Open reference architecture for engineering model spaces

Deliverable 5.1

State-of-the-Art and Guidelines for Model Stores

Project Coordinator	Olga Kattan, Philips Consumer Lifestyle B.V.		
Start date Project	April 1st 2024	**Duration**	36 months
Version	1.0		
Status	Final		
Date of issue	10-09-2025		
Dissemination level	Public		

Authors' data

Author	Beneficiary
Dilara Bayar	ACD
Hande Samanlı	ACD
Mathias Verbeke	KU Leuven
H. Burak Erözdemir	Inovasyon Muhendislik
Roelof Hamberg	Canon Production Printing Netherlands B.V.
Behrouz Khiali	Canon Production Printing Netherlands B.V.
Richard Doornbos	TNO
Tomas Molina	Thermo Fisher Scientific
Antons Prokopenko	Philips Consumer Lifestyle B.V.
Klaus Wolf	Fraunhofer-Institute for Algorithms and Scientific Computing SCAI
Bram Stalknecht	Semlab
Ersin Sığınç	Alpata
Nicolas Lammens	Siemens
Final editor's address	Olga Kattan Philips Consumer Lifestyle B.V. Oliemolenstraat 5 9203ZN Drachten / Netherlands

Executive Summary

D5.1 - State-of-the-Art and Guidelines for model stores presents the state-of-the-art analysis and initial guidelines for the design and implementation of model stores within the SmartEM framework. A model store is defined as a digital environment that enables the storage, management, retrieval, and governance of engineering models, supporting their reusability and interoperability.

The deliverable reviews current practices, technologies, and standards relevant to model storage, with a focus on semantic interoperability, structured metadata, and robust governance mechanisms. It surveys existing platforms and tools from both industrial and research contexts, identifying best practices and gaps in areas such as:

- Use of metadata and ontologies to describe models and their provenance.
- Model lifecycle management, including versioning, validation, and archival.
- Access control and IP rights management in collaborative environments.
- Search and retrieval functionalities supporting both keyword-based and semantic queries.
- Integration with AI/ML pipelines for surrogate modelling and predictive analytics.

In addition, the report establishes preliminary design principles and architectural guidelines to ensure that the SmartEM model store will:

- Support heterogeneous model types, including first-principles-based, data-driven, and surrogate models.
- Enable interoperability through standardised metadata schemas and domain ontologies.
- Provide governance workflows aligned with industrial IP protection and compliance requirements.
- Facilitate model discovery, evaluation, and reuse across domains and use cases.

The findings in D5.1 form the basis for the implementation phase of SmartEM's model store, laying the groundwork for a scalable, secure, and semantically rich platform for engineering model management.

Table of Contents

1. Introduction

1.1. Objective and Scope of D5.1

In this deliverable, guiding principles and architectures for model store implementations will be elaborated, covering aspects such as decentralisation, scalability, collaboration support, federation, interoperability, compatibility, trust management, and auditability. This work takes its inspiration not only from existing open-source components available for data spaces (e.g. from IDSA, GAIA-X or Eclipse open-source endeavours) and reference implementations proposed by the Data Spaces Support Centre, but also from the state-of-the-art in the field of Model-Driven Engineering (MDE), existing toolsets, and the state-of-practice in advanced industries.

Relation to WP5 and the Overall SmartEM Architecture

The use case specific model store implementations can use, combine and adapt multiple generic building blocks defined across WP5. Possible synergies and links with product lifecycle management (PLM) software will be investigated. Additionally, use case specific modules (e.g. for data comparison, customised performance checks, validation with real data, etc) can be implemented following the same guidelines.

2. Model Store: Concept and State-of-the-Art

This section outlines concepts, terminology, and relevant technologies associated with model stores. Model stores store models in a managed way such that collaboration in a (possibly cross-site, cross-company) engineering community is supported in a productive and secure way.

Model spaces resemble data spaces, which are discussed in Section 2.1, extended with outlining an important aspect of models, i.e. their characterisation, in Section 2.2. Section 2.3 discusses the fundamental functions of model spaces, while Section 2.4 considers actual existing tools and technologies. Lastly, some non-functional requirements are treated in Section 2.5.

2.1. Fundamentals of Data Spaces

When we talk about secure model stores, we have to mention data spaces and recognize their critical role in enabling trusted and sovereign model and data exchange. One way to consider models in model stores is to view them as specific, complex, types of data that are subject to largely the same set of functionalities as where data spaces are designed for.

2.1.1. Overview of Data Spaces

Data Spaces are data sharing frameworks that enable cross-collaboration and data exchange mechanisms between multiple organizations (e.g. public sector, businesses, research institutes and many more). Such a framework allows the use of data and services, while maintaining the full control over your assets. It focuses on data de-centralization, providing technical architecture that supports secure and privacy-preserving data and model sharing, guaranteeing that organization sensitive information is accessed only under clearly defined policies.

The structure of Data Spaces contains the following basic elements:

- Providers and Consumers: Sensors/databases/decision makers
- Governance: Policies that define how data is shared and accessed (security)
- Infrastructure: Data storage, processing (APIs, data lakes, etc).

The most well-known initiatives are Gaia-X and International Data Spaces Association (IDSA).

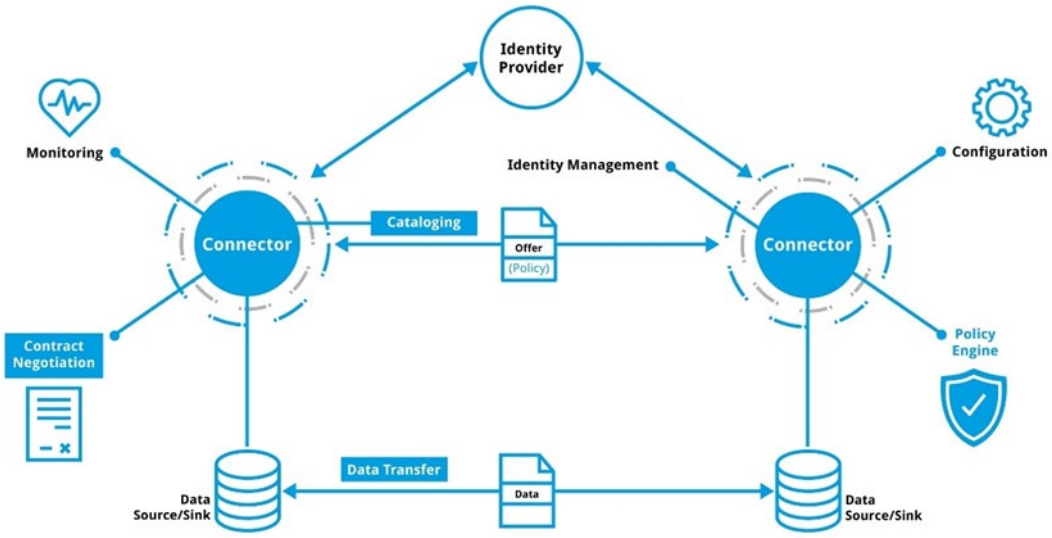

Figure 2.1: Overview on protocol and context of IDSA [1]

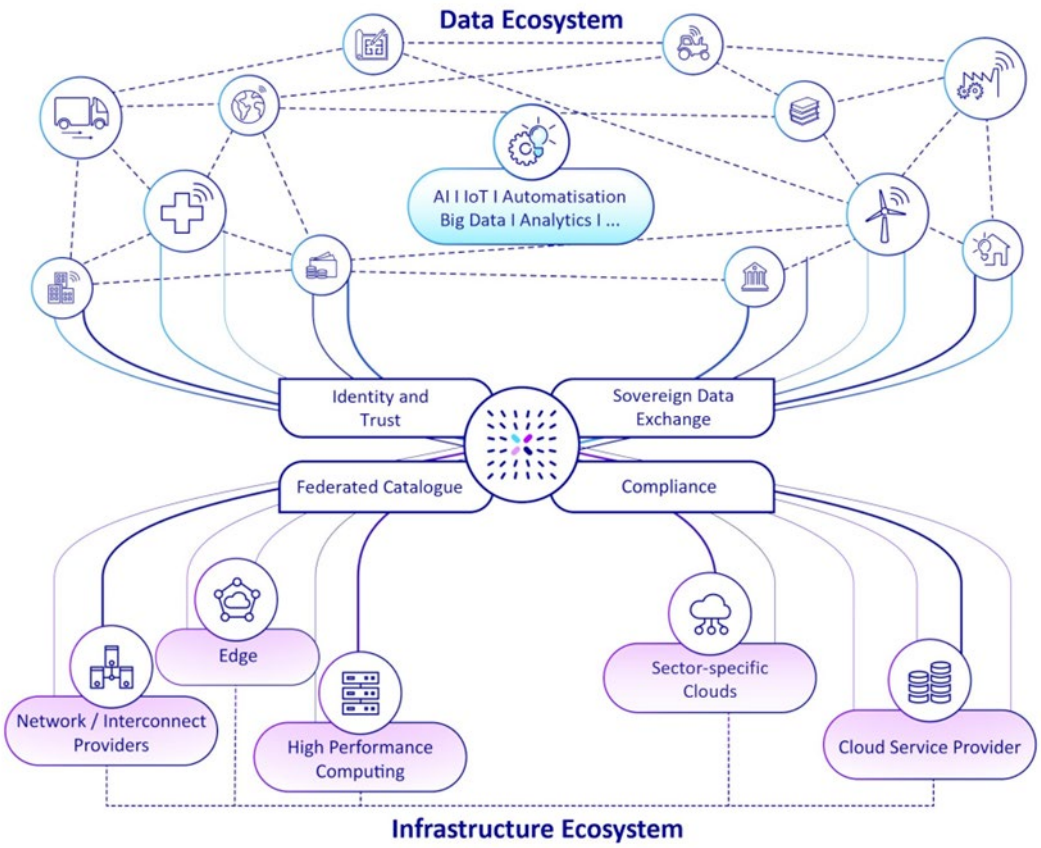

Figure 2.2: Overview of Gaia-X ecosystems [2]

IDSA is an organization develops and promotes standards for secure and sovereign data exchange. For an overview see *Figure 2.1* IDSA provides a standardized architecture for creating and managing dataspaces [1].

Gaia-X is an initiative aimed at creating a federated and secure data infrastructure. For an overview see *Figure 2.2*. Its promotes standardized protocols formats, implements security measures to protect data, ensures data owners retain control over their data [2].

In the context of engineering models exchange, which is the core focus of SmartEM, we will still have to make decisions as to which degree the architectural decompositions, formats, protocols, standards, etc. will serve as starting points for the reference architecture considering model stores. It is reasonable to expect that especially the non-functional aspects will be significantly driven by the mentioned data spaces setups: identity/trust, sovereign data exchange, secure data exchange.

2.1.2. Relevance to Engineering Model Exchange

In many industries today, like development of consumer products, energy, or manufacturing, (machine learning) models are not just used internally. Next to the so-called first principles models, also more and more ML & AI techniques are becoming valuable digital assets that companies want to share or reuse, either with partners, customers, or especially across departments.

However, sharing models can become a challenging task, especially when the developed model has a niche application & deals with business critical & confidential information. This is where data spaces play their crucial role. They provide rules and infrastructure that make it possible to safely share models across multiple organizations.

The direct advantage for a company to share their model across other organizations can be:

- Building stronger partnerships

- Potentially unlocking new revenue streams by charging fees for model use
- Optimizing its own products using real-world data
- Complying with continuously evolving industry and regulatory standards

Data Spaces ensure that the model owner retains control and transparency of how the model is being used, allowing others to contribute and benefit from using the model.

2.2. Model and Metadata Structures

In SmartEM we focus on so-called Engineering Models: models used in an engineering context, in order to simulate the behaviour of a system or predict the outcome of certain designs or design changes. Models can be e.g. mathematical models, their instantiations (forming an experiment), or a set of data (from experiments with models or the system of a part of a system).

When these models are to be reused, or shared (so not just one-time-use models, or models used by a single of very limited number of users), the models must be considered as part of a systems engineering process and managed as such. In order to be managed (stored, retrieved, (re-)used), these models must have some form of identification and description. Without these, it becomes difficult to find them, retrieve them, and use them, or hand them over to third parties.

When trying to completely describe all the possible metadata for all possible models, one will end up with a huge set of metadata. Describing the model with all these data becomes nearly impossible. For that reason, WP3 looks for a minimal high-level description or ontology for the metadata that will capture all models. Next the metadata for a specific domain or model can be extended with the domain or model specific data, that is needed for easy retrieval and reuse. The description of this metadata and ontology can be found in the results from WP3.

2.3. Model Store Fundamentals and Lifecycle

A model store is more than just a repository, it is a structured environment designed to support the full lifecycle of models while enabling collaboration, reuse, and integration across teams and tools. Therefore, three main classes of functionalities inspired by this are constituting the structure of this section.

2.3.1. Core Functionalities

To achieve reusability, traceability, and interoperability, a model store typically supports the following key capabilities:

Structured Metadata Management: Every model is accompanied by rich metadata that describes its purpose, input/output parameters, assumptions, domain, version, and licensing. This makes models searchable, understandable, and easier to integrate into different workflows.

Version Control and Traceability: Like software, models evolve. The store maintains a version history and links between versions, enabling users to trace changes, compare revisions, and understand how and why a model has changed over time. Older versions of models remain accessible where needed.

Interoperability Support: A good model store allows models to be used across different simulation tools or platforms by using standard formats, APIs, or connectors. This ensures that models are not locked into specific tools and can be embedded into broader engineering workflows.

Model Validation and Testing Hooks: Optionally, models can be linked to validation results, test data, or performance benchmarks, giving users more confidence in reusing them.

Access Control and Usage Policies: Models may be shared widely or kept within defined groups. Access permissions and licensing rules are enforced through the store, protecting intellectual property while enabling collaboration. Fine-grained access control manages who

can view, download, or modify models. Additionally, auditing should be in place to track who accesses the models. These measures ensure that the model store complies with relevant security standards and regulations, protecting sensitive intellectual property (IP) data.

2.3.2. Lifecycle Activities

The model store supports the following phases of a model's life:
Authoring and Uploading: Developers create and submit models along with metadata, version tags, and documentation. When uploading a model, users must provide essential metadata such as the model's name, description, version, and other relevant details to aid in organizing and searching for models later. Users can upload models in various formats, including simulation models and machine learning models. To ensure security, only authorized users should be allowed to upload models. This can be managed through standard authentication mechanisms. The system should validate the uploaded models to ensure they meet predefined standards and compliance requirements. This validation process may include checking for file integrity, format compliance, and the presence of required metadata.

Review and Curation: Some stores support community or expert review, tagging, and validation before models become widely accessible.

Update and History: When users update a model, the system should create a new version while retaining the previous versions which allows proper tracking of changes. Similar to the upload, the updated model must be validated to ensure it meets the standards and required formats. The system maintains a history of all the versions of the model and users can view the history including metadata and change logs.
Discovery and Use: Provision of robust search and discovery capabilities, including basic search using keywords, tags, and metadata, as well as advanced search filters. In this way, users can search for models based on metadata, parameters, or use cases, and directly download or integrate them.
Feedback and Update: Users can rate, comment on, or suggest changes to models. This feedback loop supports continuous improvement and shared understanding.
Archiving or Deprecation: Older models can be deprecated, but they remain traceable for reference or reproducibility.

2.3.3. Enabling a Modelling Community

Beyond the technical functions, an effective model store plays a social and collaborative role. It serves as a meeting point between model developers and users, fostering a modelling community that benefits from shared knowledge and best practices.

To support this, a model store may include:
Collaboration Features: Comments, annotations, user forums, or discussion threads tied to specific models.
Attribution and Recognition: Properly attributing model creators and contributors, which motivates sharing and builds trust.
Use Analytics and Provenance Tracking: Seeing how and where models are used can reinforce their value and inspire further contributions.
Templates and Guidelines: Helping model creators follow standards makes models more consistent and usable.

Ultimately, a successful model store creates a culture of sharing and reuse, where technical robustness is combined with openness, trust, and recognition. This is especially important in multi-partner initiatives like SmartEM, where interoperability and collaboration are key goals.

2.4. Existing Tools and Technologies

Model stores, in practice, come in different forms, ranging from specialized repositories to general-purpose platforms that support parts of the functionality required in engineering environments. While few tools offer all the features described earlier out-of-the-box, many

offer valuable capabilities such as versioning, metadata tagging, access control, or model sharing. Below is a more detailed look at some relevant tools and technologies:

Modelica Libraries and OpenModelica: In the world of physical modelling, Modelica [3] is a widely used language for building system models, everything from thermal systems to electrical circuits. Tools like OpenModelica [4] and Dymola [5] allow users to build and simulate these models, which are often shared as versioned libraries. These libraries are typically organized using clear naming conventions and metadata, making them easy to reuse across different projects. However, sharing is still mostly done through Git or internal repositories, so while the reuse potential is high, the ecosystem still lacks a true model store experience.

Synera: Synera [6] is a tool designed to help engineers create and share visual workflows. It's like a no-code environment where models and tools can be connected in drag-and-drop style. It supports sharing workflows across teams and connecting different tools together, which helps with interoperability. Although it's not a dedicated model store, it does encourage reusability by letting people package models into modules that can be shared or reused in other workflows.

Simcenter HEEDS: Siemens' Simcenter HEEDS [7] is a commercial tool used for design space exploration and optimization. It connects to various simulation tools and helps automating parameter sweeps and scenario testing. While it is not a model store in the traditional sense, HEEDS does offer features like model tracking, version control, and reusable workflows. It is mostly used internally within engineering teams and doesn't focus much on community sharing or external reuse.

Aras Innovator: Aras Innovator [8] is a product lifecycle management (PLM) system. It treats simulation models as part of the full digital product definition, alongside CAD files, test results, and requirements. It adds versioning, traceability, and permission controls, which are all useful for managing models in a complex industrial environment. It is very strong on governance and traceability but are often tightly linked to the company's internal infrastructure.

OpenMBEE and OpenMETA: These are open-source platforms originally developed under DARPA projects. OpenMBEE [9] focuses on systems engineering models, especially using SysML. It lets users manage both models and documentation in a connected way. OpenMETA [10] is a bit more experimental, it supports multi-domain modelling and lets you link together models from different tools. Both platforms support traceability and collaboration, and they're flexible if you're willing to do some customization.

Maple and MapleSim: Maple is a symbolic and numeric computation environment widely used in mathematics, physics, and engineering. Its companion tool, MapleSim, is a multi-domain system-level modeling and simulation platform that allows engineers to build, simulate, and analyze complex physical systems. MapleSim emphasizes reusability by supporting hierarchical modeling, parameter management, and model export to other simulation tools. While they provide strong capabilities for modeling and simulation, their ecosystem is still mostly organized around proprietary repositories and file sharing rather than a federated model store [11][12].

Zenodo, GitHub and DVC: These three are more general-purpose platforms, but they're often used in research for sharing models and data. GitHub [13] offers version control and collaboration tools, making it ideal for simulation code and scripts. Zenodo [14], on the other hand, is used to publish datasets or models with proper citations and DOIs. DVC [15] (Data Version Control) extends Git-like versioning to large datasets and machine learning models. It supports experiment tracking, remote storage, and reproducibility, making it popular in data science workflows. They are great for openness and transparency, but they don't offer domain-specific features like parameter search, simulation metadata, or tool integration out of the box.

MLflow, Seldon Core, and Hugging Face Hub: MLflow [16] is an open-source platform for managing the ML lifecycle, including experiment tracking, model packaging, and deployment. It includes a model registry with versioning and metadata tagging. Seldon Core [17] complements MLflow by focusing on scalable model deployment in Kubernetes environments. It supports explainability and monitoring but is more runtime-oriented.

Hugging Face Hub [18] is a community-driven platform for sharing pre-trained models. It supports versioning, metadata, and discoverability, and while originally focused on NLP, it now supports a broader range of models. These tools are strong in MLOps and collaboration but are less integrated with engineering simulation workflows or physical modelling.

Catena-X Model Repository and Eclipse EDC: Catena-X [19] is a data space initiative for the automotive industry. Its model repository supports semantic interoperability, traceability, and lifecycle management of digital twins and simulation models. Eclipse EDC [20] (Data Connector) provides the infrastructure for secure, governed data exchange across organizations. While not a model store itself, it enables federation, access control, and auditability, key enablers for distributed model sharing.

These tools are aligned with GAIA-X and IDSA principles and represent the emerging infrastructure for federated, cross-organizational model management.
Different tools cover different parts of the picture. Some are great at version control, others at collaboration or integration. But very few offer everything that is needed from a full-featured model store, especially one that works across organizations and toolchains. The key missing features are:

- Lack of cross-tool interoperability
- Limited support for semantic search or model discovery
- Weak integration with simulation metadata or parameter management

2.5. Reusability, Shareability, and Data Sovereignty Principles

2.5.1. Model Packaging Standards

There are several established model packaging standards that enable the secure, interoperable, and reusable exchange of machine learning models across organizational boundaries. A common practice among organizations is to encapsulate models within Docker or OCI-compliant containers, ensuring consistent and secure execution across different environments, regardless of any underlying system dependencies.

Alternative packaging approaches include frameworks such as MLflow, ONNX, and BentoML, which offer standardized ways to serialize and serve models as containerized, deployment-ready services. These formats support rich metadata, versioning, and model signature definitions, facilitating smooth integration with model registries.

Additionally, initiatives like the Open Model Packaging specification from the International Data Spaces Association (IDSA) and Gaia-X provide standardized representations of AI model assets. These specifications include metadata templates to support indexing in federated catalogues, policy enforcement, and lifecycle governance in data space ecosystems.

2.5.2. Private vs. Public models

Model Spaces offer robust security mechanisms and governance frameworks that allow organizations to maintain full control over their model assets. Nevertheless, many organizations remain cautious about sharing models externally. Sharing of models is a challenge within organizations as well: larger internal organizational distances (such as cross-project, cross-department, cross-site) lead to conservatism with regard to exchanging models.

Such conservatism is often justified by several key advantages of retaining model ownership in-house:

- High Domain Specificity: Some models are tailored for very narrow internal use cases, offering limited value outside the organization.
- Complete IP Control: Ensures proprietary models remain protected from external access or replication.

- Reduced Security Risk: Limits exposure to potential model theft or reverse engineering.
- Simplified Regulatory Compliance: Avoids the complexity of cross-organizational data governance, licensing, and usage auditing.
- Operational Efficiency: Eliminates the overhead associated with packaging, registering, and maintaining models in shared ecosystems.

On the other side of the coin there are also advantages of sharing models:

- Enhanced collaboration: Models can serve as a means to work together.
- Knowledge sharing: Models consolidate insights of experts in specific areas. As such models can be instrumental in sharing knowledge on specific domains.
- Crowdsourcing: Model assets might exist that accelerate the work of engineers, which wouldn't have been possible without sharing.
- Recognized leadership: If it is visible that an organization (or part of it) is leading the pack in a certain area, this will further strengthen the role of their models and competences by wider confrontation with challenges and applications.

The balance of the advantages and disadvantages of sharing models to a certain extent is to be determined for each company's case specifically and at every level of collaboration scope.

2.5.3. Control over model, metadata and inference

The Federated Service Governance, for example in Gaia-X, requires models to be provided with complete descriptions, like organization identity, functional metadata, licensing, associated costs or jurisdictional rules.

Using provided descriptions, specifically metadata, models become discoverable through federated catalogues. Metadata standards include information about model type, version, inputs and outputs, performance metrics, certifications and licensing terms.

When it comes to model inference (execution), Data Spaces follow several common inference approaches:

- Remote inference (Model-as-a-Service): Model stays with a provider and consumer has access to it over API.
- Trusted Execution Environment: Model is shared but executed only on isolated environments (e.g. Intel SGX).
- Local Inference: The model is available for download and capable of running offline on a local machine of a trusted partner.

3. Data Spaces and Model Store Integration

The model store component of the SmartEM framework is intended to be aligned with European data space architectures to ensure interoperability and compliance with emerging standards. In this context, models are envisaged to be not only technically accessible but also semantically discoverable, governed by well-defined rules, and documented in a structured manner. To achieve this, the framework draws upon reference architectures and technical guidelines established by prominent European initiatives, including the International Data Spaces Association (IDSA), GAIA-X, the Data Spaces Support Centre (DSSC), and the Big Data Value Association (BDVA), which collectively define interoperability, governance, and transparency requirements for cross-sector data sharing. These initiatives are referenced here as they provide the foundational principles and specifications necessary for ensuring that SmartEM can operate as part of a broader, standards-compliant data ecosystem. This section outlines the core principles and implementation proposals considered in line with these integration objectives[21][22] .

3.1. Conceptual Architecture of Data Spaces

The Model Store structure is a component designed to enable the management and sharing of models not only in a centralized manner but also in a federated way. Within the scope of SmartEM, this structure is intended to offer an architecture where each partner can host its own models while also collaborating with other stakeholders.

3.1.1. Model Store Concept

A Model Store is a core technical building block within data spaces designed to manage the registration, storage, governance, and interoperability of models of various types, ranging from computational and mathematical models to process, simulation, or data-driven models. Unlike traditional repositories limited to static assets, a model store in a data space is embedded in a broader semantic and interoperable infrastructure that ensures trust, traceability, and sovereign usage. This framework will be referred to throughout the document as the Soft Infrastructure.

Key Characteristics:

- *Semantic Discoverability:* Models are catalogued using machine-readable metadata and standardized ontologies that support their identification, classification, and semantic linking with other assets and data sources. This aligns with the metadata interoperability principles outlined in BDVA and OPEN DEI documents [21][22].
- *Lifecycle and Access Management:* A model store supports versioning, lifecycle governance, and policy-based access control, consistent with the soft infrastructure stack approach to data space interoperability [21].
- *Reuse and Exchangeability:* Through standardized APIs and data exchange protocols, models in the store can be invoked, reused, or composed within applications across domains and organizations supporting cross-domain integration and preventing fragmentation of model usage [22].
- *Alignment with Soft Infrastructure:* Model stores are not isolated systems; they are integrated into the stacked architecture of the data space soft infrastructure. This includes alignment with interoperability, trust, governance, and data value layers [21].
- *Support for Data Sovereignty:* As with data assets, models registered in the store include usage policies, provenance, and licensing metadata to ensure sovereign sharing, enabling organizations to retain control over how models are accessed and applied [21].

Functional Role in Data Spaces:

- In interoperability layers, the model store enables standardized description and referencing of models using ontologies and common taxonomies, following principles described in StandICT's ontology landscape [23].
- In governance structures, model usage is regulated through digital contracts, identity/authentication services, and audit mechanisms, just as with other data services [22].
- In service layers, models from the store are deployed and composed within domain-specific applications such as digital twins, simulations, or decision support systems [22].

This design satisfies the requirement that offerings be published in a semantically rich, interoperable manner [21].

3.1.2. Position Within the Federated Structure

One of the core components of data spaces, the federated infrastructure, may propose access to systems such as the model store through interconnected catalogues instead of a single central catalogue [22]. Through this structure:

- Each partner can host their own model store,
- These stores can be federatively connected via a central broker or catalogue connector,
- The definitions, identities, and access paths of the models can be made transparent through semantic metadata.

3.1.3. Layered Architectures and Interoperability Models

Data spaces are designed as federated ecosystems where independent actors such as companies, public institutions, or data providers, share and exchange data and services under mutually agreed rules while retaining control over their digital assets. To enable scalable and sovereign data sharing across heterogeneous systems and domains, data spaces adopt a layered architecture model and adhere to multi-dimensional interoperability frameworks.

The concept of a "soft infrastructure stack" was proposed by IDSA [21] to describe the logical separation of concerns in data spaces. Unlike traditional IT architectures that focus solely on infrastructure and software components, the soft infrastructure stack emphasizes the interplay of technical, semantic, legal, and governance aspects. This stack is composed of four major layers:

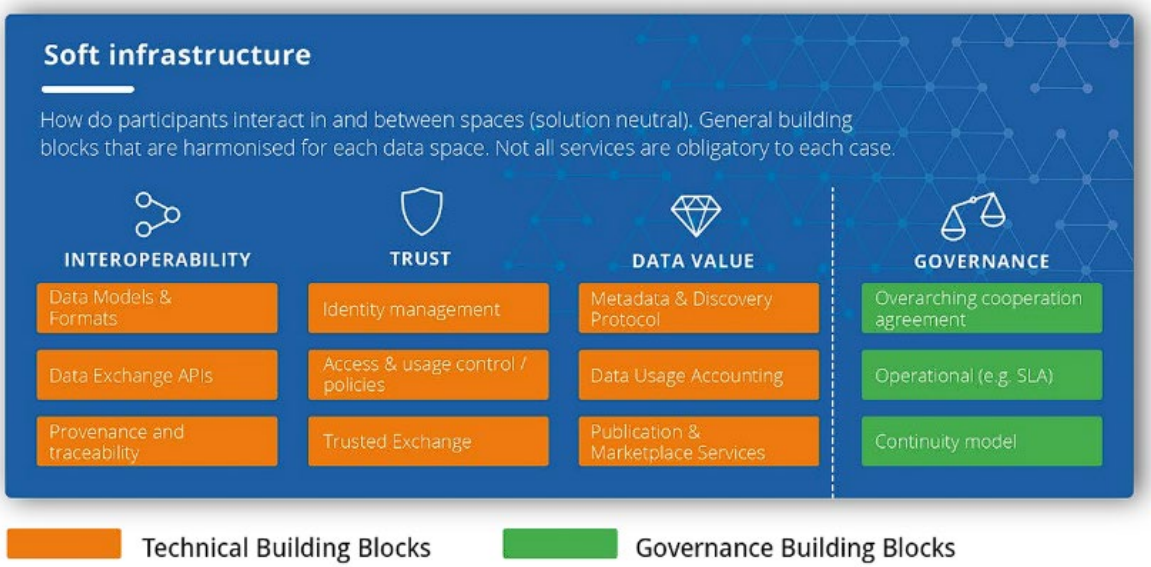

Figure 3.1: Soft Infrastructure Stack [21]

The BDVA four-layer model (technical, semantic, organisational, legal) is adopted so that SmartEM can interoperate across all dimensions [22], and the SmartEM platform can also be structured to correspond to these four levels. This would allow SmartEM to attain the flexibility to operate in an integrated manner with the European data space in both data exchange processes and governance mechanisms.

Technical Layer

This layer ensures that data sharing is secure, machine-readable, and accessible across systems. The platform delivers access to data assets needs to be controlled and secure. Authentication and authorization mechanisms like OAuth2 or token-based approaches are essential [21], it is recommended that access control be ensured securely at the technical level. In this context, the SmartEM structure is expected to include the following technical features:

- A model metadata structure configured in JSON format and readable by machines,
- Open interfaces that allow data exchange with external systems,
- Technical schema compatibility through fields such as uml_link, xml_export_link, and xsd_ready.

These structures provide components that support interoperability in terms of both data presentation and technical integrity.

Semantic Layer

The foundation of semantic interoperability is the sharing of data not only in form but also in meaning. Semantic interoperability ensures that the precise meaning of exchanged data is preserved and understood by all parties [21], it is essential that the meaning be preserved so that all stakeholders interpret the data in the same way. In this regard, SmartEM models are recommended to be enriched with fields such as:

- purpose,
- description,
- tags,
- inputs,
- outputs,

This will explicitly define the meaning of the data.

This approach enables the model to be shared not only technically but also semantically.

Organizational Layer

Interoperability at the organizational level aims to clarify roles, responsibilities, and workflows across systems. In this context:

- The definition of user profiles in the metadata with entries such as "roles": ["engineer", "analyst"],
- The specification of the model's origin, currency, and institutional responsibility via fields such as created_by, version, last_updated, can provide a structure that supports organizational transparency.

This structure facilitates authorization and role tracking when the model is used by different institutions.

Legal Layer

The legal layer aims to clearly define the conditions of use for data assets and to determine the rights and responsibilities of parties involved in data sharing. Metadata should include information such as data usage restrictions, licensing, and access policies [21], it is recommended that:

- The metadata include structures such as "access": { "authentication": "Token based", "roles": [...] },
- Licensing, access restrictions, and usage policies be technically specified, to ensure legal compliance.

This enables data-sharing processes to be legally traceable and auditable.

3.2. IDSA, GAIA-X, DSSC Compliance and Standards

The European data strategy aims to create a single market for data that will ensure Europe's global competitiveness and data sovereignty [21], the SmartEM platform is intended to be adapted in alignment with the European Commission's data strategy, supporting the principles of data sovereignty and interoperability. In this context, it is proposed that the platform be made compatible with the architectural frameworks and technical standards defined by institutions such as IDSA, GAIA-X, DSSC, and BDVA.

3.2.1. Alignment with IDSA Reference Architecture

Access to data assets needs to be controlled and secure. Authentication and authorization mechanisms like OAuth2 or token-based approaches are essential [21], it is recommended that the SmartEM platform be developed in compliance with the reference architecture of the International Data Spaces Association (IDSA). Within this scope:

- A token-based access mechanism,
- Role-based authorization (engineer, analyst),
- Definition of licensing and access rights in the model metadata,

should be made implementable within the platform. Additionally, Metadata should include information such as data usage restrictions, licensing, and access policies [21], it is recommended that access control and sharing policies be defined within the metadata of each model.

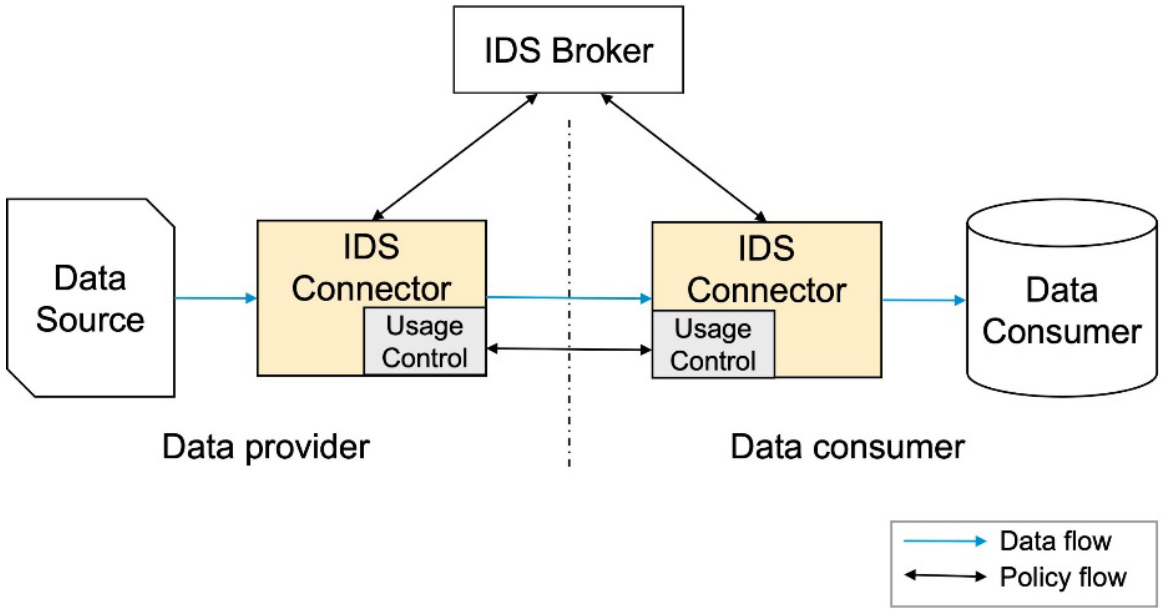

Figure 3.2: IDSA–Based Architectural Structure [21]

3.2.2. Alignment with GAIA-X Architecture

The GAIA-X architecture promotes the transparent, interoperable, and decentralized discoverability of data services. In this direction, the SmartEM model can be structured to:

- Ensure identifiability through fields such as model_id, uml_link, xml_export_link, and version,
- Enable data exchange using formats such as JSON and XML,
- Achieve semantic clarity through fields such as tags, purpose, and description.

It is recommended that these structures be organized to comply with GAIA-X's federated and semantic discovery approach.

3.2.3. DSSC Alignment Components

The Data Spaces Support Centre (DSSC) proposes open-source reference implementations and standard interfaces for ensuring interoperability. The DSSC will provide open-source reference implementations and building blocks to support common interoperability mechanisms and standardized interfaces such as REST APIs, JSON-LD metadata and usage policies [22], the SmartEM structure is recommended to include:

- Machine-readable metadata in JSON format,
- Data exchange with external systems through REST APIs,
- Configurable access policies within the metadata.

These implementations will comply with the technical compatibility structure targeted by the DSSC.

3.2.4. Alignment with BDVA's Four-Layer Interoperability Model

Within the framework presented by BDVA, The Interoperability framework includes four layers: technical, semantic, organizational and legal [22], it is recommended that the SmartEM model store be structured by considering the following layers:

Organizational: Clear definition of user roles (e.g., "engineer", "analyst") within the metadata,

Legal: Management of access permissions and licensing policies via REST APIs and metadata,

Semantic: Semantic interpretability of the model through input-output types, descriptions, and tagging,

Technical: Support for technical formats such as JSON, REST API, UML, and XML.

Considering these four layers together will provide a strong foundation for aligning SmartEM with the European data space framework.

3.3. Discovery Mechanisms and Federated Catalogues

SmartEM models are published through federated catalogues in line with the Metadata & Discovery Protocol building block, allowing data owners to register assets once and have them discovered through brokers [22].

3.4. Data Standards, Meta-Data, and Ontologies (SCAI)

SmartEM Deliverable 3.1 gives an overview of standards, and relate them to the use cases:

- These standards comprise: domain-specific technological standards (focus in use cases), standards for model descriptions and architectures, be it data models (e.g. ONNX) or physics-based models (e.g. SMILE), model exchange standards (e.g. FMI), model system standards (e.g. SSP) and neutral model formats (e.g. VMAP, STEP, BIM).
- D3.1 already identified opportunities to reuse, extend or align a model-centred digital ecosystem with these standards, as well as numerous de-facto standards, community conventions, industrial guidelines and industry standards from standardisation bodies (ISO, DIN) that imply commonplace terminology and core taxonomies. During the past decades, formal ontologies have gained more popularity with the emergence of CatenaX, SAMM and their derivatives, data space schemata (e.g. IDS ontology) and others.

While the reference architecture does not define the contents and format of single data blocks, such standards will enable an easy exchange and interpretation of data blocks by all standard compatible tools.

SmartEM deliverable 3.2 provides on meta-data attributes and ontology. Meta-data is the data used to describe the engineering models, in order to be able to store, but explicitly also to find them back, retrieve them, reuse them, and/or use them to generate surrogate models. Models is a broad concept here, including both the mathematical models, the experiments using the models, and the data generated by an experiment or measurements.

Based on D3.1 and 3.2, and in close relation to WP5, task 3.3 will further develop, adapt and deploy semantic concepts (ontologies) for model management. The ontology framework is maintained alongside a clear and structured methodological guideline for extending definitions. T3.3 essentially consists of the systematic transformation from human-logic definitions and loose hierarchies to formal taxonomies, object property restrictions and other axioms to capture the nature of the interrelated metadata and data properties needed. The task is an iterative design, implementation and test stage and as such transfers to the creation and implementation of such data attributes in support of labelling operations and SPARQL engine developments in WP5 and ontology-learning methods of WP4.

4. SmartEM Reference Architecture for Model Stores

SmartEM proposes an open reference architecture for engineering model spaces (model stores) to improve the reuse, exchange, and integration of computational engineering models. The goal is to enable domain-optimal model store implementations that all follow common rules, structures, and interfaces for reliability. In practice, this architecture covers all aspects of searchable model spaces used to organize reusable and transferable technical models within a given domain. It addresses key challenges in model sharing by ensuring that models, along with their relationships, versions, and variants, can be understood and exchanged across different organizations. The SmartEM architecture achieves this by combining proven model management concepts with widely accepted data-space standards (like the International Data Spaces framework) to foster a new ecosystem of engineering models. As a result, model store implementations based on this reference architecture are expected to demonstrate significant technical and commercial benefits in engineering workflows.

4.1. Overview of Reference Architecture

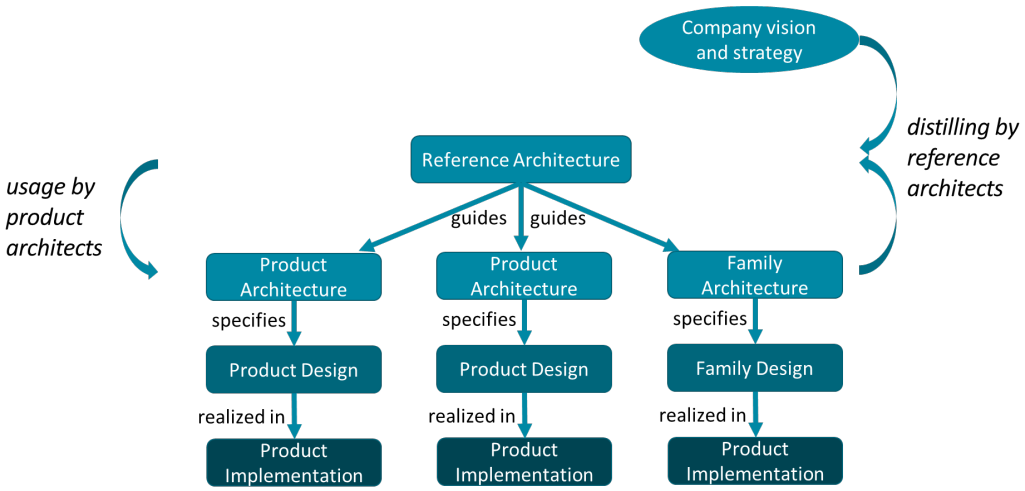

Figure 4.1: The role of a Reference Architecture in a Company

A reference architecture (RA) provides structured guidance for developing products or product families. In the SmartEM project this structure is mapped to model management as follows:

- Company vision and strategy: Model management vision and strategy (ecosystem or company level).
- Product architecture: Model management architecture.
- Product design: Model management tool design.
- Product implementation: Model management tool implementation.

4.1.1. Purpose of the Reference Architecture

Creating and documenting the RA serves several purposes:

- Alignment across teams, projects, organisations, companies, and the wider ecosystem.
- Clarity on relevant aspects, including the ability to justify why particular choices and structures exist, and to gain insight by stepping back from low-level implementation details.
- Easier communication.
- Guidance for establishing a model management mindset within an organisation.

- Direction on applying the RA to derive a concrete model management architecture in context.

For SmartEM, positioning project members against the concerns represented in the RA improves communication and exposes coverage gaps, for example missing work on model-licence contract standards.

4.1.2. Scope

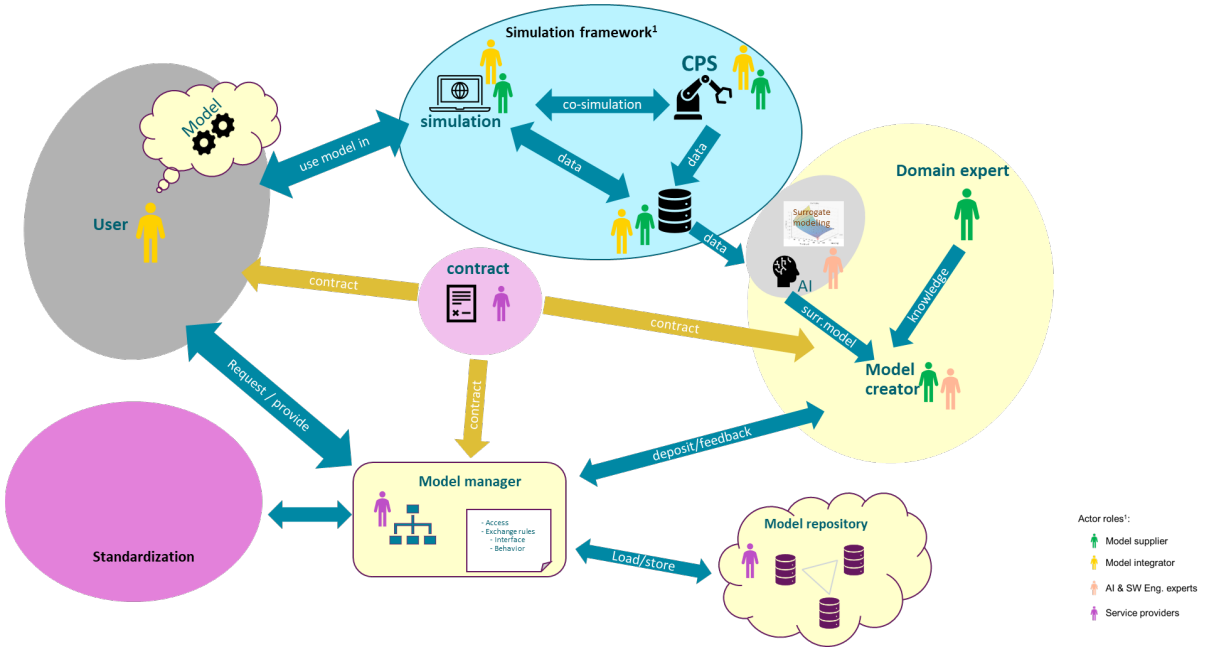

Figure 4.2: Rich Picture

The Figure 4.2 Rich Picture offers a clear starting point for defining and discussing SmartEM's scope. Because the scope has nested layers, a hierarchical perspective helps clarify the key dimensions, who, what, how, and when, across these areas:

- High-tech industry that creates products
- Engineering departments within companies
- The development phase as the primary focus
- Engineering and analytical models
- Reuse of existing models

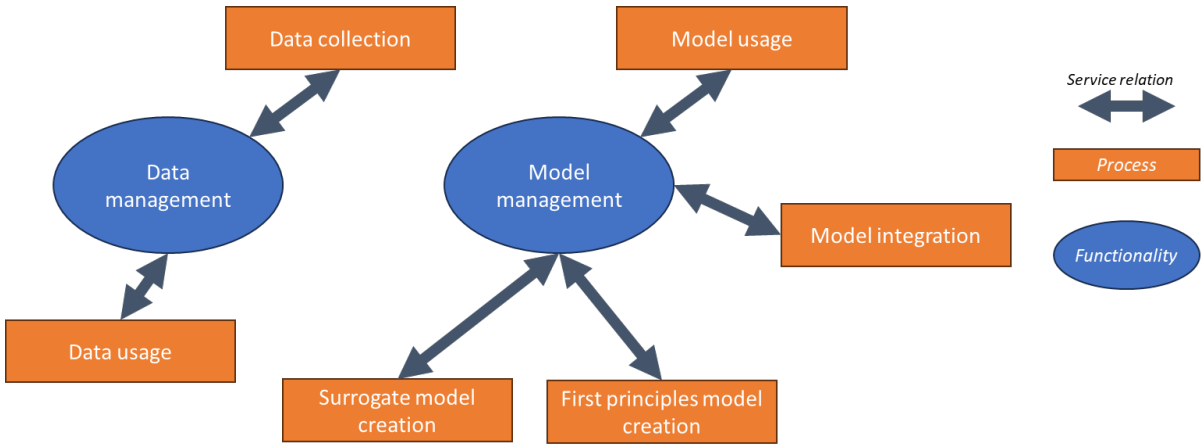

Figure 4.3: Scope Diagram Showing Functionalities and Processes Related to Data and Modelling

Figure 4.3 illustrates the system of interest, a model management system that orchestrates the full life-cycle of engineering models.

Core functionality: It brokers data collection ↔ usage, drives model creation (validation, calibration, surrogate or first-principles), governs integration, and supports model evolution over time.

Process relationships: Data management supplies inputs to the model manager; created models are integrated and used in applications, generating fresh data that loops back for continuous improvement.

Scope fit: This end-to-end flow, shown in *Figure 4.3*, aligns with the project's aim of an open reference architecture for smart engineering model spaces, confirming that the proposed RA boundary is appropriate.

4.2. High-Level Reference Architecture

At a high level, the SmartEM reference architecture adopts a federated data-space approach for model sharing rather than relying on a single central repository. Each participating organization retains its own Model Store (a repository of models and their metadata) and connects to others through standardised interfaces. An IDS Connector is deployed by each party as a gateway to the data space, enabling secure exchange of models and metadata. In an International Data Spaces (IDS) network, all data exchange happens through such connectors, every IDS Connector exposes data endpoints for others, eliminating the need for any central data hub [24]. This means each model remains under the provider's control (ensuring data sovereignty), and only authorized exchanges occur peer-to-peer via the connectors. To allow participants to discover available models across the federated network, the architecture includes a Metadata Broker service. The Metadata Broker acts as a searchable registry or "phone book" for the data space: providers (via their connectors) publish descriptive metadata (self-descriptions) of their models to the broker, and consumers query the broker to find relevant models [25]. Importantly, the broker holds only metadata, not the actual models, so actual model transfer is negotiated directly between the provider's and consumer's connectors. The use of common ontologies and metadata standards in SmartEM ensures that model descriptions are semantically interoperable and searchable across different sources [26]. Overall, this high-level architecture leverages standard IDS components to create a secure, interoperable, and scalable model-sharing environment where each stakeholder can publish, find, and access models on-demand while respecting intellectual property and usage agreements.

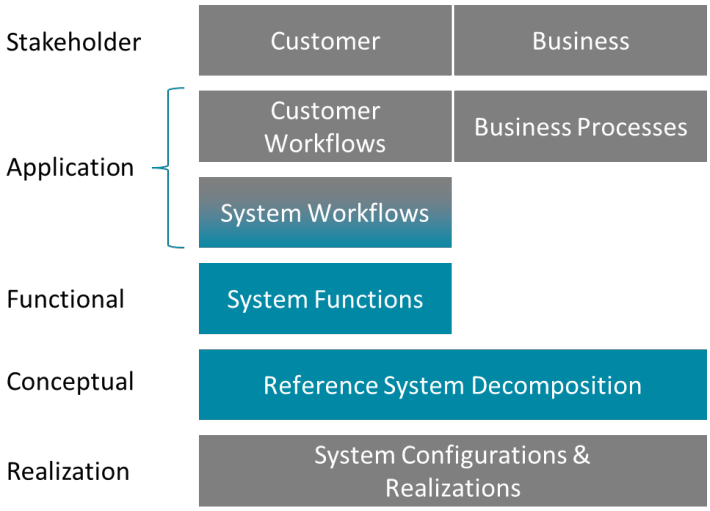

Figure 4.4: Reference Architecture Structure

Figure 4.4 outlines the backbone of SmartEM's reference architecture, cascading from stakeholder views to concrete implementations.

- Stakeholders: Customer and business perspectives define high-level needs.
- Workflows and processes: Split into customer workflows and supporting business processes, which map directly to system workflows handled by the model-management platform.
- Functions: System functions translate each workflow into discrete capabilities.
- Abstract system decomposition: A logical architecture arranges those functions into interoperable modules.
- Realisations: Specific configurations and building blocks deliver the architecture in practice.

Explicit links between every adjacent layer ensure traceability; the DAARIUS tool records and visualises these relations, allowing designers to justify each design choice and to propagate changes consistently across the stack.

Applied to SmartEM we propose to focus on the Model Manager / Model Management System. The systems-of-interest we want architectures for, are model management systems. Shown in the *Figure 4.5*.

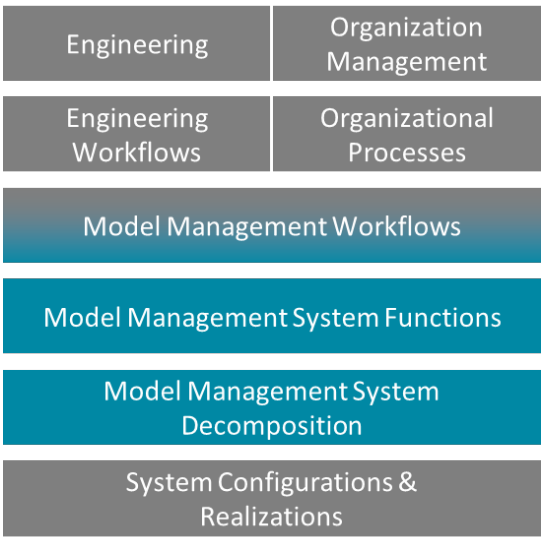

Figure 4.5: SmartEM Reference Architecture for the Model Management System

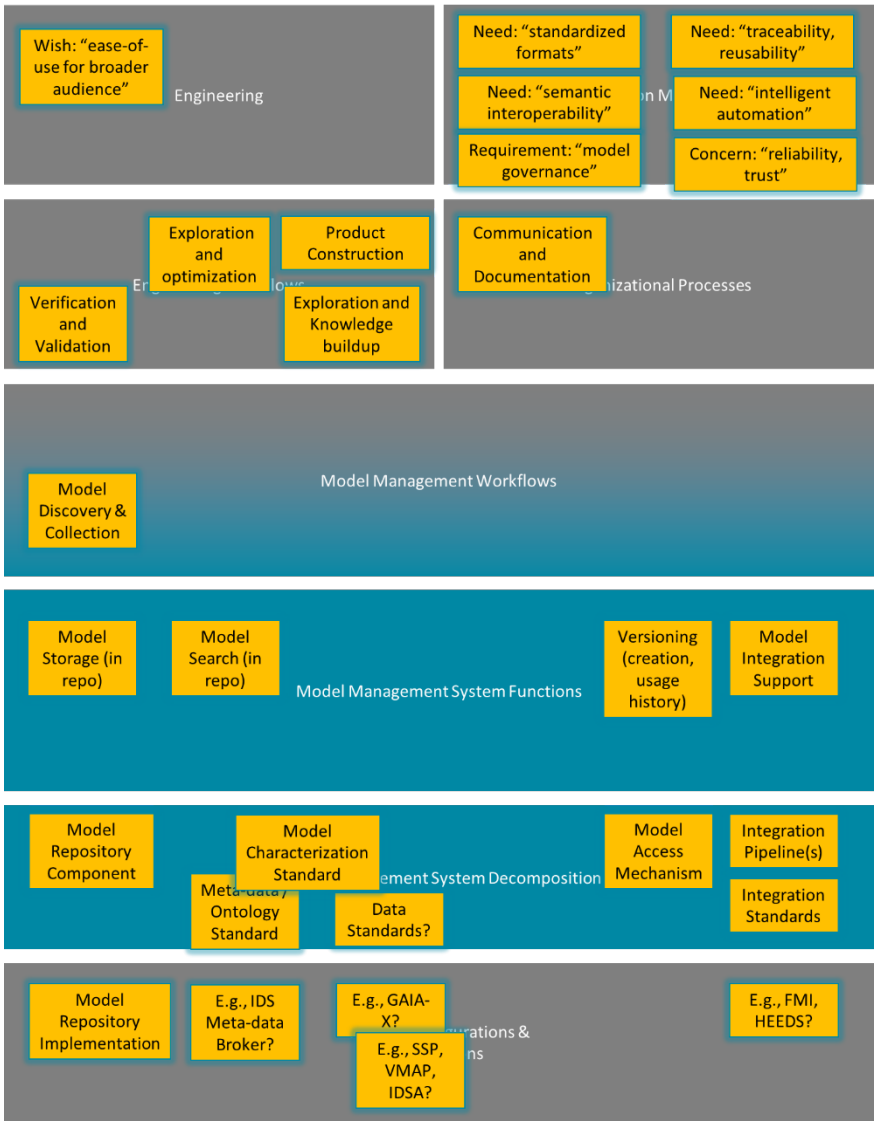

Figure 4.6: Initial set of SmartEM reference architecture elements

The Figure 4.6's layers explained in above. For yellow parts:

- Stakeholder layer, high-level needs
  - Ease of use for a broader audience (engineering view).
  - Standardised formats and semantic interoperability.
  - Model governance, intelligent automation, traceability, reusability, reliability and trust (business view).
- Application layer, engineering vs organisational concerns
  - Engineering workflows:
    - Verification and validation.
    - Exploration & optimisation.
    - Product construction.
    - Exploration and knowledge build-up.
  - Organisational processes: communication and documentation.
- Functional layer, core capabilities of a model-management platform
  - Model discovery & collection workflow.
  - System functions:
    - Model storage (repository).
    - Model search.
    - Versioning (creation and usage history).

    - Model integration support.
- Conceptual layer, logical decomposition
    - Model repository component.
    - Model characterisation standard (metadata, ontology).
    - Model access mechanism.
    - Integration pipelines and related standards (e.g. FMI, HEEDS).
    - Data and integration standards placeholders for future alignment (e.g. GAIA-X, IDS-based).
- Realisation layer, potential technology choices
    - Concrete repository implementations.
    - IDS-style metadata brokers.
    - Data-space frameworks such as GAIA-X, SSP, VMAP, IDSA.
    - Integration toolchains (e.g. FMI, HEEDS) for simulation coupling.

*Figure **4.6*** maps stakeholder wishes down to concrete building blocks, making visible how SmartEM's model-management vision translates from user needs and workflows to implementable components and technology options.

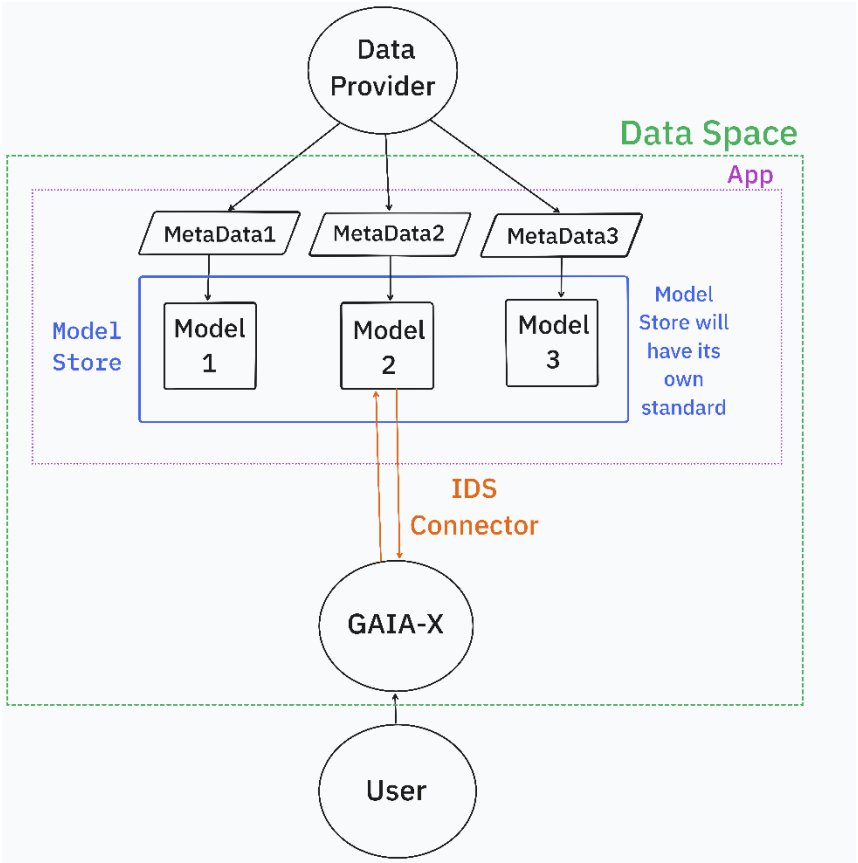

Figure 4.7: High-Level Reference Architecture Diagram

The *Figure **4.7*** shows how models and their metadata are exposed within a data, space context. A Data Provider supplies information that is represented as multiple metadata entries (MetaData1, MetaData2, MetaData3). Each metadata entry points to a corresponding model (Model 1, Model 2, Model 3) stored in a Model Store. The Model Store resides inside a Data Space application boundary and follows its own internal standard. An IDS Connector links the Model Store to GAIA-X, enabling federated discovery and controlled exchange of model resources. A User accesses models via GAIA-X, benefiting from the interoperable, standards, aligned structure defined by the RA.

4.2.1. Key Components and Their Roles

Table 4.1: Key Components and Their Roles [24],[26]

Component	Role in the SmartEM ecosystem
Model Store	Core repository for surrogate and other engineering models. It stores and indexes models, controls versions, and exposes interfaces for adding or retrieving artefacts. Implementations follow common RA guidelines so that stores interoperate across organisations. Internally, a Model Store may offer APIs or user interfaces for governance, search, and updates, and typically works with an IDS Connector to publish models to the data space.
IDS Connector	Trusted gateway linking a Model Store to the wider SmartEM network, based on International Data Spaces (IDS) standards. It handles secure communication, access control, and protocol translation, exposing models or related services via standard endpoints. Each connector enforces usage policies so that data exchange respects contractual and security requirements, thereby preserving data sovereignty while enabling direct provider-to-consumer transfers.
Metadata Broker	Directory service for model discovery. Implemented as an extended IDS Connector, the broker registers and answers queries on model Self-Descriptions (e.g., RDF/JSON-LD) without holding model files themselves. Participants use it to find models by type, domain, or capability before negotiating access through the provider's connector. This design separates discovery from transfer and supports semantic search across Model Stores.

These three components shown in the *Table **4.1*** work together as a distributed model marketplace: the Model Store manages and serves models, the IDS Connector provides the secure conduit for sharing, and the Metadata Broker enables global look-up. Collectively they realise SmartEM's vision of moving from traditional data spaces towards interoperable, secure, and well-governed model spaces, while the overarching RA ensures strategic alignment and consistent implementation throughout the ecosystem.

4.3. Centralized vs. Federated Approaches

A model store manages machine-learning models and their metadata. It can be centralised, with every model in one repository, or federated, with separate repositories per team or site that may share data. The *Table **4.2*** shows the comparison.

Table 4.2: Centralized vs Federated

Factor	Centralized	Federated (Distributed)
Performance	A single server offers very quick responses when users and applications are near the repository, because every request is handled locally. As traffic grows or users connect from distant regions, the same server becomes a bottleneck, and remote requests suffer higher round-trip times.	Each node answers local requests quickly, which is ideal for latency-sensitive workloads at branch offices or edge devices. Coordinating across nodes adds network chatter, so global searches or metadata updates can take longer than in a central system.

Scalability	Extra capacity is gained mainly by upgrading hardware or clustering that one system, which soon becomes costly and technically complex. If demand spikes beyond the limits of the central machine, the whole service can slow or fail.	Capacity grows almost linearly by adding more repositories where needed, spreading load and avoiding a single choke point. The challenge is keeping model metadata consistent so that users can still discover models held at other sites.
Cost	Consolidating all models in one place removes duplicated storage and simplifies licensing and maintenance. The drawback is a significant upfront investment in a robust platform and further expense whenever a larger server or premium cloud tier is required.	Teams can begin with modest local hardware or cloud instances, avoiding a large central purchase and paying only as they expand. Over time, multiple sites and possible data replication drive aggregate storage and administration costs higher than a single installation.
Security	One security perimeter makes it straightforward to apply consistent access controls and auditing. The trade-off is a single high-value target, so a breach or outage at the core store can expose every model or interrupt service for all teams.	Data remains in its region of origin, which limits the impact of a breach and helps satisfy data sovereignty rules. However, every node needs comparable hardening, monitoring and patching, and uneven practice at one site can weaken the whole network.
Operations	Administrators manage only one deployment, making backup, patching and governance processes simple and uniform. High availability arrangements, such as replication or fail-over clusters, are critical because the business depends on that single source of truth.	Running many repositories calls for strong DevOps automation, federation services and clear governance processes to catalogue, version and approve models across sites. With good tooling the system is resilient, because a fault at one location does not halt work elsewhere.
Best suited for	Organisations operating mainly in one region or on a modest scale, or those that value ease of governance above-edge performance. Centralisation also helps a small data science team enforce standards and track all production models in one catalogue.	Large or globally distributed enterprises that need regional autonomy, strict locality rules or very low-latency inference near the point of use. It is also attractive where business units own their own infrastructure and prefer to scale independently yet still collaborate through federation.

Hybrid strategy

Keep a main registry for global oversight and add local stores for speed and autonomy. This blends central governance with regional performance but inherits complexity from both, so careful design is essential.

Conclusion

Centralised stores give simple upkeep and unified control, fitting moderate scales and strict governance. Federated stores scale out, resist faults and keep data local, fitting large or

dispersed enterprises but adding management effort. Each organisation should match these traits to its priorities in performance, cost, security and complexity when choosing a model store [27] , [32].

4.4. Security, Authorization, and IP Management

Create and maintain a Data Management Plan covering data types, retention, and deletion. Ensure compliance with regulations (GDPR, CCPA, etc.) and map data flows. Implement encryption, network segmentation, and regular security testing. Apply anonymization and pseudonymization techniques to balance privacy and utility.

- Access Rights
  - Use Role-Based Access Control to assign permissions for data and model operations.
  - Enforce strong authentication (MFA) and integrate identity providers.
  - Maintain immutable audit logs and real-time monitoring for anomalous behaviour.
  - Define onboarding/offboarding workflows and conduct periodic privilege reviews.
- Intellectual Property
  - Classify artifacts into open-source, proprietary, or third-party components with tagged licenses.
  - Choose appropriate licensing models (MIT/Apache, custom, or dual licensing).
  - Navigate patent filings, copyright registrations, and branding trademarks.
  - Include IP clauses in collaboration agreements with clear publication and embargo policies.
- Engineering Model Stores
  - Architect secure, containerized registries with TLS and role separation.
  - Implement semantic versioning and full data-to-model lineage tracking.
  - Adopt metadata standards like Model Cards and Datasheets for transparency.
  - Automate CI/CD pipelines with validation, fairness testing, and deployment safeguards.
  - Scale serving infrastructure with canary releases, A/B testing, and drift detection.
- Implementation Plan
  - Assign roles: Data Protection Officer, MLOps engineers, legal counsel.
  - Follow a six-month roadmap: governance setup, module development, IP policy drafting, pilot and audit.
  - Schedule quarterly reviews to refine policies based on new regulations and community best practices.
- Insights
  - Explore privacy-enhancing technologies (homomorphic encryption, secure multiparty computation).
  - Leverage federated learning to keep data on-premise while sharing model improvements.
  - Integrate behavioural analytics for insider-threat detection.
  - Tie into MLOps platforms (MLflow, Seldon) for end-to-end traceability.
  - Embed ethical AI practices: environmental impact assessments and bias mitigation.

4.5. Model Sharing Scenarios

The purpose of sharing models is to take advantage of earlier efforts. These advantages may be to reduce the effort of creating models again and again or may be the easy spreading of knowledge across a community, or to expand existing knowledge. These are all valid reasons

to make models available to others, but they strongly depend on the situation of the model creator and potential model user.

There are some key questions determining the sharing scenario.

Level of Abstraction in Reuse

When considering the model on its conceptual elements, e.g. which functional parts are there, what is the meaning of certain model concepts, and how are they described (e.g., formulas), which calculation strategies are used (e.g., internal data flow setup), the abstraction level is high. This is mainly the reuse of knowledge that is embedded in the model (and its way of application). On the other hand, the reuse of an implementation/code (e.g., python code) is at a low abstraction level. The purpose of reuse can be limited to the reduction of effort. For example, the Fast Fourier Transform algorithm is hardly ever coded from scratch but copied and reused from public code libraries.

Difficulty of Gaining Sufficient Knowledge

In order to successfully acquire knowledge, reuse a model, or expand existing knowledge, it is crucial to understand the descriptions of the shared model and its meta-data. In industrial practice we hardly see reuse of models without the creator, or recent maintainers who have the knowledge present or nearby. Apparently, the model descriptions are insufficient or lacking. Therefore, with increasing ‘distance’ (direct neighbour-colleague in the team, department, unit, site, company, supplier, public) this problem becomes worse. The capabilities of the potential model user is of course critical: is there a knowledge gap? Depending on the available knowledge of the domain, company-specific knowledge, or even specific model knowledge, reuse is easier.

Confidentiality Considerations

In Section 4.4 this topic is already addressed. In industrial practice IP protection can prohibit easy reuse, even within one company (in the case of company secret models). Sharing these secret models may be a challenge, as it is difficult to screen the internals at various levels: concepts, API, code.

The large variety of sharing scenarios can be distinguished by partner types:

A. Intra-partner sharing: sharing within one organisation
B. Inter-partner federation: sharing across organisations
C. public discovery: sharing to anyone

Some typical industrial scenarios occurring in practice are described here.

Intra-partner sharing:

- E.g. reuse of a model while creators are nearby. The closeness guarantees correct and effective, and efficient reuse of the model. Often creators/maintainers perform the analysis or execution as an assignment (formally or informally arranged).
- E.g. reuse of a model implementation from a company-internal source. A lot of knowledge has to be gained by asking around, experimenting with the model, etc.
- E.g. reuse of a conceptual model, by studying the accompanying documentation and meta-data. The intended application is slightly different, therefore direct implementation reuse is not possible.

Inter-partner federation:

- E.g. Companies collaborating via a dedicated contract. A supplier needs to understand the physics of a user's problem in order to create or adapt a model of a system component. Models are being shared to solve a problem, often as part of the total solution package (consisting of e.g., investigation service, developed dedicated models, solution SW, and solution HW).
- E.g. A 'modelling company' working as a service provider, creating and 'selling' generic models. Models are sold or licensed for use.

Public discovery:

- E.g. The scientific literature is used for finding and reusing models. The sharing of models is enabled via links to databases and other sources in the papers. This requires often significant study and adaptation of implementations.
- E.g. There are open-source models. These are given freely to the public. The question from an industrial perspective is how trustworthy these models are. Also, what is the business case and who is behind them? Is a government paying?

5. Key Building Blocks of a Model Store

5.1. Functional Building Blocks (Model Ingestion, Query, Validation, Versioning)

Model stores are needed for managing the lifecycle of models and are the central repository for teams to store, discover and deploy models.

The first key building block is Model Ingestion, which involves the operation of model registration. This process includes adding a new model to the model store and ensuring that it has proper metadata. The actual model files (e.g., ONNX, HDF5, TensorFlow SavedModel) have links to their storage locations and record essential information about the model, such as description, author, dependencies, and tags/labels. This process must be governed by security and access control mechanisms that define who can ingest models and with what permissions. Support for both manual and automatic workflows should be available, and integration with CI/CD pipelines will make it easier for developers to manage model ingestion efficiently.

Querying within a model store enables users to search for, retrieve, and discover models. Users can query models based on various metadata attributes and fetch the associated model artifacts. It is also crucial to retrieve and understand the software dependencies required to run and use a specific model. Exploring the available models should be facilitated through intuitive user interfaces and accessible APIs, allowing users to efficiently navigate and utilize the model store.

Validation ensures quality, integrity, and operational readiness of the models in the model store. Core aspects for validation mean:

- This requires schema validation to verify that the ingest models expected input/output schema matches expectations.
- Artifact integrity checks ensure the model file is not corrupted and can be uses by its intended framework.
- Storing and allowing queries on key performance metrics such as accuracy, precision etc.
- Checking for known vulnerabilities in the model and its dependencies, ensuring adherence to organizational or regulatory policies.

These aspects could be run in an automated pre-deployment validation pipeline that automatically runs schema checks, tests, other evaluation scripts.

The last functional block, versioning, is essential for managing the evolution of models and ensuring traceability. The use of semantic versioning, common in software APIs and libraries, can be similarly applied to models to indicate the significance of changes. Models can be labelled with tags such as production, experimental, deprecated, etc., to help track their status and facilitate lifecycle management.

5.2. Technical Building Blocks

The technological components are based on Ontology based indexing and retrieval system with the following building block components:

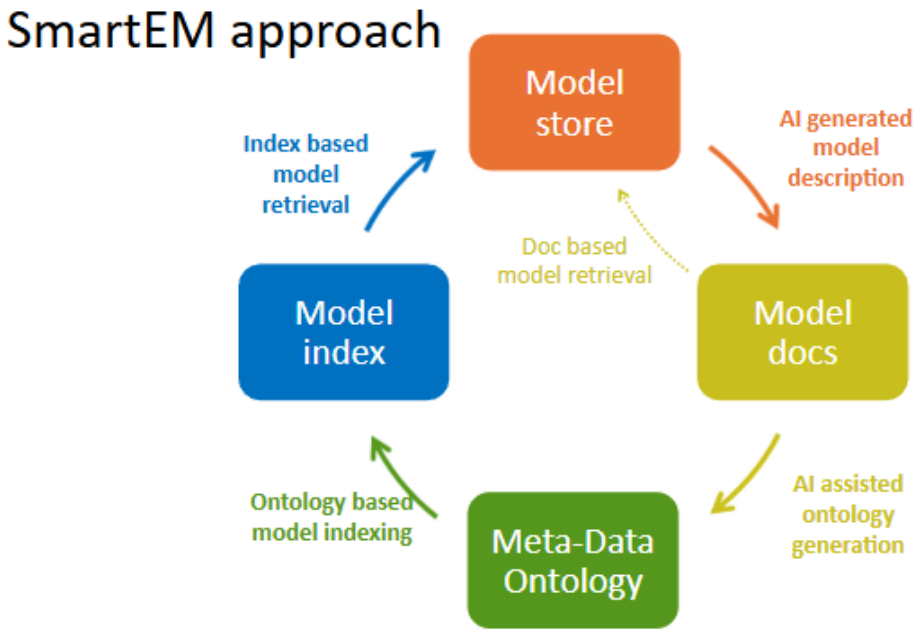

Figure 5.1: Technological components

5.2.1. Ontology-Based Indexing and Retrieval System

The system architecture is centered around an ontology-based indexing and retrieval framework, designed to enhance semantic understanding and improve discoverability across complex model repositories. This approach leverages structured metadata and AI-driven processes to deliver transparent, explainable, and highly integrable search capabilities.

Key Technological Components

The system is composed of modular building blocks that work in concert to support ontology-driven indexing and retrieval:

- **AI-Assisted Ontology Development and Maintenance**: Machine learning models assist in generating and refining domain ontologies, reducing manual effort and ensuring consistency.
- **Explicit and Editable Metadata Ontology**: Users can define and modify metadata structures, enabling tailored semantic representations aligned with specific use cases.
- **Automated Model Indexing from Generated Documentation**: AI models automatically extract and structure information from model code and documentation, streamlining the indexing process.
- **Transparent and Explainable Retrieval**: The system provides traceable search results, allowing users to understand the semantic reasoning behind each retrieval.
- **Seamless Integration with External Model Repositories**: Designed for interoperability, the system easily connects with third-party model stores and documentation sources.

Dual Technical Approach

The architecture follows two primary technical workflows:

Model Store to Model Documentation

This pipeline involves training dedicated AI models to generate natural language transcriptions of source code. It supports multiple levels of aggregation:

- **Line-by-Line Descriptions**
- **Function-Level Summaries**
- **Package-Level Overviews**

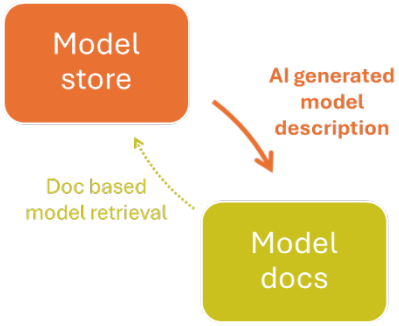

Figure 5.2: Model store to Model docs

These transcriptions incorporate existing documentation, images, and schematics to create rich, searchable content. Base models such as *CodeLlama*, *ex2*, and *ex3* are employed to ensure high-quality outputs. The result is a comprehensive documentation layer that enables semantic search across model artifacts.

Model Documentation to Metadata Ontology

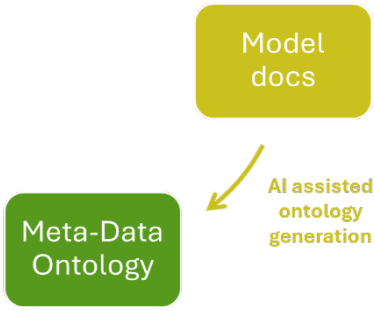

Figure 5.3: Model docs to Meta-Data Ontology

Once documentation is generated, it is parsed and mapped to a structured metadata ontology. This step transforms unstructured text into semantically indexed entities, enabling ontology-based retrieval and reasoning. The editable nature of the ontology allows for continuous refinement and alignment with evolving domain requirements.

Key Benefits:

- AI assisted ontology development and maintenance
- Explicit and editable meta-data ontology
- Automated model indexing from generated docs
- Transparent and explainable retrieval
- External model repo integration is easy

5.3. Operational and Organizational Building Blocks (User Roles, Deployment Layers, Access Policies)

Defining the organizational roles in context of building a model store are needed to ensure effective management and usage. Clear roles and responsibilities help streamline operations, maintain security, and optimize the lifecycle of models. With organization groups defined, this helps to further identify and categorize role members as users and that can be assigned to appropriate groups:

Key roles can be summarized as follows:

Data Scientists: Key group that will develop, training and validate the models. They will also ensure proper metadata is attached to the metadata. They will collaborate with other teams and continue to improve and refine the models.

Administrators: They will manage the model store infrastructure and implement and enforce security and access control policies. They will also monitor the model store's performance and address issues.

IT and Dev Ops Teams: Their responsibilities include ensuring the model's store scalability and reliability. They will support the data scientists with CI/CD pipelines for model deployment and manage the backups and system maintenance.

As data and models contained within the model store may contain restricted or confidential information, granular access control and logging needs to be ensured to guarantee compliance of the store.

Access control can be implemented through a combination of users, groups and access rules.

Users: Each user of the model store will be assigned a unique username linking back to the user and their organization to the desired level of detail (e.g. department, e-mail, phone number…).

Groups: Groups represent sets of users (and potentially other groups) that share a common property within the system. Standard groups can exist to identify system administrators, while custom groups can be created as required (e.g. a group to identify a single company, department, team…). Further company-specific organizational roles can be replicated through groups.

Access rules: Access rights are defined per item (model, dataset …) within the data store and are assigned to either an individual user, a group or a combination. Access rights define if a user/group has the right to perform certain actions on the item, for example reading, writing or executing the model. Further granularity can be added as required.
A single item can have multiple rules defined for different users or groups. In case of conflicting/overlapping rules, a logical IF is applied to determine which rights apply. (E.g. a user has "write" privileges, while a group to which that user belongs only has "read" privileges. The logical IF ensures the user has the proper "write" privileges). Certain automatic access rules can be enforced, such as enabling full access to the creator of the item.

As each user within the system is identified by its username, every action performed can be linked to a specific user and logging can be done to ensure traceability.

Deployment structures help define how the models are deployed from the model store to various environments. It is good practice to have separate environments for development, testing, and production (DTAP). The models need to be tested and validated before deployment. This includes work to ensure configurations and dependencies are correct for the models to function. Having a good CI/CD integration facilitates the automation and ensures that models are consistently moved from one environment to another, such as from development to production. Models need to pass validation checks and even performance metrics before deployment. Furthermore, version control is needed, and CI/CD integration can help manage model version and track changes to help enable roll back scenarios.

5.4. Governance and Trust Building Blocks (Usage Policies, Certification, Logs)

5.4.1. Executive Technical Summary

In SmartEM's federated architecture, governance and trust are critical enablers that make cross-organization model sharing and usage feasible, secure, and compliant. This governance layer is built on three intertwined pillars: usage policies, certification frameworks, and comprehensive logging for auditability.

5.4.2. Usage Policies and Automated Enforcement

SmartEM ensures that every surrogate model carries clearly defined, machine-readable usage policies embedded within its metadata. These policies govern aspects such as:

Purpose restrictions: Specifying if the model is limited to internal R&D, production optimization, or prohibited from resale.

Operational limits: like maximum allowed inference calls or time-bounded usage rights.

Jurisdictional/geofencing constraints: to comply with local data protection laws (e.g., GDPR within EU borders).

Requests to access or invoke a model are automatically checked against these policies by policy enforcement engines integrated into the federated connectors. This means non-compliant actions are blocked in real time, minimizing risks and contractual breaches without manual oversight.

5.4.3. Certification to Establish Technical and Regulatory Trust

Models in SmartEM can undergo multi-level certification processes that validate their technical robustness, security posture, and compliance with sector-specific standards. Certifications may include:

- Performance and reproducibility assessments, ensuring that claimed metrics hold under controlled benchmarks.
- Security validations, checking for vulnerabilities to adversarial manipulations or code exploits.
- Alignment with industry frameworks, such as ISO 27001 for information security or EU MDR for medical applications.

Certification artifacts are digitally attached to the model's metadata, allowing any partner to verify trust levels before integration into their operational workflows.

5.4.4. Transparent Logging and Cross-Partner Traceability

Every critical event, model upload, access, inference invocation, modification, or secure deletion, is recorded in immutable audit trails. Logs capture who did what, when, and under what context, supporting both internal governance needs and external regulatory audits. When workflows span multiple organizations, SmartEM ensures that relevant logs propagate securely across partners, creating a seamless, end-to-end audit chain without centralizing sensitive operational data.

5.4.5. Example Scenario

For instance, an automotive OEM using a crash simulation surrogate from a supplier would have usage policies that limit it to single-project R&D within approved network segments.

Certification assures both parties of the model's integrity and performance. Every usage is logged, and upon project completion, SmartEM governance components can enforce secure deletion, generating cryptographic proofs to demonstrate compliance.

5.4.6. Summary Statement

Through tightly integrated policy enforcement, certification mechanisms, and auditable logs, SmartEM builds a technical foundation that empowers stakeholders to share and consume high-value surrogate models confidently, while preserving intellectual property, complying with diverse regulatory regimes, and fostering long-term cross-border collaborations.

5.5. Alignment with Soft Infrastructure Design

The proposed SmartEM architecture aligns closely with the soft infrastructure principles defined by GAIA-X and IDSA, particularly in the areas of:

- Sovereignty
- Interoperability
- Trust
- Federation

At the core of SmartEM is the model ingestion process, which ensures that models are registered with rich metadata, including descriptions, authorship, dependencies, and semantic tags. This metadata serves as a foundation for GAIA-X self-descriptions, enabling models to be discoverable and interpretable across federated environments. The ingestion process is governed by granular access control policies, reflecting IDSA's usage control mechanisms and ensuring that only authorized entities can contribute models to the store.

The querying and discovery capabilities of SmartEM are built on an ontology-based indexing system, allowing users to search for models using semantic attributes. This approach supports GAIA-X's federated cataloguing and semantic interoperability, while also aligning with IDSA's broker services, which facilitate the discovery of data and services across distributed infrastructures. The use of AI for metadata extraction further enhances transparency and usability.

Validation mechanisms within SmartEM ensure the integrity, quality, and compliance of models before deployment. Automated pipelines perform schema checks, artifact integrity verification, and vulnerability scans, ensuring that models meet organizational and regulatory standards. These mechanisms contribute to the GAIA-X trust framework, which emphasizes service certification and operational readiness, and mirror IDSA's policy enforcement and auditability requirements.

To manage the evolution of models, SmartEM incorporates robust versioning strategies, including semantic versioning and lifecycle tagging (e.g., production, experimental, deprecated). This supports GAIA-X's transparency and lifecycle management principles, while also enabling IDSA-compliant traceability of model changes and usage history.

The architecture also defines clear roles and access control structures, with responsibilities distributed across data scientists, administrators, and IT/DevOps teams. Each user is uniquely identified and linked to organizational metadata, enabling fine-grained access control and logging. This setup aligns with GAIA-X's identity and access management (IAM) framework and supports IDSA's security profiles and accountability mechanisms.

SmartEM's deployment structures facilitate the movement of models across development, testing, and production environments, supported by CI/CD pipelines. These pipelines enforce validation checks and version control, ensuring that models are reliably and securely deployed. This reflects GAIA-X's emphasis on service portability and automation and supports IDSA's runtime policy enforcement.

Finally, the technical building blocks of SmartEM, particularly its ontology-driven metadata management and AI-assisted documentation generation, enable seamless integration with

external repositories and support transparent, explainable retrieval. These components are directly aligned with GAIA-X's semantic interoperability goals and IDSA's information model, providing a scalable and extensible foundation for federated model management.

In summary, SmartEM embodies the soft infrastructure principles of sovereignty, interoperability, trust, and federation as defined by GAIA-X and IDSA, offering a robust and future-proof architecture for managing engineering models in distributed, multi-stakeholder environments.

6. Technical and Functional Requirements

The model store structures examined in Task 5.1 should be designed in such a way that each partner can create its own model store in the SmartEM project. In this direction, it is aimed to create a decentralized, secure and interoperable system structure. In order to achieve this goal, the following technical and functional requirements should be taken into consideration.

6.1. Decentralization

The model stores must be capable of operating independently within each partner's infrastructure, without relying solely on a central server. This approach enhances data sovereignty and forms the foundation of a federated architecture compatible with data space frameworks.

6.2. Scalability

The system should be designed to accommodate the addition of new partners, models, and data sources. Scalability must align with frameworks such as the GAIA-X Federated Catalogue and the IDSA Reference Architecture to ensure long-term sustainability and adoption.

6.3. Collaboration Support

The platform must support collaborative features such as model sharing, co-editing, and version control across multiple stakeholders. Connector components in the IDSA architecture offer an effective example of such functionality.

6.4. Interoperability (Data Formats, APIs, Semantic Alignment)

Model stores must be able to integrate seamlessly through shared data structures, semantic metadata, and standardized APIs. This capability should be interoperable with synchronization frameworks such as the IDSA Metadata Broker and the GAIA-X ecosystem.

6.5. Compatibility

The model store should be compatible with diverse software platforms, data formats, and system infrastructures. Employing semantic web technologies, open data models, and standardized model store schemas is critical to easing integration efforts. Organizations like DSSC and IDSA play a pivotal role in defining and guiding relevant standards.

6.6. Trust Management

The system must implement robust mechanisms for data security, user authentication, and access control. The security and authorization components provided by IDSA are fundamental in addressing this requirement.

6.7. Auditability

All actions related to model creation, modification, and usage must be logged with detailed metadata, ensuring full traceability. This aligns with metadata transparency and traceability principles defined in the GAIA-X Catalogue and IDSA's security components.

6.8. Federation

The system should enable model stores hosted by different partners to interoperate within a unified ecosystem. This requirement is supported by the federated architecture principles promoted by IDSA and data space initiatives.

7. Conclusion

This deliverable has presented the current state-of-the-art, design principles, and guidelines for the development of SmartEM model stores. Building on established open-source data space components and relevant industrial initiatives, it has defined the fundamental building blocks required to implement a secure, interoperable, and application-neutral reference architecture.

The work carried out in D5.1 has identified key technical and organisational challenges such as interoperability, scalability, governance, IP management, and security, and has outlined strategies to address them. The proposed architecture and associated guidelines form a solid basis for subsequent implementation activities within WP5, ensuring that the model store concept can be effectively realised and integrated with other SmartEM components.

The outcomes of this task will directly support the development of both use case-specific and general-purpose model store implementations, enabling trusted and efficient model exchange across organisational boundaries. The work described here provides an important foundation for the practical demonstrations and validations planned in later project stages, ultimately contributing to the overall goal of fostering a sustainable, collaborative ecosystem for engineering model sharing within SmartEM and beyond.

8. References

[1] Turkmayali, A., & Gras, N. (2024). Making the Dataspace Protocol an international standard (1.0). Zenodo. https://doi.org/10.5281/zenodo.12663036

[2] Gaia-X, "Gaia-X explained", [Online]. Available: https://gaia-x,hub.de/en/gaia-x-explained/ [Accessed: 28-Jul-2025]

[3] Modelica, [Online]. Available: https://modelica.org [Accessed: 7-Aug-2025]

[4] OpenModelica, [Online]. Available: https://openmodelica.org [Accessed: 7-Aug-2025]

[5] Dymola, [Online]. Available: https://www.3ds.com/products/catia/dymola [Accessed: 7-Aug-2025]

[6] Synera, [Online]. Available: https://www.synera.io [Accessed: 7-Aug-2025]

[7] Simcenter HEEDS, [Online]. Available: https://plm.sw.siemens.com/en,US/simcenter/integration,solutions/heeds/ [Accessed: 7-Aug-2025]

[8] Aras Innovator, [Online]. Available: https://aras.com [Accessed: 7-Aug-2025]

[9] OpenMBEE, [Online]. Available: https://www.openmbee.org [Accessed: 7-Aug-2025]

[10] OpenMETA, [Online]. Available: https://openmeta.metamorphsoftware.com [Accessed: 7-Aug-2025]

[11] Maplesoft, Maple - The Essential Tool for Mathematics, [Online]. Available: https://www.maplesoft.com/products/Maple/ [Accessed: 26-Aug-2025]

[12] Maplesoft, MapleSim - Advanced System-Level Modeling and Simulation, [Online]. Available: https://www.maplesoft.com/products/maplesim/ [Accessed: 26-Aug-2025]

[13] GibHub, [Online]. Available: https://github.com [Accessed: 7-Aug-2025]

[14] Zenodo, [Online]. Available: https://zenodo.org [Accessed: 7-Aug-2025]

[15] DVC, [Online]. Available: https://dvc.org [Accessed: 7-Aug-2025]

[16] MLFlow, [Online]. Available: http://mlflow.org [Accessed: 7-Aug-2025]

[17] Seldon Core, [Online]. Available: https://www.seldon.io [Accessed: 7-Aug-2025]

[18] Hugging Face Hub, [Online]. Available: https://huggingface.co/docs/hub/en/index [Accessed: 7-Aug-2025].

[19] Centena,X, [Online]. Available: https://catena,x.net [Accessed: 7-Aug-2025]

[20] Eclipse EDC, [Online]. Available: https://projects.eclipse.org/projects/technology.edc [Accessed: 7-Aug-2025]

[21] International Data Spaces Association, *Design Principles for Data Spaces* [Position Paper], 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5244997

[22] BDVA, *Discussion Paper – Data Sharing Spaces and Interoperability*, Dec. 2023

[23] StandICT.eu, *Landscape of Ontologies Standards*, ver. 1.0, 202

[24] International Data Spaces Association, "3.5.2 IDS Connector," IDS Reference Architecture Model 4.0, [Online]. Available: https://docs.internationaldataspaces.org/ids,knowledgebase/ids,ram,4/layers,of,the,reference,architecture,model/3,layers,of,the,reference,architecture,model/3_5_0_system_layer/3_5_2_ids_connector [Accessed: 21-Jul-2025]

[25] International Data Spaces Association, "3.5.4 Metadata Broker," *IDS Reference Architecture Model 4.0*, [Online]. Available: https://docs.internationaldataspaces.org/ids,knowledgebase/ids,ram,4/layers,of,the,reference,architecture,model/3,layers,of,the,reference,architecture,model/3_5_0_system_layer/3_5_4_metadata_broker [Accessed: 22-Jul-2025]

[26] Smartem Store, "The reference architecture will provide digital twin space communities," *Smartem Store* [Online]. Available: https://smartem.store/#:~:text=The%20reference%20architecture%20will%20provide,space%20communities [Accessed: 22-Jul-2025]

[27] Diligent Corporation, “Centralized vs. Distributed Databases: Which is Better for Governance?” Diligent Insights, Oct. 17, 2022. [Online]. Available: https://www.diligent.com/resources/blog/centralized,vs,distributed,databases [Accessed: 23-Jul-2025]

[28] K. J. O’Connell, “Centralized vs. Decentralized Data Management: A Side,by,Side Look,” Medium, Aug. 23, 2022. [Online]. Available: https://datacated.medium.com/centralized,vs,decentralized,data,management,a,side,by,side,look,063ae18839c9 [Accessed: 23-Jul-2025]

[29] Simbo.AI, “Understanding the Differences Between Centralized and Federated Models in Health Information Exchange,” Simbo.AI Blog, Oct. 25, 2023. [Online]. Available: https://www.simbo.ai/blog/understanding,the,differences,between,centralized,and,federated,models,in,health,information,exchange,3745986/ [Accessed: 23-Jul-2025]

[30] Amazon Web Services, “OPA Design Comparison,” AWS Prescriptive Guidance, 2024. [Online]. Available: https://docs.aws.amazon.com/prescriptive,guidance/lat (Turkmayali, 2024)est/saas,multitenant,api,access,authorization/opa,design,comparison.html [Accessed: 23-Jul-2025]

[31] GeeksforGeeks, “Comparison Centralized, Decentralized and Distributed Systems,” GeeksforGeeks, 2023. [Online]. Available: https://www.geeksforgeeks.org/system,design/comparison,centralized,decentralized,and,distributed,systems/ [Accessed: 23-Jul-2025]

[32] Amazon Web Services, “Centralize Model Governance With SageMaker Model Registry + Resource Access Manager Sharing,” *AWS Machine Learning Blog*, 2024. [Online]. Available: https://aws.amazon.com/tr/blogs/machine,learning/centralize,model,governance,with,sagemaker,model,registry,resource,access,manager,sharing/ [Accessed: 23-Jul-2025]