants in the context of DOME 4.0 to FAIR principles directly measurable. Additionally, DOME 4.0 will promote FAIR data through provenance tracking. For this work, DOME 4.0 will look to other projects and communities and try to build on what has already been achieved. For example, the F-UJI assessment tool interpretation of FAIR with corresponding codes.

3.2.1 Work done by other projects and communities

Of a particular importance are the outcomes of the Research Data Alliance (RDA) FAIR Data Maturity Model Working Group [23] that produced a set of FAIRness indicators using a regular RDA process for community engagement. This set of indicators underpins the actual software tools developed by the FAIRsFAIR [24] and EOSC-Synergy projects [25]. It is worth noting that both projects end in 2022, so in a way DOME 4.0 is taking over from them regarding making progress with the actual implementation of FAIR measurement tools.

The most mature software tool originating in the mentioned projects is F-UJI Automated FAIR Data Assessment tool [21]. developed by FAIRsFAIR that implements most of the metrics developed by the mentioned RDA group. It should be kept in mind that RDA-developed metrics leave much room for their interpretation, and only by looking into the code one can really make sense of how a particular FAIR metric is interpreted.

Another tool called FAIR Evaluator [26] was developed by EOSC-Synergy along the same lines based on the RDA metrics, but it seems to be less mature than F-UJI and perhaps not worth reusing. Having this said, EOSC-Synergy tried to move one step further and looked into developing another set of FAIR metrics devoted to assessment of not a dataset but an entire data repository. This can be a promising line of work conceptually, and to some extent practically, as we may want to evaluate certain repositories from which DOME 4.0 is fetching data and then just "trust" them, i.e., automatically assign "default" FAIR metrics to datasets originating from a certain repository.

Furthermore, FAIRsharing.org [27] has developed the FAIRsharing FAIR Evaluation Services [28] that encompass various resources and guidelines to assess the FAIRness of digital assets. One of the most promising resources developed by FAIRsharing.org is the List of Maturity Indicators [29]. It contains some popular indicators and allows for registration of one's own FAIR maturity indicator. The newly registered indicator can then be referred to using a standard FAIRsharing citation mechanism including a DOI assigned to an indicator. This bears a good potential for defining FAIR metrics that reflect a specific notion of FAIRness well-fit with DOME 4.0 design and purposes, and further using these metrics in FAIR assessment tools and semantic assets such as ontologies.

3.3 Data models

A data model is an abstract model that provides a standardized method for defining and formatting data instances. Data models are a foundational element of software development and analytics. A data model is commonly made up of an identifier and some properties. These data model properties define the metadata for the "actual" data, the data model represents. The identifier is metadata for the data model.

Different ecosystems have traditionally been developed independently. As a result, there is a lot of variation in the way data is represented by different ecosystems, with many different data models describing the same or similar concepts in various ways. This makes the process of achieving smooth interoperability between data, devices, and software tools a costly and complex challenge.

A possible route towards solving this problem is that of ontologies and mappings.

3.4 Mappings

While data models represent data and ontologies represent the concepts, domain knowledge and the inherent semantics, a set of mappings is the tool to bind the two together.

A mapping is a semantic definition of how a specific data model property corresponds or maps to an ontological concept. A set of mappings for all of a data model's properties constitutes a complete ontological mapping of the data model. Note, the target ontological concepts can exist in the same or separate ontologies. Indeed, a single data model property can have multiple mappings, however, in this case it would not be expected that these mappings are targeting concepts within the same ontology, but it is not generally disallowed.

When all properties of a data model are mapped to ontological concepts, the data model is said to be a fully semantic data model. A set of fully semantic data model properties is essential for semantically- and data model-driven interoperability. Note the difference between a fully semantic data model and a set of fully semantic data model properties. The main point being that while a single data model may not be fully semantic, if the desired data properties from the given data model comprises a set of fully semantic data model properties, it will facilitate fully semantic interoperability. Otherwise, it will fall back to non-semantic data model-driven interoperability.

More than that, a set of mappings facilitates the properties of a single data model to be semantically described by several different ontologies simultaneously without issue (given the ontological concepts are not disjoint), facilitating data model reusability and easy semantical extendibility.

For further reusability, a set of mappings should be stored once created, in a similar fashion to how data models are stored, to ensure the same reusability of mappings as is the case for data models. It should be noted that some mappings only make sense in a specific context and should only be available within that context. This is e.g., the case of a generic deep learning model whose input and output may be mapped to possible disjoint ontological concepts depending on the context the deep learning model is applied to.

3.5 Separation of concern

An advantage of splitting mappings and data models in separate entities is that mappings depend on both the data model and an ontology. A data model depends only on a data provider's knowledge about their data, while an ontology depends only on the domain knowledge of a domain expert, as well as ontology experts. Separating data model and ontology with mappings means independent development, and that purely data-driven interoperability frameworks and technologies can be developed and implemented independently of the ontology development. The semantics can then be added as a layer on top of this when the more time-consuming development of ontologies is finishing, but it is not a strict requirement. An illustration of this separation can be seen in Figure 1.

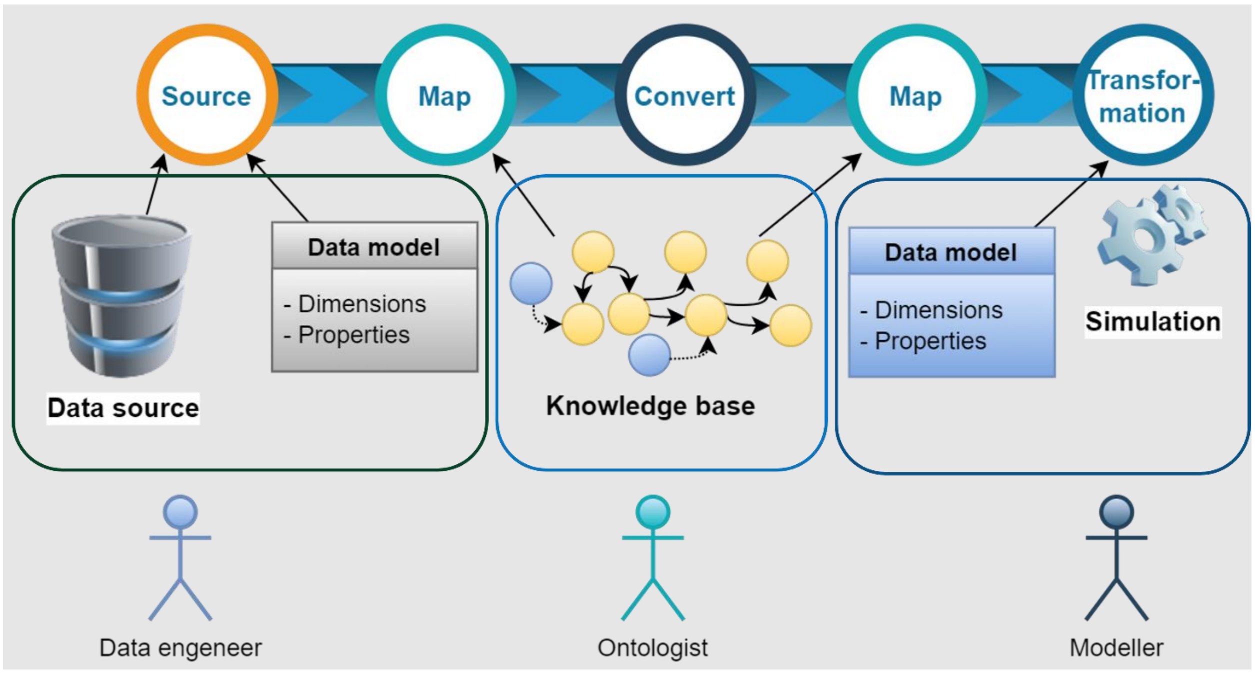

Figure 1: Illustration of separation of concern

4 Best practices for Software Development

Current best practices for software development do not always align with a project that has a team that is spread over several locations and organizations and must be planned in total before the start of the project, as is the case for DOME 4.0. This by design then excludes some of the best practices that are meant for a well-oiled (small) software team located in the same location. We will in this section focus on best practices that are relevant for DOME 4.0

4.1 Agile development

Even though the description of action assumes a classic waterfall approach to development, there is enough flexibility to consider an agile approach. An agile approach is an iterative development methodology, with short-term development goals building toward the final product. This short-term development cycle takes feedback from the previous cycles to improve the quality of the final product. As DOME 4.0 has partners working on different features in parallel, a waterfall model in the long term might produce incompatible features that cannot be integrated or cannot work together, this can be prevented if an agile strategy is used i.e., using iterative steps and develop the platform incrementally, so before we exhaust the project resources, we can solve incompatibilities. Hence an agile approach with shorter development cycles will increase the quality of the platform with constant feedback, as well as improve the collaboration between partners.

4.2 Microservices

The DOME 4.0 platform consists of many integrated parts, as can be seen in Section 5. The development team is also spread over several organizations. Conway's law states "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure” [30] This all means that it is best practice to break the development down into small pieces that can be developed as independently as possible. Microservices is an architectural style where the platform/ application is developed as a collection of small services, each service addressing a single feature. This enables different services to evolve at different rates and be updated without affecting the other services. DOME 4.0 has multiple partners in different locations working on different features, in different programming languages. If these are developed as separate services that constitute the platform i.e., creating different functionalities as microservices, then this will make it easier to integrate, maintain and debug the platform in the long-term evolution of the project.

4.3 Source Control Management

DOME 4.0 is to a large extent a software project and we will have multiple developers/partners contributing to the development process, and various features being developed over time. Hence it is important to store the history of changes, who made them, and when so we can revert or look back to previous versions and see how the project has evolved. This tracking is essential to maintain the integrity of the code base. Also, in case of any issues or bugs those errors can be figured out easily by comparing them to the last working version. And if required these errors can be reverted. Version control also helps developers collaborate on a project easily, as everyone has access to the previous versions, and they can work simultaneously on the same project from various locations.

Some of the best practices to keep in mind are A) Writing good commit messages to understand what the changes are and why it was changed. B) Test before committing. C) Get the code reviewed before merging changes.

The most recommended and popular version control tool is Git. Git is free and open source, and is supported by most providers offering infrastructure for code management.

4.4 Project Management Tool

The main function of a project management tool is to help plan, organize, manage tasks and estimate timelines. This is very important for a project where multiple people are involved and the tasks, milestones, and deadlines are spread across the years. As an initial planning tool, the Gantt chart was used. But when the development process starts, we want a tool that can track the progress of tasks and subtasks included in each milestone, hence as a standard, we can use a Kanban board that can visualize the work and progress and the people associated with different tasks. There are many digital Kanban boards available to choose from (for example GitHub projects and Jira Kanban boards). Popular providers of integrated tools for software development, like GitHub [31], GitLab [32] and Bitbucket [33] all have tools where the project management tool is connected to the source control management and the CI/CD.

4.5 Continuous Integration/Continuous Delivery (CI/CD)

This is the combination of continuous integration (CI) and continuous delivery or continuous deployment (CD). Continuous integration refers to how fast the developed features are integrated into the source code, and continuous deployment automatically deploys these new features to the production system. Lastly, continuous delivery means new versions of software can be released anytime the latest version is deployed.

This is very important for a project like DOME 4.0 as new deliverables and tasks can be added incrementally to the platform, and this enables all the partners and platform users to see how the platform is evolving. Automated tests to all the repositories will ensure that coding standards are maintained by all developers and continuous deployment will allow partners to test and give feedback.

5 Software

This section lists software relevant to DOME 4.0 in different tasks required for the implementation of the DOME 4.0 platform. An overall diagram showing the different categories can be seen in Figure 2. The selection of software must be seen as a non-exhaustive list and should be read as a suggestion for best practices based on our experiences in other relevant ongoing and finished projects where we were involved, like NanoSim [1][34], SimPhoNy [35] [2], VIMMP [36] [3], MarketPlace [37] [4], SimDome [38] [5], OntoTrans [6][39], OpenModel [40] [7], and VIPCOAT [41] [8]. The OntoCommons project [2] did an extensive landscape analysis of ontological engineering tools [42], only what we think is the most relevant selection is mentioned below.

The content of this section is in its nature quite technical and is intended as a help for developers in DOME 4.0 to choose the right tools. Consequently, it might be hard to read for the casual reader.

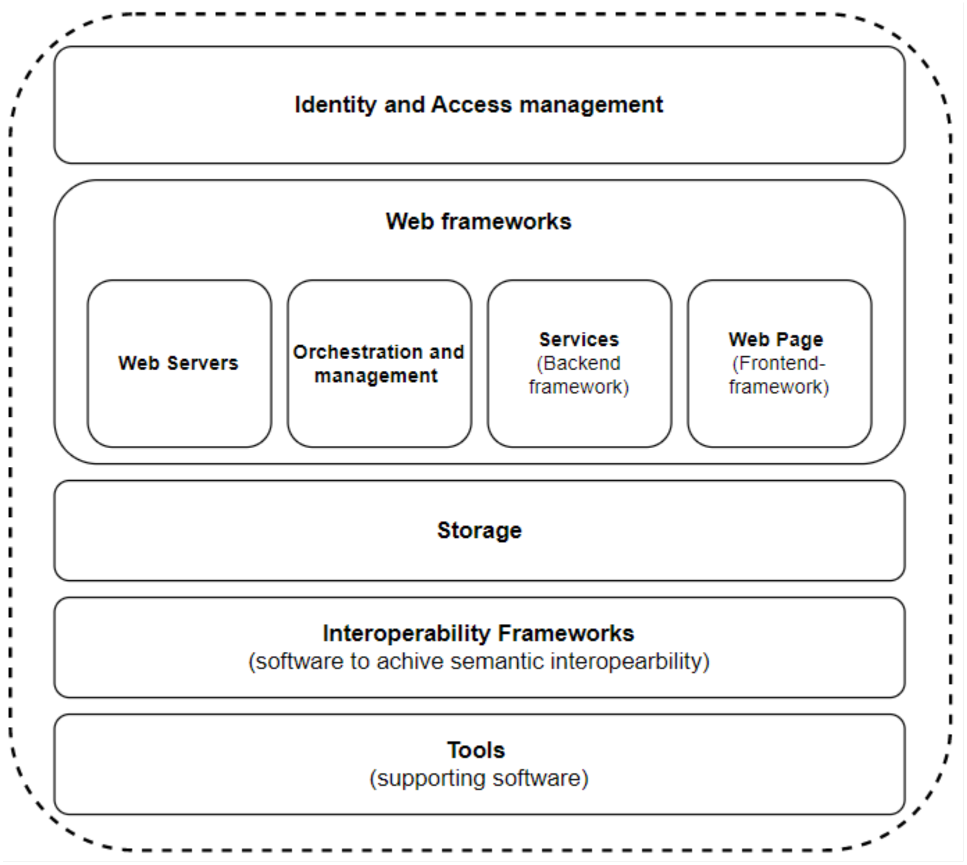
Identity and Access management
Web frameworks
Web Servers
Orchestration and management
Services (Backend framework)
Web Page (Frontend-framework)
Storage
Interoperability Frameworks (software to achive semantic interopearbility)
Tools (supporting software)

Figure 2: Software components for DOME 4.0 platform development

5.1 Identity and Access management

Authentication, authorization, and access control are very important factors to achieve cyber security. Here authentication is the process of verifying the identity of the users, authorization is the process of checking if a user is allowed to access a specific resource. Access control makes sure that the user doesn't access any resources that shouldn't be accessed. For this, we can use pre-existing access control solutions with features required for DOME 4.0 (see Table 2 and Table 3).

Table 2 Keycloack overview

Name	**KeyCloak**
Description	KeyCloak is an open-source identity and access management solution.
Rationale	The DOME 4.0 web platform will require a way to manage authorization and authentication of users. Used by MarketPlace [37], OntoTrans [39], VIPCOAT [41]
Cons	• No out of the box solution is available in KeyCloak to generate API keys (with specific scopes, expiry etc.).
Pros	• A range of options are available out of the box. • It can be integrated with existing user systems without the need for migrating these data.

	• Single sign-on: users log in with a single ID and password and gain access to all connected systems. • Meets a range of security standards. • Can save time in development by offering authentication, authorization, and single sign-on all in one. • Offers an admin console (web-based GUI) that is easy to use.
References	• https://www.keycloak.org/

Table 3 ORY Network overview

Name	**ORY Network**
Description	The Ory Network is an access management system that provides full-scale authentication, authorization, federation, and user management for mobile apps, web apps, and any cloud service.
Rationale	The DOME 4.0 web platform will require a way to manage authorization and authentication of users.
Cons	• Smaller community compared to KeyCloak. • Less complete compared to KeyCloak.
Pros	• Lightweight.
References	• https://www.ory.sh/

5.2 Web Frameworks

Web frameworks support the development of web applications (web services, web APIs and web resources). They also provide a standard way to build and deploy web applications and help to automate some processes during development.

5.2.1 Web Servers

Web servers are used to serve web content. The web server stores, processes, and delivers web content to users (via web browsers). It refers to both software and hardware. Table 4 and Table 5 provide an overview of available software.

Table 4 NGINX overview

Name	**NGINX**
Description	Nginx is an open-source, high-performance HTTP server and reverse proxy. It is commonly used for load balancing, web acceleration, and security and anonymity. Used by MarketPlace [37], OntoTrans [39], VIPCOAT [41]
Rationale	The DOME 4.0 web platform needs to ensure smooth flow of network traffic between clients and servers. It also requires load balancing features. Used in VIPCOAT [41] and MarketPlace [37].
Cons	• Less extensive list of features and functionality compared to Apache. • Less community support compared to Apache.

Pros	• Lightweight: requires relatively few resources and memory. • Provides load balancing.
References	• https://docs.nginx.com/

Table 5 Apache HTTP server overview

Name	**Apache**
Description	The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.
Rationale	The DOME 4.0 web platform needs to be stable, fast, and secure. This can be achieved using an Apache Server for the deployment.
Cons	• Consumes more ram under heavier load than nginx. • Spawn new processes for each request making things less efficient.
Pros	• Provides an admin console. • Greater selection of features compared to nginx.
References	• https://httpd.apache.org/

5.2.2 Orchestration and Management

Container orchestration and management helps automate creation, deployment, management, scaling and networking of containers. This helps manage containers in microservice architecture at scale. Table 6, Table 7, Table 8, and Table 9 provide an overview of commonly used software.

Table 6 Kubernetes overview

Name	**Kubernetes**
Description	Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
Rationale	System to manage and deploy application services; it groups containers that make up an application into logical units for easy management and discovery.
Cons	Not every organization has multiple containers to deploy, nor need for high availability. Can be complex to get running, with steep learning curve for starters, but can be worth-while for organization with sufficient resources with the right profile, who can dedicate sufficient time.
Pros	Strong community with many years of experience and track record. Provides advantages on-premise, hybrid or public cloud infrastructure, flexible to scale without increasing the operations team. Powerful capabilities.
References	• https://kubernetes.io/

Table 7 Docker overview

Name	**Docker**

Description	Docker is OS-level virtualization platform that provides software to run in deployable containers which are self-contained with all required dependencies. This allows for a more effective way of deploying software applications. Used by MarketPlace [37], OntoTrans [39], VIPCOAT [41]
Rationale	With the numerous services that run in the DOME 4.0 framework, the trouble of installing each element on different hosts are avoided by using containers that any cloud service or on-site mainframes can run.
Cons	• Has gaps in documentation, especially for Mac. • Not good for monitoring (for monitoring we need to use docker desktop which is not free).
Pros	• Makes deployment easier and more effective. • Allows for isolating environments for easier debugging • Comes with a comprehensive container registry with thousands of container images.
References	• https://docs.docker.com/ • https://cloudinfrastructureservices.co.uk/ansible-vs-docker-whats-the-difference-between-devops-tool/

Table 8 Ansible overview

Name	**Ansible**
Description	Ansible is an open-source automation tool from Red Hat that automates provisioning, configuration management, application deployment, orchestration, and many other manual IT processes.
Rationale	The DOME 4.0 web platform might benefit from automating some tasks.
Cons	• It does not have a good user interface. • Requires complex data structures for network automation tasks. • It has limited Windows support.
Pros	• Provides step-by-step reporting that provides information on task success and failure. • Simplifies CI/CD (continuous integration and deployment) processes, thus reducing human errors.
References	• https://www.ansible.com/ • https://cloudinfrastructureservices.co.uk/ansible-vs-docker-whats-the-difference-between-devops-tool/

Table 9 Docker Compose overview

Name	**Docker Compose**
Description	Managing several different docker containers can be a tedious task. Docker Compose is a tool that helps in overcoming this problem, and that allows us to easily handle multiple containers at once. Used by MarketPlace [37], OntoTrans [39], VIPCOAT [41]
Rationale	The DOME 4.0 web platform will consist of several different services. Using Docker in each of these services and tying them all together using Docker Compose can simplify both development and deployment.

Cons	• A container cannot be replaced without downtime (con for production in particular). • There are no health checks available in production with Docker Compose: docker run and docker-compose won't re-create containers that failed a built-in health check.
Pros	• Fast and simple configuration with YAML scripts. • It allows for hosting multiple isolated environments on a single host, leading to an efficient use of resources. • Portability: All the services are defined inside a docker-compose file, and the entire configuration can easily be accessed and shared.
References	• https://docs.docker.com/compose/

5.2.3 Back-end Frameworks

Backend web frameworks are tools that help in the development of server-side software (all the APIs, architecture, connecting to databases, and functionalities that happen behind the scenes) This mainly focuses on server-side scripting languages like Python, JavaScript, and PHP. Concerning specifically Python, we should use a Python version that is currently supported, which currently means a minimum of Python 3.7 for all Python-based packages developed within the project [43]. Similar considerations should be done when using other languages. A detailed overview of tools used in the project is presented in Table 10, Table 11, Table 12, and Table 13.

Table 10 Flask overivew

Name	**Flask**
Description	Flask is a free and open-source micro web framework for developing web applications and services written in Python. It is classified as micro web framework as it is light weight and provides only necessary components
Rationale	The DOME 4.0 project needs a fast, user-friendly web platform for registering, searching, and accessing data. Using a microframework such as Flask simplifies the development of this platform. Used by MarketPlace [37], VIPCOAT [41]
Cons	• Not a lot of tools: Lacks a large toolbox which means that developers might have to manually add extensions such as libraries. Adding many (third-party) extensions can cause the app to slow down and might also pose a security risk. • API documentation documents must be created manually.
Pros	• Flexible and minimalistic: almost all the parts of flask are open to change. • Lightweight: there are few constituent parts need to be assembled and reassembled, and it does not rely on many extensions to function. • Supports modular programming. • Well documented.
References	• https://pythonbasics.org/what-is-flask-python/

Table 11 FastAPI overview

Name	**FastAPI**

Description	FastAPI is a modern, fast (high-performance), web framework for building APIs based on standard Python type hints. Used by MarketPlace [37]
Rationale	The DOME 4.0 web platform needs a set of APIs for the different services to communicate with each other and for DOME 4.0 to communicate with external data sources and data sinks. Using a web framework such as FastAPI simplifies the development of these APIs. Used in Marketplace [37].
Cons	• Small community of developers compared to flask • Only compatible with versions Python 3.7+
Pros	• Fast: high performance, on par with NODEJS and Go • Fast to code - saves time in development of APIs. • Based on, and fully compatible with, the open standards for APIs: OpenAPI (previously known as Swagger) and JSON Schema. • Easy testing. • Well documented.
References	• https://fastapi.tiangolo.com/

Table 12 Django overview

Name	**Django**
Description	Django is a free and open-source, Python-based web framework for developing web applications and services.
Rationale	The DOME 4.0 project needs a fast, user-friendly web platform for registering, searching for, and getting access to data. Using a microframework such as Django simplifies the development of this platform.
Cons	• Has more complicated features than Flask, which can lead to a larger on-boarding/learning process.
Pros	• Saves time in development, as the whole idea of Django is to build and scale up projects incredibly quickly. • Large community and many learning resources. • Well documented. • Comes with security features such as a user authentication system already set up.
References	• https://www.djangoproject.com/

Table 13 PHP overview

Name	**PHP**
Description	PHP stands for Hypertext Pre-processor and is an open-source general-purpose scripting language. It can be embedded into HTML and is suited for web development.
Rationale	The DOME 4.0 project needs a fast, user-friendly web platform for registering, searching for, and accessing data. Using PHP could simplify the development of this platform.
Cons	• PHP has a weak type, which may lead to incorrect data and unexpected bugs.

	• Differences between older and updated versions can introduce errors in scripts.
Pros	• Simple (small barrier for developers), but still offers many advanced features. • It is platform-independent: PHP-based applications can run on any OS. • Well-documented.
References	• https://www.php.net/

5.2.4 Front-end Frameworks

Frontend web frameworks provide packages and libraries for GUI (graphical user interface) development. This usually provides reusable widgets and code snippets to manage user interaction with the web application. Commonly used frameworks are presented in Table 14, Table 15, Table 16, Table 17.

Table 14 AngularJS / Angular overview

Name	**AngularJS / Angular**
Description	Angular is a development platform, built on TypeScript. As a platform, Angular includes a component-based framework for building scalable web applications, a collection of libraries covering a wide variety of features, and a suite of developer tools. Used by MarketPlace [37], VIPCOAT [41].
Rationale	The DOME 4.0 web platform needs a clean and user-friendly front-end. Using a front-end development framework will simplify and speed up the development. Used in VIPCOAT [41].
Cons	• Angular is complex and verbose, so it's not suitable for small apps. It's designed for complex systems. • It has a steep learning curve, and it may take some time to master it. • Angular is a SPA (single page application), so it has limited SEO (search engine optimization) capabilities.
Pros	• Angular sites are SPAs. SPAs allow us to load new pages without requesting the server, giving a more dynamic and interactive experience. • It has a lot of features built in. • It is cross-platform.
References	• https://angular.io/docs

Table 15 React overview

Name	**React**
Description	React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.
Rationale	The DOME 4.0 web platform needs a clean and user-friendly front-end. Using a front-end development framework will simplify and speed up the development.
Cons	• Not well documented.

	• Uses JSX (syntax extension that allowing for mixing of HTML and JavaScript). This has its benefits but can be a barrier for developers.
Pros	• Large community. • Backward compatibility (in particular, seldom large changes to API).
References	• https://reactjs.org/

Table 16 Node.js overview

Name	**Node.js**
Description	Node.js is an open-source, cross-platform runtime environment for server-side and networking applications. It can be used in both front- and back-end.
Rationale	The DOME 4.0 web platform needs a clean and user-friendly front-end. Using a front-end development framework will simplify and speed up the development.
Cons	• Unstable APIs – migrating code from an older version of node.js to a new one might be cumbersome since the APIs keep changing. • Lacks a strong library support system, which means that developers might have to rely on third-party libraries.
Pros	• Large community. • Offers easy scalability due to its asynchronous and event-driven nature. This makes Node.js suited for high throughput and real-time applications. • High performance compared to other server-side scripting languages (such as Java or PHP). • It is cross-platform.
References	• https://nodejs.org/en/about/

Table 17 Bootstrap overview

Name	**Bootstrap**
Description	Bootstrap is an open-source front-end web development framework, based on HTML5, CSS, and JavaScript. Bootstrap makes it possible for developers to quickly launch a fully featured, mobile-responsive site. Used by MarketPlace [37], VIPCOAT [41].
Rationale	The DOME 4.0 web platform needs a clean and user-friendly front-end. Using a front-end development framework will simplify and speed up the development. Used in VIPCOAT [41] and MarketPlace [37].
Cons	• Its naming scheme can be confusing. • Everything built with Bootstrap will have similar looks (one might be able to override and modify style sheets manually, but this can be cumbersome).
Pros	• It is responsive, meaning that the design will automatically resize to suit the page (important since many people will browse the web from a mobile phone or tablet). • It includes major components such as dropdowns and navigation bars. • No CSS knowledge needed.
References	• https://getbootstrap.com/docs/4.1/about/overview/

5.3 Storage

5.3.1 Standard databases

Databases are used to store and retrieve data. There are many types of databases. What type of database is used by a particular web platform depends on how and what kind of data will be used in that platform. Table 18, Table 19, and Table 20 presented commonly used databases.

Table 18 PostgreSQL overview

Name	**PostgreSQL**
Description	PostgreSQL is an open-source database management system. It supports both SQL and JSON for relational and non-relational queries. Used by MarketPlace [37], VIPCOAT [41].
Rationale	The DOME 4.0 platform may need to store (meta)data.
Cons	• Open-source apps that support MySQL may not support PostgreSQL • Slower than MySQL on performance metrics.
Pros	• It is highly extendible – if you need an additional feature, you can add it yourself. • Low maintenance and administration for both embedded and enterprise use of PostgreSQL.
References	• https://www.postgresql.org/

Table 19 MongoDB overview

Name	**MongoDB**
Description	MongoDB is a free to use document database. It stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. Used by VIPCOAT [41].
Rationale	The DOME 4.0 platform may need to store (meta)data. Used in VIPCOAT.
Cons	• Less flexibility with querying than a relational database (e.g., JOINs are not supported)
Pros	• Scalability: MongoDB supports horizontal scaling through sharding. • Flexibility in that it is schema-less.
References	• https://www.mongodb.com/what-is-mongodb

Table 20 Redis overview

Name	**Redis**
Description	Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, message broker, and streaming engine. Used by OntoTrans[39]
Rationale	The DOME 4.0 platform may need to store (meta)data. Used by OntoTrans.
Cons	• It does not support a query language, so there is no case of using ad-hoc queries. Data access paths must be designed, which results in a loss of flexibility.

Pros	• Speed: It is one of the fastest caching technologies on the market. • Easy setup • It has flexible data structures – supports almost all data structures.
References	• https://redis.io/docs/about/

5.3.2 Triplestore

Triplestore is a database that stores and retrieves them through semantic queries. Triplestore stores data as triples. A triple is a data entity composed of subject-object-predicate. Table 21 and Table 22 present commonly used triple stores.

Table 21 AllegroGraph overview

Name	**AllegroGraph**
Description	High performance, persistent RDF store with additional support for Graph DBMS (database management system). Commercial, but has a free limited community edition. Used by MarketPlace [37], OntoTrans [39].
Rationale	DOME 4.0 aims to facilitate maximum knowledge extraction with the help of ontology-driven semantic data interoperability and modern data processing technologies. In this context, the DOME 4.0 platform needs to store RDF triples, and the triples must be searchable through queries. This can be achieved using a triplestore such as AllegroGraph. Used in MarketPlace[37] as a demo.
Cons	• One of AllegroGraph limitations is it focuses on geo-temporal reasoning and social network analysis.
Pros	• Has efficient use of memory by combining disk storage, making it possible to scale up to one billion nodes, always maintaining top performance. • Suited to support ad-hoc queries through SPARQL, Prolog and languages like JavaScript. • Triple level security with security filters. • SOLR and MongoDB integration. • All clients are based on REST protocol.
References	• https://dl.acm.org/doi/10.5220/0006910203730380 • https://allegrograph.com/

Table 22 Stardog overview

Name	**Stardog**
Description	Enterprise Knowledge Graph platform and graph DBMS with high availability, high performance reasoning, and virtualization. Commercial, but has a 60-day fully featured trial license and a 1-year fully featured non-commercial use license for academics/students.
Rationale	DOME 4.0 aims to facilitate maximum knowledge extraction with the help of ontology-driven semantic data interoperability and modern data processing technologies. In this context, the DOME 4.0 platform needs to store RDF triples, and the triples must be searchable through queries. This can be achieved using a graph database such as Stardog. Used in OntoTrans [39].

Cons	• Relatively small community.
Pros	• Built-in machine learning • Based on open standards
References	• https://www.stardog.com/

5.4 Interoperability Framework

Interoperability frameworks provide a set of tools and concepts to describe data and enable data to be reused. Commonly used software is presented in Table 23 and Table 24.

Table 23 DLite (SOFT) overview

Name	**DLite (SOFT)**
Description	DLite is a lightweight interoperability framework, for working with and sharing scientific data. DLite is a C implementation of the SINTEF Open Framework and Tools (SOFT), which is a set of concepts and tools for how to efficiently describe and work with scientific data. Used by OntoTrans [39], VIPCOAT [41].
Rationale	DOME 4.0 aims to facilitate maximum knowledge extraction with the help of ontology-driven semantic data interoperability and modern data processing technologies. For this, an interoperability framework is needed.
Cons	• May feel little abstract the first time the user is confronted with DLite.
Pros	• Simplistic data models that are close to the data source/sink providing easy onboarding. • Available as a python package • Full semantics is obtained via mappings to ontological concepts. • Via ontological mappings, DLite allows to transparently convert and represent the data as the form the user needs. • Separation of concerns: Data models and ontologies are created separately (i.e., ontologies are not required as inputs). • Not bound to a specific ontological framework. Enables cross-domain interoperability. • Fast and efficient exchange of data between different platforms.
References	• https://github.com/SINTEF/dlite

Table 24 CUDS/CUBA overview

Name	**CUDS/CUBA**
Description	CUDS (Common Universal Data Structure) is a data structure that is used to uniformly represent ontology concepts in programming code. The CUDS object is an ontology individual that can be used like a container. It has attributes and is connected to other cuds objects via relationships. It is used to facilitate semantic interoperability within the SimPhoNy framework. Used by MarketPlace [37], SimPhoNy [35], and SimDome [38].
Rationale	DOME 4.0 aims to facilitate maximum knowledge extraction with the help of ontology-driven semantic data interoperability and modern data processing technologies. For this, an interoperability framework is needed.
Cons	• Requires ontologies as inputs.

Pros	• Allows for a semantic demonstration of the data with more specialised domain knowledge. • Allow a faster search of the relevant search string. • Standardise the format of the data presented in DOME 4.0. • Reducing the computing cost and other costs associated with human intervention. • Allowing fast and efficient data exchange among other platforms and services.
References	• https://simphony.readthedocs.io/en/latest/overview.html • https://simphony.readthedocs.io/en/latest/jupyter/cuds_api.html

5.5 Tools

The tools in Table 25, Table 26, and Table 27 might be useful for different tasks in DOME 4.0 like developing reference connectors and ontologies.

Table 25 Cookiecutter overview

Name	**Cookiecutter**
Description	A command-line utility that creates projects from cookiecutters (project templates), e.g., creating a Python package project from a Python package project template. Used by OntoTrans [39].
Rationale	The DOME 4.0 web platform will consist of several different services. Using a scaffolding generator such as cookiecutter can save time in development and ensure consistency across the services by defining a project structure. Used in OntoTrans[39].
Cons	• Relatively small community.
Pros	• Cross-platform • Project templates can be in any programming language or markup format: Python, JavaScript, Ruby, CoffeeScript, RST, Markdown, CSS, HTML, etc. • Well-documented. • You can use multiple languages in the same project template.
References	• https://www.cookiecutter.io/

Table 26 Protégé overview

Name	**Protégé**
Description	Protégé is an open-source ontology editor and a knowledge management system.
Rationale	The DOME 4.0 web platform will be using ontologies for its functionalities and protégé will be useful in developing it.
Cons	• Some plugins for reasoning are difficult to install
Pros	• Support for W3C standards • Active community (both academic, government and corporate)
References	• https://protege.stanford.edu/

Table 27 EMMOntoPy overview

Name	**EMMOntoPy**
Description	It is a python API to work with ontology. Developed to work with EMMO based ontologies. It is an extension of owlready2.
Rationale	The DOME 4.0 web platform will be using ontologies for its functionalities, and this might support developing it.
Cons	• Command line-based coding tool no GUI.
Pros	• Can generate an ontology by filling out an excel template to improve interaction between domain experts and otologists. • Work with ontologies using python (Make, generate, extend, visualize)
References	• https://github.com/emmo-repo/EMMOntoPy • https://emmo-repo.github.io/

6 Technologies

As opposed to the specific software described in the previous section, this section describes some of the technologies that are the best practices and standards for implementing authentication, authorization, communication interfaces, API specification standard and persistence. This is all required for the DOME 4.0 platform. The required technologies for DOME 4.0 are shown in Figure 3.

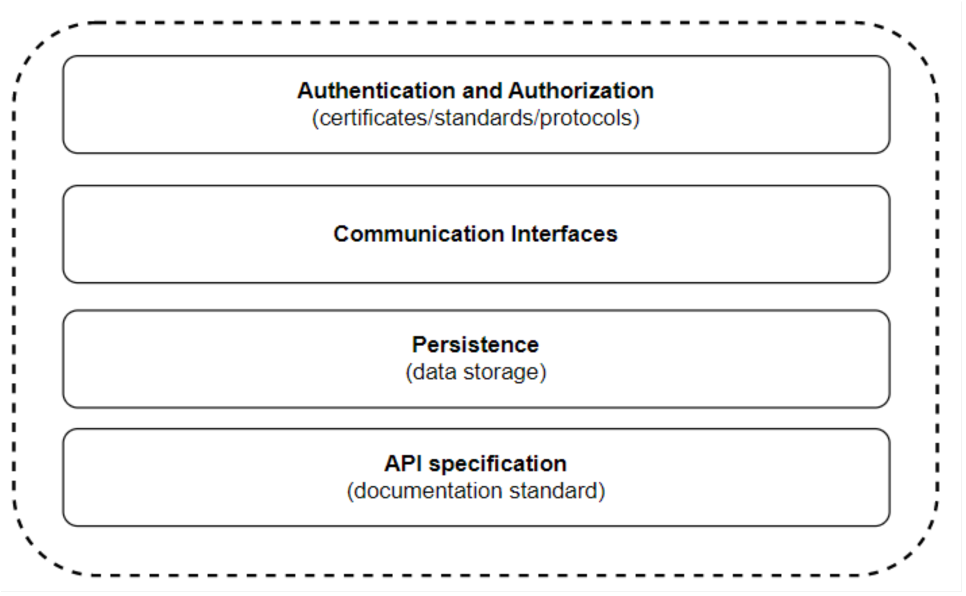

Figure 3: Required Technologies

6.1 Authentication and Authorization

This is a very important part of any web application. We need certificates and other protocols to make sure a web application is not a malicious one or something that potentially harms our web server if interacted with. We need to make sure the identity is verified, and communication is secured. Table 28 and Table 29 present the relevant technologies.

Table 28 X.509 (certificate) overview

Name	**X.509 (certificate)**
Description	X.509 is a standard format for public key certificates. This can be used to verify the identity of a web server and secure communications.
Rationale	Must have for connecting with IDS platforms. And any other secure platforms.
Cons	• Requires centralized administration and chain of trust.
Pros	• Proven technology. • Basis for https.
References	• RFC5280: https://www.rfc-editor.org/rfc/rfc5280

Table 29 OAuth2 overview

Name	**OAuth2**
Description	OAuth2 is a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user.
Rationale	Must have for connecting with other platforms.
Cons	• There is no common format, as a result, each service requires its own implementation. • In the process of user verification, sometimes additional requests must be made to get minimal user information. This can be solved using JWT token, but not all services support it.
Pros	• Well documented. • Can be used on almost any platform.
References	• https://oauth.net/2/

6.2 Communication interfaces

Communication interfaces enable different software, databases, and networks to communicate with each other. Here we are focusing on the interfaces that can enable internal communication between different components and for the DOME 4.0 platform to interact with external systems. Table 30 and Table 31 present the relevant technologies.

Table 30 RESTful API overview

Name	**RESTful API**
Description	A REST (or RESTful) API is an API that conforms to the constraints of the REST architectural style and allows for interaction with RESTful web services. REST is short for representational state transfer. Used by MarketPlace [37], OntoTrans [39], VIPCOAT [41].
Rationale	The DOME 4.0 web platform needs a set of APIs for the different services to communicate with each other and for DOME 4.0 to communicate with external data sources and data sinks.
Cons	• It is not suitable for passing confidential data between client and server since it does not impose security.
Pros	• It provides a lot of flexibility • REST APIs are the most used APIs for web services.

	• It uses standard HTTP procedure callouts to retrieve data and requests. • It allows for standard-based protection using OAuth protocols to verify the REST requests.
References	• https://www.ibm.com/cloud/learn/rest-apis

Table 31 GraphQL overview

Name	**GraphQL**
Description	GraphQL is an open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data
Rationale	The DOME 4.0 web platform needs to communicate with different services and with external data sources and data sinks. And between its internal services.
Cons	• Difficult to implement cache • Increased query complexity
Pros	• Fast development • Fetches data based on query. No predefined schema like REST.
References	• https://graphql.org/

6.3 API Specification

API specification/ documentation is necessary to understand the capabilities of a service. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic (see Table 32).

Table 32 OpenAPI overview

Name	**OpenAPI 3**
Description	The OpenAPI Specification (OAS) defines a standard interface to REST APIs. This interface is language-agnostic and allows both humans and computers to discover and understand the service's capabilities without access to source code, documentation, or through network traffic inspection. Used by MarketPlace[37], OntoTrans[39], VIPCOAT[41]
Rationale	The backend of the DOME 4.0 web platform needs a set of APIs for the different services to communicate with each other and for DOME 4.0 to communicate with external data sources and data sinks.
Cons	• The fact that OpenAPI is language-agnostic has its benefits, but also introduces a lot of complexity in the incorporation of language-specific frameworks, third party APIs, and server-based extensions.
Pros	• Stable implementation. • Large community. • Large ecosystem of tools (created under the Swagger brand) that can be used to speed up the API development process.
References	• https://swagger.io/specification/

6.4 Persistence

Data is an important part of any web platform. Different kinds of storage technologies are required to deal with this data. What storage technology to choose will vary with different requirements of different web platforms. Relevant technologies are presented in Table 33, Table 34, and Table 35.

Table 33 Triplestore overview

Name	**Triplestore**
Description	The triplestore is a database that stores RDF triples. It can handle semantic queries and use inference for uncovering new information out of the existing relations.
Rationale	DOME 4.0 aims to facilitate maximum knowledge extraction with the help of ontology-driven semantic data interoperability and modern data processing technologies. In this context, the DOME 4.0 platform needs to store RDF triples, and the triples must be searchable through queries.
Cons	• Difficult to store data that can't be defined as triples (like documents)
Pros	• More flexible than a relational database. • Will typically be less costly than a relational database. • Facilitates easy sharing of data with URIs. • Easy import and export of data.
References	• https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/

Table 34 "Standard" database (relational/NoSQL) overview

Name	**"Standard" database (relational/NoSQL)**
Description	A relational database (RDB) is a collection of data items structured as tables, rows, and columns. An RDB can establish relationships between these items by joining tables, which makes it easy to understand and gain insights about the relationship between various data points. NoSQL databases are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.
Rationale	The DOME 4.0 platform may need to store (meta)data.
Cons	• RBD is costly to set up and maintain. • RBD requires a lot of physical memory. • Less flexibility with querying in NoSQL
Pros	• An RDB is strictly defined and well-organized, which prevents duplication of data. • A NoSQL can store both structured and semi-structured data • Collaboration: Multiple users can access the DB to retrieve information at the same time, even when data is being updated.
References	• https://cloud.google.com/learn/what-is-a-relational-database • https://www.mongodb.com/nosql-explained

Table 35 Cache storage (key/value store) overview

Name	**Cache storage (key/value store)**
Description	A cache is a hardware or software component that stores data with the purpose of serving future requests for that data faster.
Rationale	The DOME 4.0 web platform needs to be stable, fast, and secure. A cache storage can help improve both the speed and the stability.
Cons	• Risk of presenting users with stale data, that is, outdated content. • Lowering the risk of stale data typically introduces a lot of complexity, making the code more difficult to maintain.
Pros	• Faster loading times for websites and applications. • Reduces the load on back-end databases. • Accelerates data retrieval (faster queries).
References	• https://www.techtarget.com/searchstorage/definition/cache

7 Chemical Substance Identifiers

Searching for chemical substances is not as straight forward as it might seem. To alleviate this, there are several ways to represent chemical substances in a systematic way. We mention some of them relevant for DOME 4.0 in the next sections, but the list of relevant identifiers may be extended in the future.

7.1 IUPAC Naming Rules

The International Union of Pure and Applied Chemistry (IUPAC) provides a set of recommendations on how to name chemical substances in a systematic, unambiguous manner. These recommendations are listed in the red book and the blue book for inorganic [44] and organic chemistry [45] respectively. The advantage with these naming recommendations is that it is possible to recognize, without a doubt, what chemical substance the user is looking for. The disadvantage is that the rules are quite complex and might be hard to parse for a computer. IUPAC naming rules are supported by some data sources relevant for DOME 4.0. For instance Cheméo [46], PubChem [47].

7.2 SMILES

The Simplified Molecular-Input Line-Entry System (SMILES) is a way to textually represent the structure of chemical substances [48]. The advantage of SMILES is that they are to a certain degree easy to read (complex structures are complex regardless of the representation). A thing to note is that it is possible to construct several SMILES that will lead to the same chemical substance. It is also possible to create SMILES that are not ambiguous. Chirality might for instance not be taken into account (but it can). SMILES are supported by some data sources relevant for DOME 4.0. For instance Cheméo [46], PubChem [47].

7.3 CAS Registry Number

CAS registry numbers [49] are generated for every chemical substance in the open scientific literature by the American Chemical Society. The big advantage with the CAS registry number is that every known chemical substance will have one, but there is no way to know what chemical substance a CAS registry number represents without looking it up somewhere. This means the CAS registry number cannot be used as the only way to recognize a chemical substance in practical applications. CAS registry numbers are supported by some data sources relevant for DOME 4.0. For instance Cheméo [46], PubChem [47].

7.4 EC number

The EC number is a unique 7 digit identifier assigned to each substance regulated by the European Union. This number is currently not in use by any of the sources relevant to the DOME 4.0 platform, but should not be disregarded in the future. It only carries limited information and as with the CAS number the chemical must be looked up [50].

7.5 InChI/InChIKey

Similar to SMILES, The International Chemical Identifier (InChI) is another textual identifier for chemical substances proposed by IUPAC [51]. The InChI is generated by an algorithm taking the chemical structure as the input. The InChI carries chemical information, is human readable and is guaranteed to be unique. The InChI can then be seen as a compromise between CAS registry numbers and IUPAC naming rules. It does require an algorithm to generate the InChI, but the InChI is well suited as an input to search data sources for a given chemical substance.

InChIKey is a hashed version of InChI consisting of 27 characters. This can be seen as a no-human readable short version of InChI. A given InChI will always give the same InChIKey, but the only way to go the other way is to look up the InChIKey in a database.

InChIKeys are supported by some data sources relevant for DOME 4.0. For instance PubChem [47].

8 Conclusions / Next steps

This report summarizes the standards and best practices collected for the development of the DOME 4.0 ecosystem. A lot of the recommendation from this report has already been considered and is being used in the core work packages.

Even though this report is written, the work to make sure the DOME 4.0 is kept up to date with the current standards and best practices will continue. It makes sense to do another round of contact with relevant projects, as these all have progressed further since the previous round. As the development of the core platform progress, even more of the standards and best practices listed in this report will become useful.

9 Lessons learnt and deviations from Annex 1

While working on this deliverable we have realized that the task of collecting relevant standards and best practices for a project such as DOME 4.0 is a complex and open-ended task. To alleviate this challenge, we focused on what we thought would be most relevant to the implementation of the DOME 4.0 ecosystem. We had contact with a lot of projects and initiatives (through Task 5.1-5.3), but with limited time for collaboration and several topics to cover, we might have missed some things, and further collaboration will be sought. This was a lot easier for the projects were some of the DOME 4.0 partners were also partners in other projects. We then had a direct line of communication.

There are no known deviations from Annex 1 in the description of action.

10 References

[1] M. D. Wilkinson *et al.*, ‘The FAIR Guiding Principles for scientific data management and stewardship’, *Sci. Data*, vol. 3, no. 1, Art. no. 1, Mar. 2016, doi: 10.1038/sdata.2016.18.

[2] ‘OntoCommons’, *(2020-2023) received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement no. 958371.*

[3] L. A. Slaughter and J. Otten, ‘OntoCommons D2.2 - TLOMLO Landscape Analysis Report’, Jan. 2022, doi: 10.5281/zenodo.6504440.

[4] ‘Basic Formal Ontology (BFO) | Home’. https://basic-formal-ontology.org/ (accessed Nov. 11, 2022).

[5] R. Arp, B. Smith, and A. D. Spear, ‘Building Ontologies with Basic Formal Ontology’, *MIT Press*. https://mitpress.mit.edu/9780262527811/building-ontologies-with-basic-formal-ontology/ (accessed Nov. 11, 2022).

[6] S. Borgo and C. Masolo, ‘Ontological Foundations of dolce’, in *Theory and Applications of Ontology: Computer Applications*, R. Poli, M. Healy, and A. Kameas, Eds. Dordrecht: Springer Netherlands, 2010, pp. 279–295. doi: 10.1007/978-90-481-8847-5_13.

[7] ‘Elementary Multiperspective Material Ontology (EMMO)’. Elementary Multiperspective Material Ontology (EMMO), Nov. 03, 2022. Accessed: Nov. 11, 2022. [Online]. Available: https://github.com/emmo-repo/EMMO

[8] ‘EMMC ASBL | The European Materials Modelling Council’, *https://emmc.eu/*. https://emmc.eu/ (accessed Nov. 11, 2022).

[9] ‘Data Catalog Vocabulary (DCAT) - Version 2’. https://www.w3.org/TR/vocab-dcat-2/

[10] ‘DCMI Metadata Terms’. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ (accessed Nov. 11, 2022).

[11] ‘FOAF Vocabulary Specification’. http://xmlns.com/foaf/0.1/ (accessed Nov. 11, 2022).

[12] ‘PROV-O: The PROV Ontology’. https://www.w3.org/TR/prov-o/ (accessed Nov. 11, 2022).

[13] E. Ghedini, A. Hashibon, and J. Friis, ‘Deliverable D3.1 - “Semantic data exchange ontology”’, 2022. [Online]. Available: https://dome40.eu/sites/default/files/2022-11/DOME 4.0 D3.1 Semantic data exchange ontology 2022.08.05 PU - revised.pdf

[14] ‘OntoCommons ontology catalogue’. https://data.ontocommons.linkeddata.es/index (accessed Nov. 11, 2022).

[15] Y. L. Franc, ‘OntoCommons D3.2 - Report on existing domain ontologies in’, Mar. 2022, doi: 10.5281/zenodo.6504553.

[16] M. Poveda-Villalón, ‘OntoCommons D3.3 - Report on populated domain ontology registry’, Mar. 2022, doi: 10.5281/zenodo.6504584.

[17] ‘European Science Vocabulary (EuroSciVoc) - EU Vocabularies - Publications Office of the EU’. https://op.europa.eu/en/web/eu-vocabularies/euroscivoc (accessed Nov. 11, 2022).

[18] Directorate-General for Research and Innovation (European Commission) and A. F. de Baas, *What makes a material function?: let me compute the ways : modelling in H2020 LEIT NMBP programme materials and nanotechnology projects : sixth version*. LU: Publications Office of the European Union, 2017. Accessed: Nov. 11, 2022. [Online]. Available: https://data.europa.eu/doi/10.2777/417118

[19] ‘FAIR metrics and Data Quality | EOSC Association’. https://www.eosc.eu/advisory-groups/fair-metrics-and-data-quality (accessed Nov. 16, 2022).

[20] M. D. Wilkinson, S.-A. Sansone, E. Schultes, P. Doorn, L. O. Bonino da Silva Santos, and M. Dumontier, ‘A design framework and exemplar metrics for FAIRness’, *Sci. Data*, vol. 5, no. 1, Art. no. 1, Jun. 2018, doi: 10.1038/sdata.2018.118.

[21] R. Huber and A. Devaraju, ‘F-UJI : An Automated Tool for the Assessment and Improvement of the FAIRness of Research Data’, Copernicus Meetings, EGU21-15922, Mar. 2021. doi: 10.5194/egusphere-egu21-15922.
[22] ‘Data on the Web Best Practices’. https://www.w3.org/TR/dwbp/ (accessed Nov. 16, 2022).
[23] ‘FAIR Data Maturity Model WG’, *RDA*, Sep. 23, 2018. https://www.rd-alliance.org/groups/fair-data-maturity-model-wg (accessed Nov. 16, 2022).
[24] ‘FAIRsFAIR project home page’. FAIRsFAIR.eu
[25] ‘EOSC synergy – Building capacity, developing capability’. https://www.eosc-synergy.eu/ (accessed Nov. 16, 2022).
[26] ‘FAIR Framework – EOSC synergy’. https://www.eosc-synergy.eu/results/fair-framework/ (accessed Nov. 16, 2022).
[27] ‘FAIRsharing | Home’. https://fairsharing.org/ (accessed Nov. 16, 2022).
[28] ‘The FAIR Maturity Evaluation Service’. https://fairsharing.github.io/FAIR-Evaluator-FrontEnd/#!/ (accessed Nov. 16, 2022).
[29] ‘FAIRsharing Maturity Indicators’. https://fairsharing.org/search?q=&selected_facets=type_exact%3Ametric&fairsharingRegistry=Standard (accessed Nov. 16, 2022).
[30] M. E. Conway, ‘How Do Committees Invent?’, *Datamation magazine*, Apr. 1968. Accessed: Nov. 14, 2022. [Online]. Available: http://www.melconway.com/Home/Committees_Paper.html
[31] ‘GitHub’, *GitHub*. https://github.com/ (accessed Nov. 15, 2022).
[32] ‘GitLab’. https://about.gitlab.com/ (accessed Nov. 15, 2022).
[33] Atlassian, ‘Bitbucket’, *Bitbucket*. https://bitbucket.org/product (accessed Nov. 15, 2022).
[34] ‘NanoSim’, *(2014-2017) received funding from the European Union’s FP7-NMP research and innovation programme under grant agreement no 604656*.
[35] ‘SimPhoNy’, *(2014-2017) receives funding from the European Union’s FP7 under NMP-2013-1.4-1 call with Grant agreement no: 604005.*
[36] ‘Virtual Materials Market Place (VIMMP)’, *(2018-2022) receives funding from the European Union’s Horizon 2020 Research and Innovation Programme, under Grant Agreement no. 760907*.
[37] ‘MarketPlace’, *(2018-2022) receives funding from the European Union’s Horizon 2020 Research and Innovation Programme - H2020-NMBP-25-2017 Open Innovation Platform for Materials Modelling, under Grant Agreement no: 760173.*
[38] ‘SimDome’, *(2019-2023) received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement number 814492.*
[39] ‘OntoTrans’, *(2020-2024) receives funding from the European Union’s Horizon 2020 Research and Innovation Programme, under Grant Agreement no. 862136.*
[40] ‘OpenModel’, *(2021-2025) receives funding from the European Union’s Horizon 2020 Research and Innovation Programme, under Grant Agreement no. 953167*.
[41] ‘VIPCOAT’, *(2021-2025) receives funding from the European Union’s Horizon 2020 Research and Innovation Programme - DT-NMBP-11-2020 Open Innovation Platform for Materials Modelling, under Grant Agreement no: 952903.*
[42] M. G. Skjæveland, L. A. Slaughter, and C. Kindermann, ‘OntoCommons D4.3 - Report on Landscape Analysis of Ontology Engineering Tools’, Apr. 2022, doi: 10.5281/zenodo.6504670.
[43] ‘Status of Python Versions’. https://devguide.python.org/versions/ (accessed Nov. 15, 2022).
[44] N. G. Connelly, T. Damhus, R. M. Hartshorn, and A. T. Hutton, Eds., *Nomenclature of Inorganic Chemistry: IUPAC Recommendations 2005*. The Royal Society of Chemistry, 2005.
[45] H. A. Favre and W. H. Powell, *Nomenclature of Organic Chemistry: IUPAC Recommendations and Preferred Names 2013*. The Royal Society of Chemistry, 2014. doi: 10.1039/9781849733069.
[46] ‘Cheméo’, *Cheméo*. https://www.chemeo.com/ (accessed Nov. 11, 2022).

[47] PubChem, ‘PubChem’. https://pubchem.ncbi.nlm.nih.gov/ (accessed Nov. 11, 2022).
[48] D. Weininger, ‘SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules’, *J. Chem. Inf. Comput. Sci.*, vol. 28, no. 1, pp. 31–36, Feb. 1988, doi: 10.1021/ci00057a005.
[49] ‘CAS REGISTRY and CAS Registry Number FAQs’, *CAS*. https://www.cas.org/support/documentation/chemical-substances/faqs (accessed Nov. 11, 2022).
[50] ‘EC Inventory - ECHA’. https://echa.europa.eu/information-on-chemicals/ec-inventory (accessed Nov. 15, 2022).
[51] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi, ‘InChI, the IUPAC International Chemical Identifier’, *J. Cheminformatics*, vol. 7, no. 1, p. 23, May 2015, doi: 10.1186/s13321-015-0068-4.

11 Acknowledgement

The author(s) would like to thank the partners in the project for their valuable comments on previous drafts and for performing the review.

Project partners:

#	Type	Partner	Partner full name
1	SME	CMCL	Computational Modelling Cambridge Limited
2	Research	FHG	Fraunhofer Gesellschaft zur Förderung der Angewandten Forschung E.V.
3	Research	INTRA	Intrasoft International SA
4	University	UNIBO	Alma Mater Studiorum – Universita di Bologna
5	University	EPFL	Ecole Polytechnique Federale de Lausanne
6	Research	UKRI	United Kingdom Research and Innovation
7	Large Industry	SISW	Siemens Industry Software NV
8	Large Industry	BOSCH	Robert Bosch GmbH
9	SME	UNR	Uniresearch B.V.
10	Research	SINTEF	SINTEF AS
11	SME	CNT	Cambridge Nanomaterials Technology LTD
12	University	UCL	University College London

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953163. It is the property of the DOME 4.0 consortium and do not necessarily reflect the views of the European Commission.

12 Table of Abbreviations

Abbreviation	Explanation
ADMS	Asset Description Metadata Schema
API	Application Programming Interface
BFO	Basic Formal Ontology
BSD	Berkeley Software Distribution
CAS	Chemical Abstracts Service
CI/CD	Continuous Integration/Continuous Deployment
COAR	Confederation of Open Access Repositories
CORDIS	Community Research and Development Information Service
CSS	Cascading Style Sheets
CSVW	Namespace Vocabulary Terms
CUDS	Common Universal Data Structure
DBMS	Database Management System
DCAT	Data Catalog Vocabulary
DCTERMS	Dublin Core metadata initiative metadata terms
DOI	Digital Object Identifier
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DQV	Data Quality Vocabulary
EC	European Commission
EMMC	European Materials Modelling Council
EMMO	Elemental Multiperspective Material Ontology
EOSC	European Open Science Cloud
FAIR	Findable, Accessible, Interoperable, and Reusable
FOAF	Friend of a Friend
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDS	International Data Spaces
IRI	Internationalized Resource Identifier
ISBN	International Standard Book Number
ISO	International Organization for Standardization
IUPAC	International Union of Pure and Applied Chemistry
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
MD5	Message-Digest algorithm
OAS	OpenAPI Specification
OCES	Ontology Commons EcoSystem
ORCID	Open Researcher and Contributor ID
OS	Operating System
OWL	Web Ontology Language

PHP	Hypertext Pre-processor
PROV-O	The provenance ontology
QUDT	Quantities, Units, Dimensions and Data Types Ontologies
RDA	Research Data Alliance
RDB	relational database
RDF	Resource Description Framework
REST	REpresentational State Transfer
RFC	Request for Comments
RoMM	Review of Materials Modelling
RST	reStructuredText
SDMX	Statistical Data and Metadata Exchange
SEO	Search Engine Optimization
SKOS	Simple Knowledge Organization System
SMILES	Simplified Molecular-Input Line-Entry System
SOFT	SINTEF Open Framework and Tools
SPA	Single Page Application
SPARQL	SPARQL Protocol and RDF Query Language
SPDX	Software Package Data Exchange
SQL	Structured Query Language
UUID	Universally Unique Identifier
W3C	World Wide Web Consortium
YAML	YAML Ain't Markup Language

Annex 1

RDFS Data Exchange Vocabularies (RDFS-DEV)

Here we briefly list the RDFS-DEV from which we will select the terms relevant for the scope of the project DOME 4.0.

DCTERMS

Dublin Core[3] is a set of properties (vocabulary) for associating metadata with resources. It was originally developed to describe library resources, particularly documents, video files, books etc., and later extended for web resources, and it has also been used to describe a variety of other physical and digital resources.

The Dublin Core metadata initiative includes the fifteen terms in the Dublin Core Metadata Element Set in addition to a larger set of properties, classes, datatypes, and schemes. Together, they are collectively referred to as "DCMI metadata terms" or "Dublin Core terms" (DCTERMS) for short.

DCTERMS are expressed in RDF vocabularies. Each term is identified with a Uniform Resource Identifier (URI), which is a global identifier usable in Linked Data. Built into the Dublin Core standard are definitions of each metadata element – like native content standard – that state what kinds of information should be recorded where and how. Associated with many of the data elements are data value standards such as the DCMI Type Vocabulary and ISO 639 language codes.

We will hereafter refer to the RDF representation of the http://purl.org/dc/terms/ namespace published on 2020-01-20, and available at https://www.dublincore.org/schemas/rdfs/.

DCAT

DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogues published on the Web. It enables a publisher to describe datasets and data services in a catalogue using a standard model and vocabulary that facilitates the consumption and aggregation of metadata from multiple catalogues. This can increase the discoverability of datasets and data services. It also makes it possible to have a decentralised approach to publishing data catalogues and makes federated search for datasets across catalogues in multiple sites possible using the same query mechanism and structure. Aggregated DCAT metadata can serve as a manifest file as part of the digital preservation process.

As illustrated in Figure 1, DCAT relies on the FOAF and DCTERMS vocabularies. Note that while the current widely used DCAT version is 2.0, this work covers both the stable version 2.0 and the upcoming version 3.0. Whenever needed the explicit version will be mentioned.

We will refer to the official RDF representation of DCAT version 2 available at https://github.com/w3c/dxwg/blob/gh-pages/dcat/rdf/dcat2.ttl.

[3] https://www.dublincore.org/

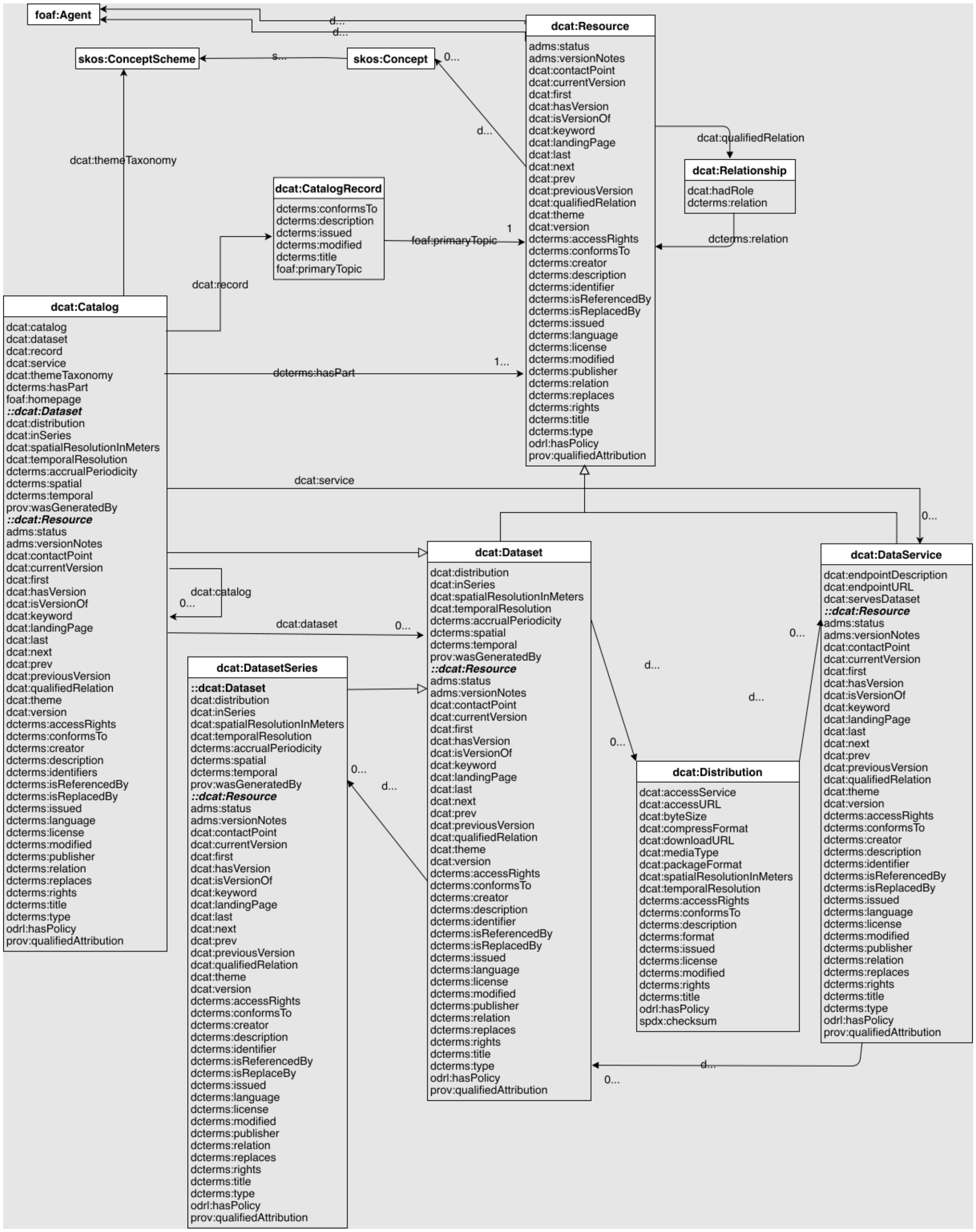

Figure 4: The DCAT (shown is Version 3) schema relies on DCTERMS, FOAF, SKOS, etc. Image from https://www.w3.org/TR/vocab-dcat-3/images/dcat-all-attributes.svg

PROV-O

The provenance ontology, PROV (PROV-O) expresses the so called PROV Data Model[4] using the OWL2 Web Ontology Language (OWL2). PROV-O aims to provide a set of classes, properties, and restrictions that can be used to represent and interchange provenance information generated in different systems and under different contexts. PROV-O has three different main parts, arranged from the most simple and fundamental terms (and concepts) needed for simple applications of provenance to more complex ones. These are the 1) Starting Point terms, 2) Expanded terms, and 3) terms for Qualifying relationships.

The **Starting Point classes** and properties provide the basis for the rest of the PROV Ontology and are used to create simple provenance descriptions. These include as shown in Figure 3 terms such as wasDerivedFrom, wasGeneratedBy, etc. These provide the minimal provenance elements.

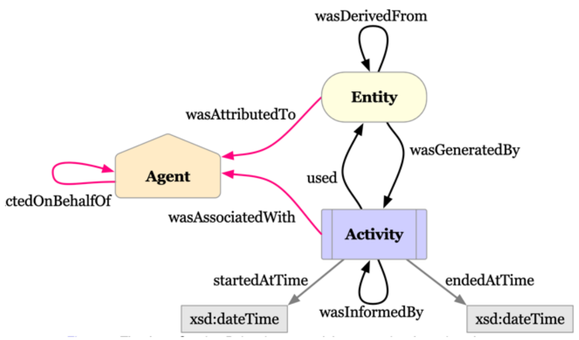

Figure 5: The three Starting Point classes and the properties that relate them. From https://www.w3.org/TR/prov-o/#description-starting-point-terms.

The **Expanded classes** and properties provide additional terms such as the special concepts that generate a dataset e.g., Person, or Organisation while the **Qualified classes** and properties provide elaborated information about binary relations asserted using Starting Point and Expanded properties. These include e.g., Start, End, Usage, of a data set and similar concepts. The entire PROV-O can be consumed by EMMO directly with the elementary mappings proposed here.

[4] http://www.w3.org/TR/2013/REC-prov-dm-20130430/

Annex 2

The FAIR codes listed in the table below is based on [20], and more information about the FAIR metrics can be found on fairmetrics.org.

FAIR CODE	Very brief explanation	Standards, suggestions with comments.
FM1-F1A	Global identifiers used.	e.g. • DOI, ror.org, ORCID and well known IRI schemes based on good practices e.g., concepts and their associated UUID or Hex hash digest. • IRI Internationalised URI. RFC 3987 (identifiers) • DOI(Crossref): Digital Object Identifier. Popular with scientific assets like published papers and data. • ORCID: Open Researcher and Contributor ID • Ror.org: Permanent ID registry for organisation and institutions. • Hex hash digest (among others) • MD5 • Hash algorithm. RFC 1321 • SHA • Family of hash algorithms RFC 4634. • W3C Asset Description Metadata Schema (ADMS) • We can use ADMS for identifiers that are text literals like e.g. ISBN
FM1-F1B	The identifier must be permanent. (Persistent Identifier)	Same as above.
FM-F2	Rich metadata.	In our case, semantic linked data. Extendible in many ways and with multiple serialisations.
FM-F3	Data includes identifiers.	The metadata contains the identifiers of other objects, including the data.
FM-F4	Data indexed in a searchable resource.	Public data can be searched somewhere public.
FMA1	Retrievable by identifier using a standard protocol	Yes, HTTP

FM-A1.1	Open, free etc communications protocol.	Yes, HTTP
FM-A1.2	Protocol allows for authentication and authorisation.	Yes
FM-A2	Metadata serialisation is accessible even when data files are no longer available.	Yes
FM-I1	It uses language for knowledge representation.	<u>Enumeration of standards.</u> This list can be expanded as needs arise. • W3C OWL 2: Web Ontology Language • Topics categories (coarse)/ Themes (fine) (i.e. not free keywords) • Scientific categories: EuroSciVoc • Multiple serialisations of RDF: • W3C RDF Schema> Resource Description Framework. (e.g., *W3C JSON-LD JSON for Linked Data. RDF document (serialisation). 100% backward compatible with JSON.*) • COAR for *access level* (e.g.) Confederation of Open Access Repositories. • Metadata for CSV files: W3C CSVW Vocabulary for tabular data on the web. Datasets: • W3C DCAT Data Catalog Vocabulary based on OWL2 and RDF. Includes elements from DCTERMS, FOAF and PROV-O. Non-normative vocabularies and ontologies also suggested. Further provenance and lineage: • W3C PROV-V Provenance Vocabulary.

		Duration and timestamping: • ISO 639-1: Two characters language ID. • ISO 639-2: Three characters language ID. • ISO 8601: Datetime standard (RFC 3339). Scientific parameters and I/O: • W3C DQV Data Quality Vocabulary. (e.g., for scientific I/O, parametrisation) • ISO 17369:2013 aka SDMX 2.1 Statistical Data and Metadata Exchange (e.g. for scientific attributes) • QUDT (for SI units) Scientific Description: • EMMO
FM-I2	Vocabularies used are themselves FAIR.	Yes. International standards.
FM-I4	It uses qualified references to other metadata.	Yes. International standards.
FM-R1	Richly described with accurate and relevant attributes.	Yes, it is extendible and progressively enriched.
FM-R1.1	It must have open licences identifiers. Other licences identifiers can be added.	ISO/IEC 5962:2021 aka SPDX 2.2.1 Software Package Data Exchange. (e.g., Open licences identifiers)
FM-R1.2	Detailed provenance must be possible	DCAT (datasets) and PROV-O.
FM-R1.3	Meets community standards	Metadata and data (files) are processable by users. Open file standards encouraged. Data views on request (if applicable).