DiMAT

D3.2 DIMAT ARCHITECTURE V2

15/11/2024

Grant Agreement No.: 101091496
Call: HORIZON-CL4-2022-RESILIENCE-01
Topic: HORIZON-CL4-2022-RESILIENCE-01-25
Type of action: HORIZON Innovation Actions

D3.2 DIMAT ARCHITECTURE V2

Grant agreement number	101091496	Acronym	DiMAT
Full title	Digital Modelling and Simulation for Design, Processing and Manufacturing of Advanced Materials		
Start date	01/01/2023	Duration	36 months
Project url	HTTPS://CORDIS.EUROPA.EU/PROJECT/ID/101091496		
Work package	WP3 - DESIGN: DiMAT Framework Design		
Deliverable	D3.2 – DiMAT Architecture v2		
Task	T3.1 – DiMAT Architecture		
Due date	30/09/2023		
Submission date	15/11/2024		
Nature	Report	Dissemination level	Public
Deliverable lead	3-Fraunhofer		
Version	1.1		
Authors	Lukas Morand, Juan Pablo de Andres, Kuo-I Chang, Helm Dirk, Yoav Nahshon (Fraunhofer)		
Contributions	CERTH, UPV, Technical Partners, Toolkit Development Leaders		
Reviewers	Miguel Angel Mateo Casali (UPV), Adrian Martinez (TECNORED)		
Abstract	This deliverable presents the architecture that the DiMAT project will follow, its main features and characteristics, as		

	well as a general description of its components
Keywords	DiMAT, Architecture, Standard, Deployment, Interoperability, Security, Communication, Scalability

Document Revision History

Version	Date	Description of change	List of contributor(s)
0.1	19-Jul-2023	ToC	Fraunhofer, CERTH, UPV
0.2	20-Sep-2023	1st Draft	Fraunhofer, Technical Partners, Toolkit Developers
0.3	22 -Sep-2023	Internal review	UPV, TECNORED
0.4	25-Sep-2023	2nd Draft addressing the comments from internal reviewers	Fraunhofer
0,5	27-Sep-2023	Quality Control	CERTH
1.0	29-Sep-2023	Final quality check and issue of final document	CERTH
1.1	15-Nov-2024	Update of deliverable according to PO/reviewers comments and quality check	Fraunhofer, CERTH

EXECUTIVE SUMMARY

This deliverable will present the second iteration on the architecture for DiMAT. Rather than focusing on the groundwork introduced in D3.1, it will emphasize the actual structure that DiMAT will follow and complement the work that D3.3 will elaborate on the viewpoints.

For context, relevant methodologies and terms are briefly presented first. The focus of the deliverable is however on the architecture, expanding on the concept and main features that rule the design. Next, the concept as applied to DiMAT is shown, with a brief explanation on each toolkit presented, as well as their deployment strategy and a list of the expected data exchanges among toolkits.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CHADA	Characterisation Data
DL	Deep Learning
DSMS	Dataspace Management System
DTPC	Digital Twin for Process Control.
DT	Digital Twin
EMMO	Elementary Multiperspective Material Ontology
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
IoT	Internet of Things.
IaaS	Infrastructure as a Service
KG	Knowledge Graph
Laravel	A PHP web application framework
LCA	Life Cycle Assessment
LCC	Life Cycle Costing
ML	Machine Learning
MOAM	Message-Oriented Application Model.
MOM	Message-Oriented Middleware.
MODA	Modelling Data
MQ	Message Queues.
Neo4J	A type of graph database.
PaaS	Platform as a Service
PHP	Hypertext Preprocessor
RESTful API	Representational State Transfer Application Programming Interface.
SaaS	Software as a Service
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer.
UI	User Interface
VLAN	Virtual Local Area Network

DiMAT Toolkits

CMDB	Cloud Materials Database
KAF	Knowledge Acquisition Framework
MEC-LCA	Materials Environmental and Cost Life Cycle Assessment
MDF	Materials Design Framework
MM	Materials Modeler
MD	Materials Designer
MMS	Materials Mechanical Properties Simulator
MPS	Materials Processing Simulator
DTPC	Digital Twin for Process Control

1 INTRODUCTION

The DiMAT architecture has been developed following the ISO/IEC/IEEE 42010 standard [1]. For an analysis of this standard and relevant reference architectures, refer to the first version of this deliverable (D3.1) [2]. The reference architectures identified therein (Industrial Internet Reference Architecture, Reference Architecture Model Industrie 4.0, International Data Spaces – Reference Architecture Model, Intelligent Manufacturing Systems Architecture, MarketPlace) served as a basis for discussion about the architecture to be developed for DiMAT. Specifically, the DiMAT architecture follows the Industrial Internet Reference Architecture (IIRA) in terms of the defined viewpoints. The IIRA considers four viewpoints, which are the business, the usage, the functional and the implementation viewpoint. The DiMAT architecture also considers these viewpoints. In the business viewpoint opinions from the relevant industry stakeholders on the toolkits and the suites are gathered and each toolkit is further analysed in detail in terms of usage, functional and implementation viewpoints providing in this way a thorough analysis covering multiple perspectives relevant to each toolkit The specific aspects of each viewpoint are provided in dedicated deliverables D3.3 and D3.4. Besides the IIRA, also elements of other reference architectures are taken into account like MarketPlace, which resmebles the CMDB toolkit. This document will focus on providing an overview of the DiMAT architecture, and complements the first version of D3.1, which served as overview over existing solutions.

While deliverables D3.3 [3] and D3.4 will present details pertaining the different aspects of the design (business, usage, implementation, and functional), this deliverable will fill in the missing gaps and connections to establish a modular deployment design that can be scaled to support further solutions in the future.

The goal of this document is then not just to present how the DiMAT solutions will be created, but to define an architecture that others can use as a reference in the future for integrating software solutions to an ecosystem hosting secure toolkits in a similar way to what DiMAT envisions. This document aims to support the implementation and deployment providing an overview and guidelines on how the technical work in the project shall proceed.

2 GLOSSARY

- **CHADA:** The purpose of CHADA is to provide a standard structure for documenting materials characterisation methods, by including information on the use case, the specific experiment, the raw data, and the data processing. It has been developed in the OYSTER project [4], following the 'template' of the MODA. CHADA has also been the subject of a CEN Workshop Agreement [5]. More information regarding CHADA can be found on [6].
- **Digital Twin:** Although the term is not standardized, a Digital Twin is a digital model of a tangible or intangible real-world entity. A digital twin refers to the complete virtualization of a system (e.g., manufacturing equipment, electronic device, room, etc.). This virtualization allows the monitoring of the system, providing access to functions such as forecasting, process simulation and emulation based on the functionalities that the physical twin (I.e., the actual equipment) partakes. The Digital Twin models allow for near real-time communication with the Physical Twin, enabling also the remote control of the equipment.
- **EMMO:** The Elementary Multiperspective Ontology has been developed in the European Materials Modelling Council (EMMC) framework to provide a standardized and structured vocabulary for describing materials and their properties in a machine-readable format. EMMO aims to promote interoperability and seamless data exchange between different materials modelling software and data sources.
- **Knowledge Graph:** The knowledge graph is a structured representation of data. Analogous to the mathematical object graph, it consists of nodes and edges. The nodes of the knowledge graph represent entities and, the edges connect different entities, denoting the relationships among them. Furthermore, the entities and relationships can maintain several attributes, providing supplementary information. Knowledge graphs are a very intuitive method of representing semantic data, revealing intricate and maybe hidden correlations.
- **Life cycle assessment:** Life Cycle Assessment (LCA) is an analysis whereby the potential environmental impacts of a process, product or service are quantified. This analysis can consider the entire life cycle of the product/process, from raw material extraction to production and consumer use, and what happens when end-of-life is reached. At the same time, a Life Cycle Costing (LCC) method can be implemented to assess the economic impact of a product or process.
- **MODA:** The purpose of the MODA is to standardize the documentation of materials modelling information for specific simulation. Such a standardized terminology is very important for the exchange of information in a seamless manner between experts. In the EMMC-CSA project [7], Fraunhofer developed a MODA webtool, offering a template for users to describe material models. It includes information on raw data, physics-based models, raw output, and processed output to describe workflows at a high level. MODA has been the subject of a CEN Workshop Agreement [8].

3 ARCHITECTURE

Before presenting the specific DiMAT architecture, we will explain the ruling concept and characteristics that have guided the design.

3.1 CONCEPT

Our final draft of the architecture is based on a modular, scalable design that integrates individual toolkits in a distributed manner, allowing communication both intra- and inter-toolkit and a sharing of resources:

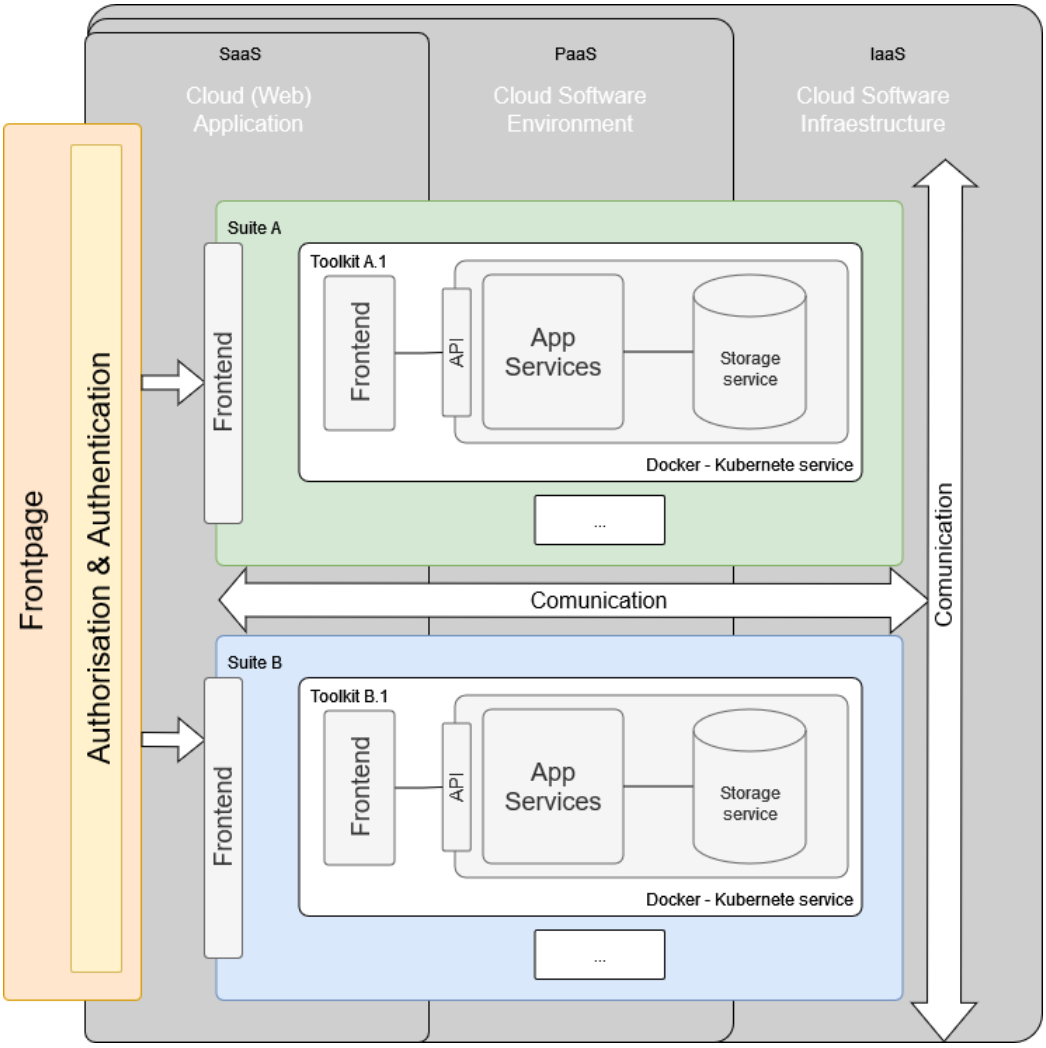

Figure 1: Architecture concept

Toolkits (see *Toolkit A.1* and *Toolkit B.1*) are generally standalone, with their own frontend and backend services, as well as storage. They can be initially grouped in Suites (see *Suite A* and *B*), offering shared resources such as a common frontend. Suites can be connected among

themselves and share resources, such as an authorization and authentication system or a global frontpage.

The following subsections will elaborate in the most relevant aspects taken into consideration in this conceptual design.

3.1.1 Communication

In the DiMAT project, the toolkits can use the modern RESTful API approaches or traditional Message Queues (MQ) for communication implementations. DiMAT's dual support ensures seamless integration, flexibility, and scalability, catering to diverse communication needs and enhancing the platform's robustness in various deployment scenarios.

The DiMAT communication framework can use the Message-Oriented Application Model (MOAM), emphasizing message-based communication over traditional methods. This approach is ideal for distributed systems, enabling interaction across diverse platforms and networks. Central to MOAM is the Message-Oriented Middleware (MOM), a software bus that integrates varied applications, ensuring efficient message delivery while abstracting communication complexities.

As web technologies advance, there is an emerging adoption towards RESTful APIs for streamlined communication interfaces. DiMAT will use as the primary communication method RESTful APIs approach that offers several advantages. Firstly, it capitalizes on the ubiquity and familiarity of HTTP, making integration more straightforward across diverse platforms and languages. Secondly, it aligns with the stateless nature of the web, ensuring scalability and ease of deployment. Furthermore, using REST APIs promotes a more lightweight, flexible, and web-friendly architecture, making it particularly suitable for cloud-native applications and microservices-oriented designs.

DiMAT's cloud environment leverages messaging, fostering communication among its toolkits and ensuring scalability and resilience. This asynchronous operation decouples service consumers from implementers. The DiMAT message bus, using HTTPS, offers a robust security model and a universally accepted transport protocol.

Where necessary, the classic MOMs two communication styles (point-to-point and publish-subscribe) will be used within DiMAT. The former involves direct communication between two entities, while the latter, apt for ubiquitous computing environments, facilitates many-to-many interactions. Here, publishers send messages to topics, and subscribers retrieve them, promoting a decoupled architecture. This decoupling enhances flexibility and scalability, minimizing dependencies between software entities. Consequently, changes in one component don't impact others. This model complements the service-oriented architecture, with services referring to entities within the MOM.

In the DiMAT project, the RoboFuse [12] platform can be used to support both traditional Message Queues (MQ) and the modern RESTful API approaches for Message-Oriented Middleware (MOM) implementations.

Communication perspective on Toolkits integration

For the DiMAT project, when addressing inter-application communication and integration, the RESTful API over HTTPS is the preferred technology. This choice is informed by the project's requirements, and the integration approach will be organized based on the classification framework proposed by Linthicum [10].

1. Data Level: Pertains to data transfer between various sources. It involves extracting, processing, and relocating data. While this level is cost-effective and requires minimal toolkit modifications, it confines business logic to the primary toolkit, limiting real-time transactions.
2. Application Interface Level: Focuses on toolkit interoperability by sharing common business logic through a predefined programming interface. It primarily exposes interfaces from packaged or custom toolkits for service consumption or data retrieval.
3. Method Level: Often termed the business integration level, it encompasses shared organizational business logic, including data access services, security, and foundational rules.
4. User Interface Level: Aims to establish standardized, typically browser-based, interfaces for a collection of applications, often legacy ones. While integrations at this level can be more tightly coupled, leading to higher maintenance costs, they are simpler to implement. This level is crucial for ensuring a uniform and efficient user experience, particularly with legacy systems.

DiMAT will address these points as follows:

1. User Interface Level: The goal for this level is to ensure that DiMAT toolkits adhere to consistent UI (User Interface) design principles. This consistency provides users with a uniform "look and feel" across all DiMAT interactions, making it straightforward to recognize applications stemming from the DiMAT project.
2. Data level: Cloud Materials Database is the main storage solution in DiMAT, storing process information and datasets of heterogenous data.
3. Method Level: it encompasses shared organizational business logic, including data access services, security, and foundational rules.
4. Application Interface Level: DiMAT expose integration APIs as Web services; Web services follow the guidelines provided by the integration platform.

Communication security

The DiMAT platform is designed to be cloud-based, and with this technological direction, specific security measures are imperative (details on Annexes):

- Secure Access: Our API endpoints, which serve as access points, support HTTPS. This ensures that communication sessions are established securely using SSL.
- Integrated Firewalls: The platform offers configurable firewall rules, allowing us to determine the accessibility of our instances. Whether it is selected to be entirely public, completely private, or a balance of both, it is covered. Additionally, for instances within a VLAN (Virtual Local Area Network) subnet, both egress and ingress can be managed.
- API Authorization: To access the different available resources, users must be logged in and provide the required authentication information.

By prioritizing secure access, leveraging integrated firewalls, and emphasizing API authorization, we are taking proactive steps to safeguard DiMAT platform.

3.1.2 SECURITY

Security is a very important aspect that should be considered from the beginning of the project to identify the potential risks to the different assets, the probability of them occurring, and the severity of their impact, to implement mitigation policies.

3.1.2.1 Authentication and authorization

To control that only registered users can access the toolkits and which resources they are allowed to read or modify, DiMAT needs an Identity and Access Management system. For this purpose, we will use the tool called Keycloak [13]. This tool is a versatile, widespread open-source solution that offers Single-Sign On and can connect to external Identity Providers, potentially allowing users to log in via other platforms where they already have an account.

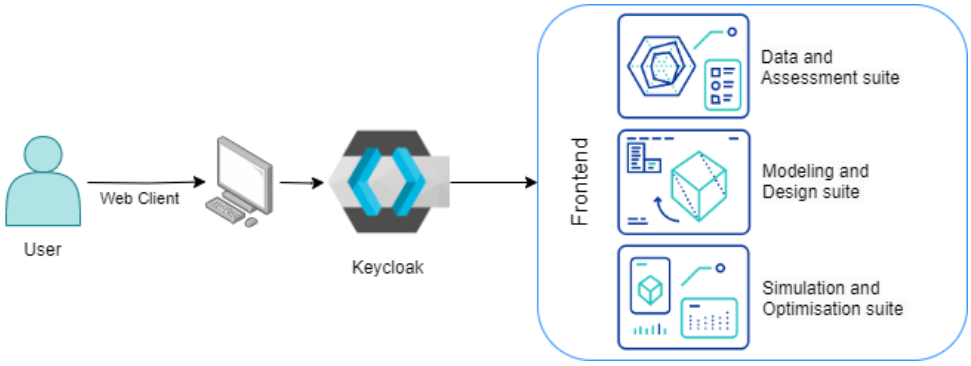

Figure 2: Keycloak integration in the DiMAT Architecture

To make the user experience among the different toolkits seamless and smooth, DiMAT will use a unified, central Keycloak deployment (see Figure 2). This approach offers several significant benefits for our users:

- **Unified Login:** Keycloak enables users to use the same login credentials across all DiMAT tools, simplifying access.

- **Enhanced Security:** Keycloak provides robust security features such as two-factor authentication and secure password management, ensuring the protection of user data.
- **Centralized User Management:** Keycloak streamlines user administration by allowing administrators to manage user accounts centrally, making access management and permission assignment more efficient.
- **Consistent User Experience:** Keycloak ensures a consistent user experience across all DiMAT applications, reducing user confusion and improving usability.
- **Time and Resource Savings:** Keycloak eliminates the need to maintain multiple accounts and authentication systems, saving users and technical support staff time and resources for more strategic activities.

3.1.2.2 Back-ups and Recovery

A back-up and recovery system must be developed and maintained to minimize the impact of possible failures, losses, or attacks.

There are different approaches to back-ups:

- **Full back-ups:** complete system snapshots at certain times. Require more space and time to create but are useful against major losses or failures.
- **Incremental back-ups:** include only the changes since the last back-up. They require less resources but are more complex to restore.
- **Differential back-ups:** similar to incremental back-ups, they include only the changes that occurred, but in this case, in relation to the latest full back-up (ignoring other differential back-ups that might have taken place in the meantime).

Independently of the approach used for backing up, some things to consider are:

- **Frequency:** back-ups should be done often enough to recover from any issue, while considering the available resources.
- **Retention policies:** how long back-ups will be stored.
- **Storage:** back-ups should not be saved in the same location/system as the original data. Otherwise, they might also be affected by the loss/failure that might occur on the running system.
- **Security:** only certain people should have access to the back-ups for privacy and integrity of the data contained.
- **Testing:** Back-ups should be periodically checked to ensure they are functioning and comprehensive enough.

While the partner hosting the toolkit is responsible for the back-ups, a policy will be communicated and reviewed with the relevant DiMAT partners.

3.1.2.3 Security updates

Security updates are of utmost importance to ensure data protection, regulatory compliance and system availability.

The chosen deployment strategy will allow for additional quick bugfix releases that can tackle threats as they are identified, without having to wait for the following feature release.

3.1.2.4 Documented best development practices

As part of the initial efforts to partially systemize the development efforts among all partners and toolkits, a live "development manifesto" has been compiled. Said document defines some best practices and policies for all developers to follow, which shall increase the readability, maintainability, and security of the system.

Some suggestions worth mentioning here are:

- Safety of dependencies: All dependencies and third-party libraries used shall be periodically checked for bugs and other issues. For this, existing tools such as safety [14] (for Python) can be integrated for a more automatized review.
- Input validation and sanitization: handle inputs from users with care, both to avoid potential errors and attacks.

3.1.2.5 Logging and monitorization

All toolkits shall produce appropriate logging messages to help pin down issues and identify their source. Said logging should be categorized based on the urgency of the message provided (e.g., debug, info, warning, or error).

Furthermore, a system will be set up to periodically check the system's availability and notify in case of failures or deviations.

3.1.3 Scalability

Inside the Reference Architecture of DiMAT, scalability refers to the ability of a system to adapt and grow as the demand or workload increases. In system design, considering this capacity is crucial, especially for enterprise or high-demand systems. A scalable reference architecture allows for the addition of further resources, such as servers, storage, or network bandwidth, to meet changing needs. This enables the system to operate efficiently even with significant growth in users or data. In a reference architecture, there are two main types of scalabilities:

- **Vertical scalability** involves adding more resources to a specific instance in order to increase the system's capacity. This may involve upgrading hardware, such as adding

more RAM or increasing CPU processing power. However, vertical scalability has a physical limit as a maximum number of resources can be added to a single instance.

- **Horizontal scalability** involves adding more instances of the system instead of increasing the resources of a single instance. This is achieved by distributing the workload among multiple servers, allowing the system to handle a higher workload concurrently. Techniques such as clustering, load balancing, and data partitioning are used to achieve horizontal scalability.

Focusing on the reference architecture defined for DiMAT, it has been designed considering the forementioned aspects. To this end, resources can be added or removed as needed, so that it is adaptable to the changing requirements of the system. These aspects can be defined at the time of deployment of the solutions, thanks to technologies such as Docker or Kubernetes (which we will define later). In such a way, the system is guaranteed to be scalable to meet all the demands under the available performance without affecting the quality of service. The strategy to be followed in DiMAT will be:

- **Cloud deployment:** This strategy leverages cloud services to flexibly implement and scale the system. Cloud service providers offer on-demand computing and storage resources, allowing capacity to be adjusted according to changing needs. This provides the advantage of rapid horizontal scalability.
- **Container deployment:** Containers provide a lightweight and isolated environment for running applications. This strategy involves packaging system components into individual containers and deploying them independently.

DiMAT will be developed to be deployable for these deployment strategies using Docker and Kubernetes. Common technologies within scalable reference architecture:

- **Docker** is a platform that allows applications and their dependencies to be encapsulated in independent containers. These containers are lightweight and provide a consistent way to run applications in different environments, ensuring dependency control. When using Docker in a reference architecture, individual containers can be created for each system component, facilitating resource management, replication, and independent deployment.
- **Kubernetes** is a container orchestration platform. It enables the management and coordination of multiple containers in a server cluster. Kubernetes simplifies horizontal scalability by allowing dynamic management of containers, load balancing, and automatic scalability based on demand. By leveraging Kubernetes, a reference architecture can maximize the scalability and availability benefits of Docker containers.

Combining Docker and Kubernetes in a reference architecture allows for an agile and scalable infrastructure. Docker facilitates the consistent packaging and deployment of system components in containers, while Kubernetes provides the necessary tools to orchestrate and manage those containers efficiently. These deployment technologies are

highly valued in designing scalable and flexible architectures, as they enable efficient resource management and agile deployment in both local and cloud environments.

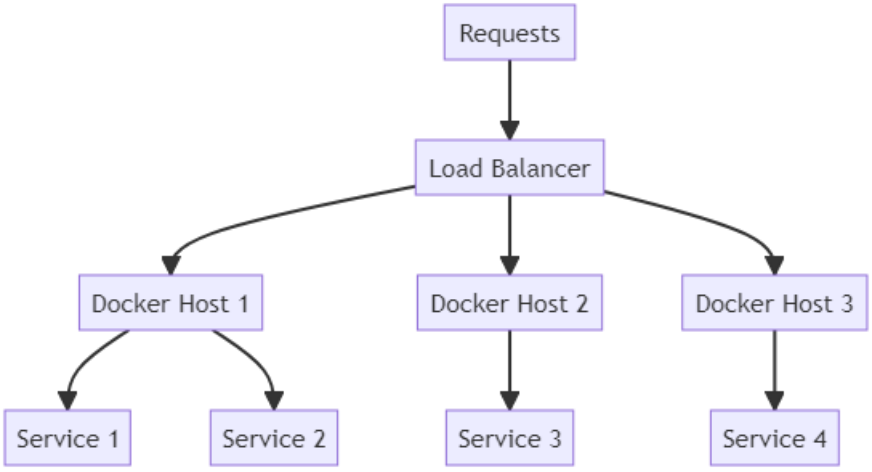

Figure 3: Docker horizontal scalability approach

- **Requests**: Incoming traffic or requests from users or systems.
- **Load Balancer**: Distributes incoming traffic across multiple Docker hosts.
- **Docker Hosts:** Individual servers or virtual machines that run Docker containers.
- **Services**: Individual applications or microservices running inside Docker containers.

In this setup, as the traffic increases, more Docker hosts can be added to handle the load, ensuring that each service remains responsive. The DiMAT platform's decision to deploy using Docker brings forth a multitude of practical advantages, especially in terms of scalability. Docker containers encapsulate the DiMAT applications and their dependencies, ensuring a consistent environment across various development and deployment stages. This uniformity simplifies scaling processes, as new instances can be spun up without the typical concerns of environmental discrepancies.

Docker's lightweight nature means that the DiMAT platform can rapidly scale out (horizontal scalability) by deploying additional containers as the demand grows. This is especially beneficial during peak usage times, ensuring that the platform remains responsive and efficient. Moreover, Docker's compatibility with orchestration tools, like Kubernetes, further enhances scalability. Kubernetes can automatically scale the number of containers based on the workload, ensuring real-time optimal resource utilization.

Furthermore, Docker's inherent design promotes microservices architecture. By breaking down the DiMAT platform into smaller, independent services, each encapsulated in its container, the platform can scale specific services based on their individual demands rather than scaling the entire application. This selective scaling is cost-effective and ensures efficient resource allocation. By leveraging Docker, the DiMAT platform is poised to handle varying

loads, ensuring robust performance, reduced overhead costs, and an enhanced user experience, all while maintaining the platform's integrity and consistency.

3.1.4 Adaptability

Section 3.1.3 introduced the concepts of vertical and horizontal scalability in regards to the deployment and the resources invested in it. DiMAT's conceptual architecture takes this one step further ensuring the solutions themselves are scalable. The toolkits within each suite can be customized and extended when new tools are developed, and more suites might be created to satisfy new needs that arise. Suites serve not only as an architectural grouping that enables sharing of resources among a subset of toolkits, but also as a conceptual organization of tools that are aligned in their functionality. Section 3.2 will show how this has been applied in the case of the DiMAT project itself.

The modular design means this concept is easily scalable, but also that it can be adapted and tuned based on the requirements of the system being designed. Each toolkit is designed to operate independently and function in a standalone way. However, interactions between tools are both supported and beneficial. For instance, a tool might implement its own storage mechanism to host input and output parameters from a simulation. Nevertheless, it might also support connecting to another toolkit and using the information made available by this second toolkit (either because it has data storage itself or because it might be a data source generating new data).

Such flexibility will allow users to benefit from those solutions relevant to them without having to familiarize themselves with a complex system that has features unrelated to their needs.

3.1.5 Interoperability

In the context of DiMAT, interoperability stands as a cornerstone to ensure seamless communication (addressed at 3.1.1) and integration, especially given the diverse toolkits and systems involved. As digital ecosystems grow more complex, the need for smooth interaction between different solutions becomes essential. Addressing interoperability ensures that digital barriers are minimized, promoting effective collaboration. Here is how DiMAT addresses this:

- **Unified Terminology:** DiMAT emphasizes the importance of a common language during the design phase. This shared terminology ensures clarity, reduces misunderstandings by fostering mutual understanding, streamlining integration efforts.
- **Stable Service Interfaces:** DiMAT focuses on crafting stable and consistent service interfaces, ensuring reliable communication and reducing integration complexities.
- **Description of Basic Components**: A clear outline of the foundational components of existing solutions within DiMAT provides a coherent base for further development,

ensuring compatibility.

- **Standardized Data Structure**: DiMAT adopts a standardized data structure, facilitating coherent data interpretation and exchange across its solutions.
- **Common Communication Protocol**: DiMAT employs a standardized communication protocol, ensuring smooth data and information flow across its platform.
- **Collaboration and Knowledge Sharing**: The DiMAT ecosystem encourages collaboration, with partners sharing insights and best practices to enhance interoperability.

As the DiMAT platform embarks on its journey to ensure smooth interactions across its toolkits and services, a structured approach is essential. The following methodology outlines the systematic steps and considerations that DiMAT adopts to ensure robust interoperability, ensuring that all components not only coexist but also synergize to deliver an integrated, efficient experience.

- **Interoperability Assessment**: Organize assessment sessions of the current systems, tools, and platforms in use. Identify potential bottlenecks, compatibility issues, and areas that require integration.
- **Standardization**: Adopt or develop standardized protocols, data formats, and communication methods. Ensuring that all systems adhere to a common set of standards is crucial for seamless integration.
- **Modular Design**: Design systems in a modular fashion, ensuring that each module or component can be integrated or replaced without affecting the overall system's functionality.
- **Integration Testing**: Once systems are integrated, conduct rigorous testing to identify any issues or gaps in interoperability. This should include scenario-based testing to simulate real-world interactions.
- **Feedback Loop**: Establish a continuous feedback mechanism with end-users and stakeholders. Their feedback can provide insights into any unforeseen challenges or areas of improvement.
- **Continuous Monitoring and Updates**: The digital landscape is ever evolving. Regularly monitor the integrated systems for any changes or updates that might affect interoperability. Ensure that systems are updated in tandem to maintain seamless communication.
- **Collaborative Ecosystem**: Foster a collaborative environment where solution providers, developers, and stakeholders can share knowledge, tools, and best practices related to interoperability.
- **Review and Iteration:** Periodically review the interoperability strategy and make necessary iterations based on technological advancements, changing requirements, or feedback.

Figure 4 shows how these principles should be considered both in the design of each specific solution and between solutions to ensure an optimal degree of interoperability inter- and intra-toolkits.

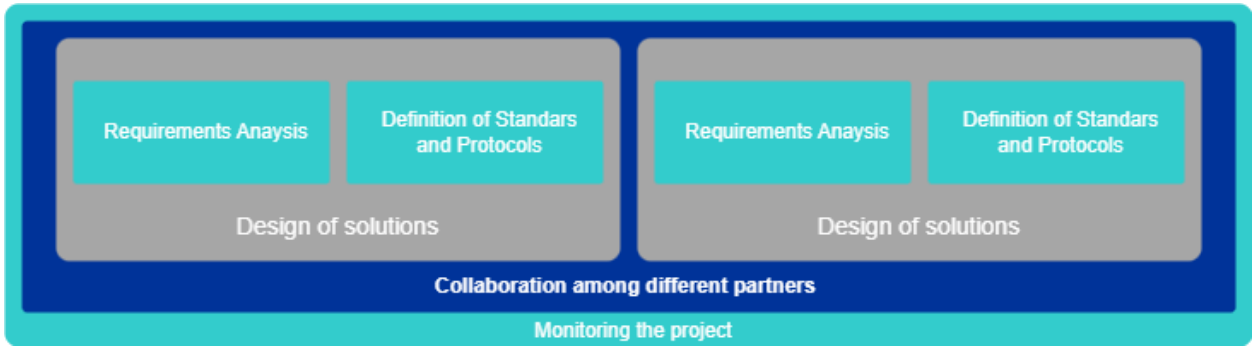

Figure 4: Interoperability methodology

Achieving interoperability among solution providers is crucial to ensure seamless and efficient integration in the digital environment. By following the proposed methodology, digital barriers can be overcome, and effective collaboration can be promoted.

3.2 DIMAT'S ARCHITECTURE

Figure 5 shows how the concept defined in Section 3.1 as it is applied to the specific toolkits and use cases of DiMAT. While it focuses on the toolkits described in the Description of Action, we have seen how this framework could easily be adapted to include further toolkits and suites. The DiMAT architecture also includes a centralized Keycloak instance to manage user access across all the toolkits. Additionally, a frontend that serves as a starting point for users, where all the toolkits are linked, is provided.

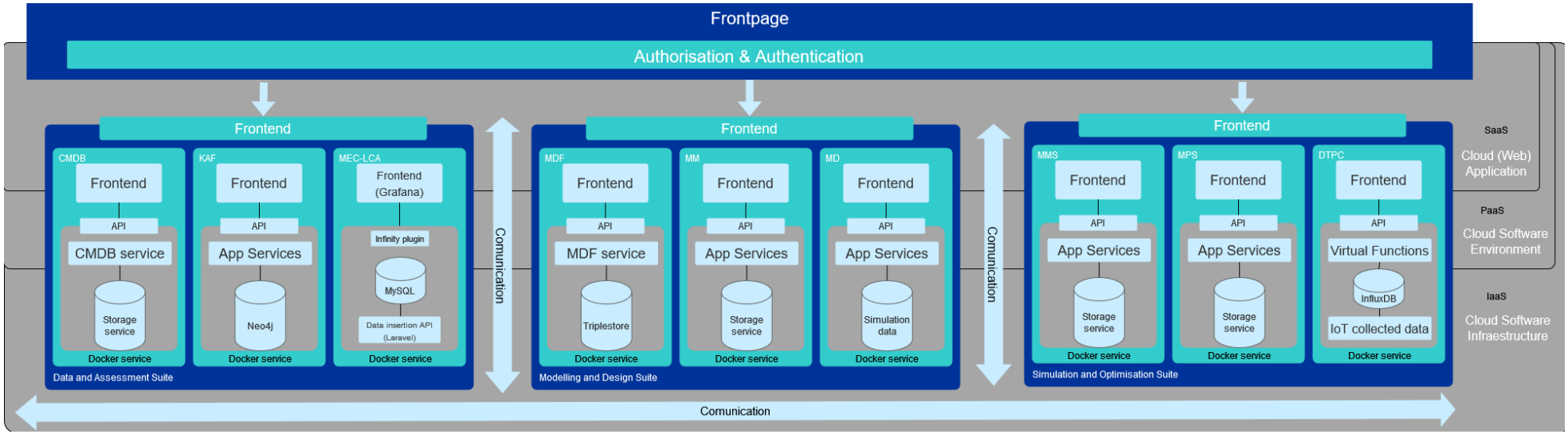

Figure 5: DiMAT architecture

Figure 5 provides only a flat overview on the toolkits within the DiMAT architecture, in the following, a detailed overview of the architecture of each toolkit is provided per DiMAT Suite. Basically, the nine toolkits that DiMAT will develop are grouped in three suites based on their own purpose. Deployment strategies and preliminary interconnections between the toolkits will be listed in the Sections 3.2.1, 3.2.2, and 3.2.3 to give an idea of how the different toolkits will enrich and benefit each other.

DiMAT Data and Assessment Suite

The architectures of the toolkits of Suite 1 are shown in detail in Figure 6. The Cloud Materials Database (CMDB) toolkit is a system for storing, sharing, and exploration of material data for materials design, processing, and manufacturing processes. It consists of a frontend and a backend API. Every node in the depicted in Figure 6 runs in a separate Docker container (note that the containers are not depicted for visualisation purposes and to maintain clarity). Besides using the frontend, users can interact via a JupyterHub environment and via user-defined apps. A core component of the backend is the vocabulary service that forms the basis for a common vocabulary used together with the KAF toolkit. Based on the vocabulary, CMDB and KAF follow the FAIR data principles to facilitate interoperability between all the toolkits. A direct access to the data stored in the KAF toolkit will be established.

The Knowledge Acquisition Framework (KAF) toolkit consists of three basic components with each one acting as a separate docker container. A custom frontend that serves as the User Interface through which the users can access the stored data and the supported functionalities. The latter are provided through the backend that consists of data population and querying mechanisms as well as visualization algorithms. At the heart of the toolkit lies the graph database which connects with the relevant services (functionalities) through dedicated APIs, exchanging data.

The Materials Environmental and Cost Life Cycle Assessment (MEC-LCA) toolkit is set up as a Docker service with three docker containers. The first container includes the MEC-LCA Frontend that utilizes Grafana for data visualization, with data stored in a Grafana database and accessible through both frontend and backend, and the functionalities included in Grafana Backend API such as data calculations, uploads (using the Infinity plugin), downloads, data editing, and dashboard modifications. These functionalities are supported by a MySQL database, which is populated by an LCA Data Population API developed in Laravel, both running in its own container.

More detailed descriptions can be found in the business, usage, functional, and implementation viewpoint in deliverable D3.4.

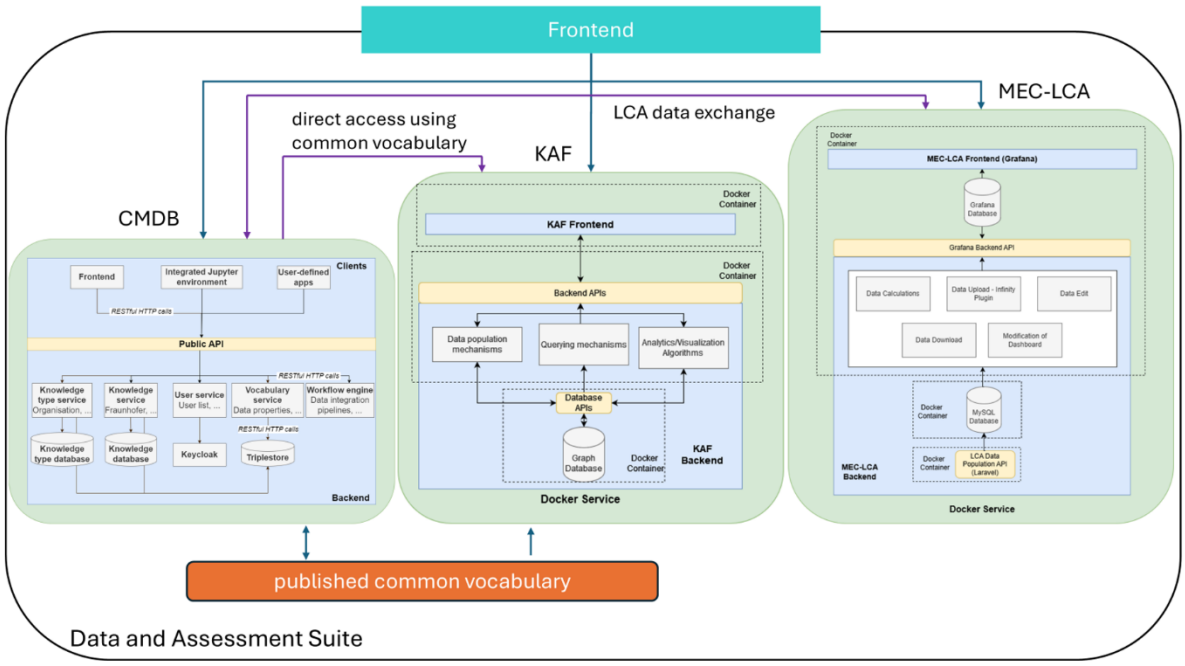

Figure 6: Architecture of the DiMAT Data and Assessment Suite and inter-Suite interactions

DiMAT Modelling and Design Suite

The architectures of the toolkits of Suite 2 are shown in detail in Figure 7. The Materials Design Framework (MDF) toolkit consists of three separate components (apps), each having its own frontend and a separate backend API and run inside individual Docker containers. The components can be access via a common MDF frontend that runs in a separate Docker container. Also, all the components act on the CMDB toolkit, either by using it as a database (materials relation component) or by accessing and evaluating stored data (correlation and search component).

The Materials Modeler (MM) toolkit is a Dockerized system for data manipulation, querying, and analytics on materials data. It includes a frontend interface and a backend with components for data cleaning, retrieval, and analysis, all connected to a central database. Running in Docker containers, MM is modular, scalable, and ideal for research and production use.

The Materials Designer (MD) toolkit is organized in orchestrated Docker containers which encompass the toolkit frontend and backend. The first container implements the toolkit frontend, which exposes a web interface and allows the user to interact with the toolkit itself. The toolkit backend is made of different containers, each implementing a specific service.

The first one is responsible of managing the user demands and queries and to request the calculation of new material properties. Another container is the material database, which stores all the information of the defined materials and their properties, once calculated. The third container implements the scheduler, which takes care to prepare the input data for the calculation code and to start its execution. This container manages an internal simulation database.

More detailed descriptions can be found in the business, usage, functional, and implementation viewpoint in deliverable D3.4.

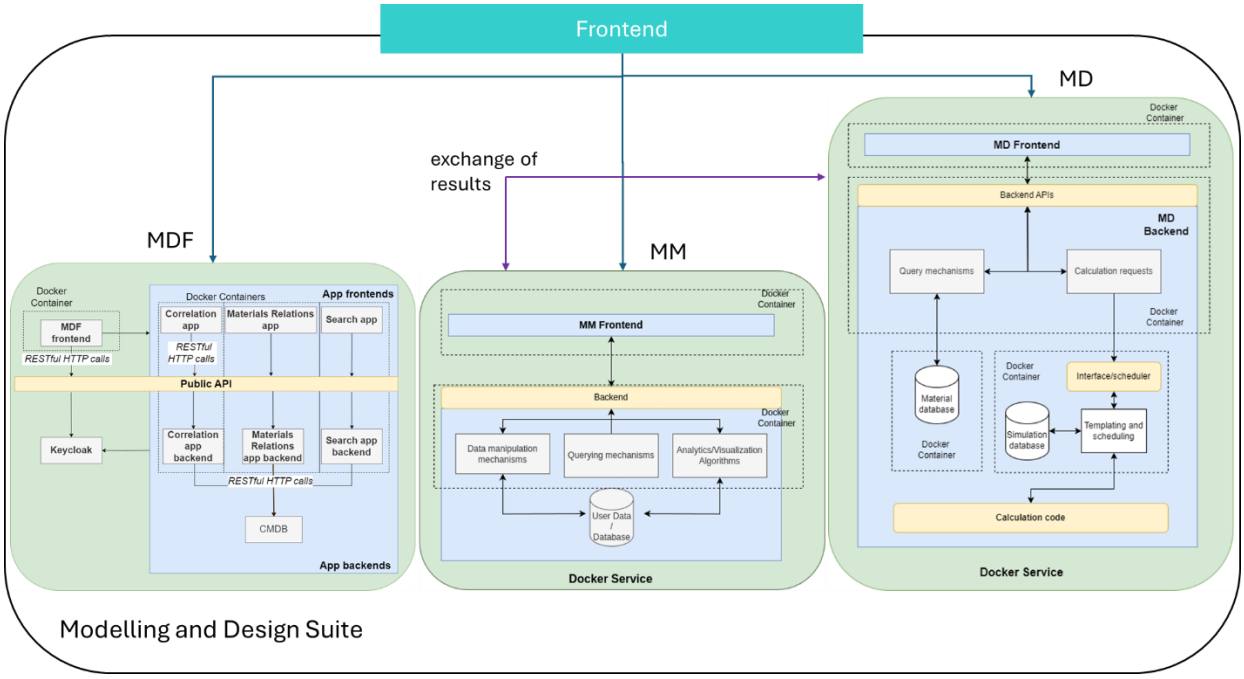

Figure 7: Architecture of the DiMAT Modelling and Design Suite and inter-Suite interactions

DiMAT Simulation and Optimizsation Suite

The architectures of the toolkits of Suite 3 are shown in detail in Figure 8. The Materials Mechanical Properties Simulator (MMS) toolkit is a real-time, Dockerized platform developed to predict the mechanical properties of materials. It features a frontend interface and a backend API that manages input, output retrieval, and data visualization. The system ensures high performance for both research and industrial applications.

The Materials Processing Simlation (MPS) toolkit will use data population mechanisms to manipulate the different kinds of input and output to develop proper manufacturing process simulations and store them on the DiMAT database. Querying mechanisms will need the correct manipulation of the toolkit by the user. Functionalities of analytics and visualization simulation results will bring users, by a frontend interface, a proper understanding of the process simulation and the changes needed. The different functionalities will be connected to Database APIs to provide proper data to the user and continue feeding the DiMAT database with the possibility of interacting with other toolkits

The Digital Twin for Process Control toolkit (DTPC) is organized as three separate docker containers that communicate over exposed APIs. The frontend allows the interaction with the toolkit over a webpage. Then, the backend supports various functionalities, covering data analytics, dashboards for effective visualization of the status of the twin as well as specific control functionalities. The last part of the DTPC architecture is the Digital Twin of the supported system. The NEPHELE VO software stack is employed for this task, and the DT is represented as a combination of Virtual Objects (VOs) that can interact with each other as well as with the actual IoT devices. The VOs can support various virtual functions and also provide data for necessary computations in the backend part of DTPC.

More detailed descriptions can be found in the business, usage, functional, and implementation viewpoint in deliverable D3.4.

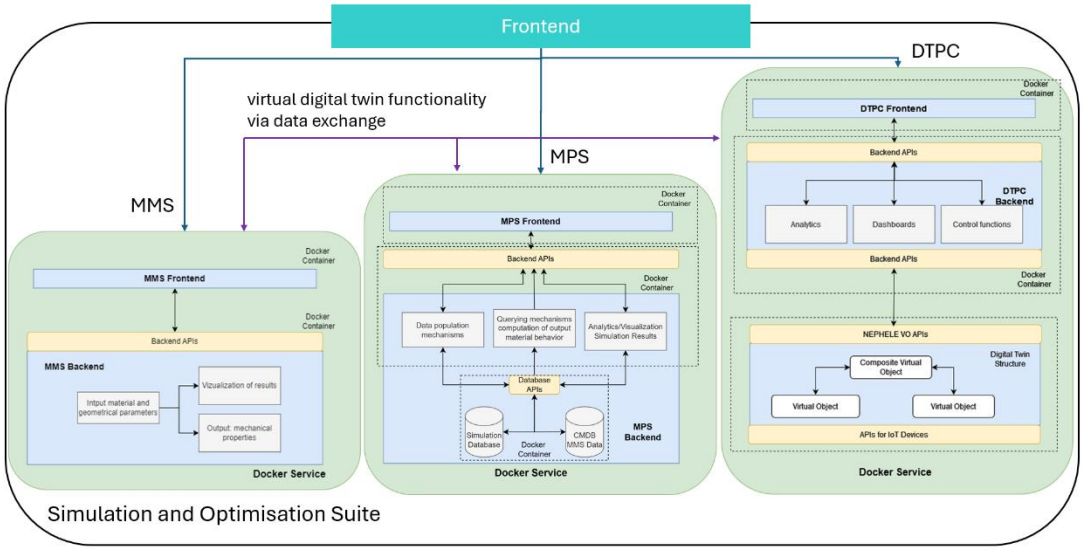

Figure 8: Architecture of the DiMAT Simulation and Optimisation Suite and inter-Suite interactions

DiMAT toolkit interactions and dependencies

In Figure 9, the planned toolkit interactions and dependencies between the toolkits are depicted. DiMAT will enable a vast exchange of data and results within the toolkits. However, all the toolkits, except for MDF, are not dependent on this exchange of data and are developed to be usable stand-alone. The MDF has a dependency on CMDB, which is used therein as the main source of data (for analysing correlations) and for storing materials relations. Furthermore, CMDB can act as a central data repository for all of the toolkits. Data exchange between the toolkits can be established via direct interactions or via CMDB in the form of a multi-toolkit interaction. Apart from that, the toolkits MPS, MMS, and DTPC (Suite 3) plan to provide a virtual digital twin functionality via interactions. The design and development of this functionality is ongoing.

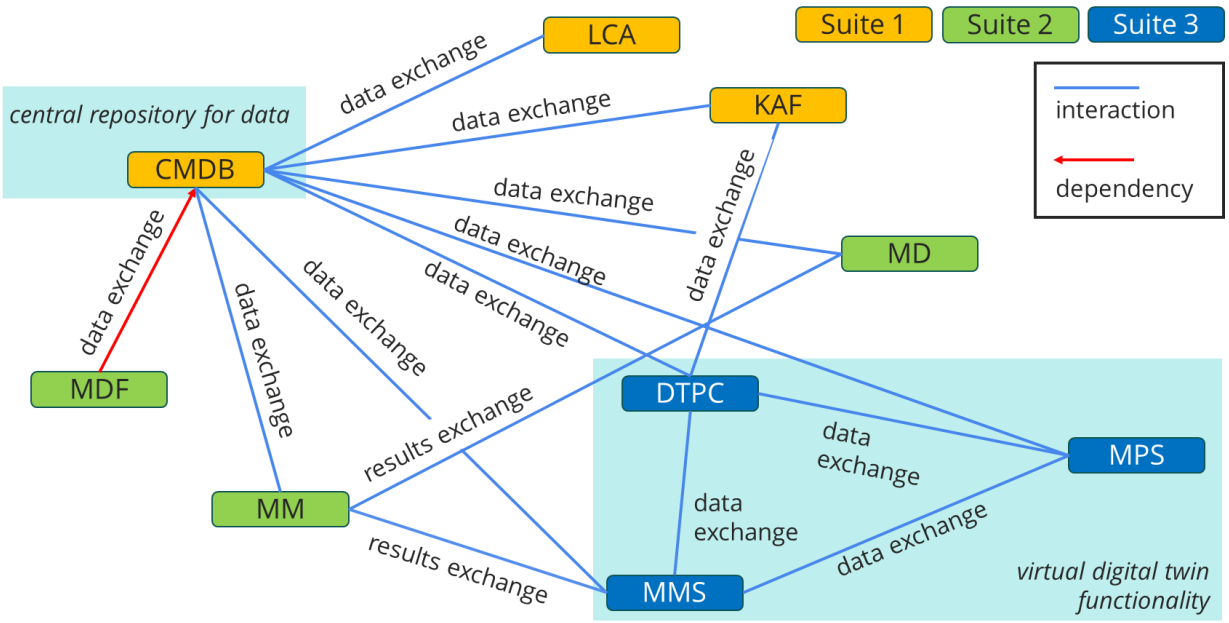

Figure 9 : Toolkit interactions and dependencies

3.2.1 Data and Assessment Suite

A central set of digital technologies for the data storage, data management, and data usage driven by semantic technologies.

3.2.1.1 CMDB

The open Cloud Materials Database is the main storage solution in DiMAT, being able to store process information and datasets of heterogenous data.

Based on the DSMS (Data Space Management System) developed by Fraunhofer, it will allow interaction both via a frontend and via an API, with a Python library being developed for easier handling of the requests.

The CMDB service manages and synchronizes the access to the different storages, namely the relational database, the triplestore and the object storage. It includes characterization and modelling data and their documentation via CHADA and MODA.

Deployment is done via Docker. With regard to already existing solutions (analysed in Deliverable 2.2), CMDB use postgreSQL to store datasets on a low level of semantic integration. To reach higher-level semantic integration, additional technologies (e.g. triplestore) need to be employed.

The following dependencies to other toolkits are expected:

- **KAF:** CMDB will use ontological material information stored in the KAF to add metadata to the stored processes and datasets.
- **CMDB** will interact also with other toolkits to provide possibilities to store input/ output data.

3.2.1.2 KAF

The Knowledge Acquisition Framework (KAF) toolkit enables the representation of materials and their properties in a structured manner in the form of a knowledge graph. A knowledge graph representing a material consists of entities (nodes) and the edges connecting them denote certain relationships. Using KAF, material experts (manufacturers, scientists, etc.) can retrieve information regarding materials and through data analytics and visualization techniques, identify potential correlations among them.

Access to KAF will be provided through an interface and an API allowing the execution of different algorithms based on the role assumed by the user. A graph database like Neo4J will be used for storing the material information, and custom-made data analysis algorithms will be developed in Python. The KAF toolkit will be deployed via Docker.

From the examined available open-source solutions for storing and handling knowledge graphs, the open-source version of Neo4J was selected as the most appropriate graph

database. This is because it is one of the most popular and most maintained graph databases, it has reasonable minimum requirements, can be deployed easily with Docker and exhibits better performance in large graph databases compared to ontologies management and reasoning tools. The required visualizations will be developed with specific tools built on top of Neo4J such as Neodash and other libraries such as Neovis. Finally, the analysis of the data stored in the graph database of KAF will be developed using Python libraries that adhere to the benchmarking criteria of D2.2, such as being open-source, popular and with active communities surrounding them.

KAF is expected to connect to:

- **CMDB**: KAF will connect to CMDB in order to retrieve data concerning materials stored by the pilots.
- **DTPC**: Data exchange about manufacturing processes

3.2.1.3 MEC-LCA

The DiMAT Materials Environmental and Cost Life Cycle Assessment toolkit provides high-level assessment on the environmental and economic impact of the pilot use cases, visualized through a UI. The toolkit will provide Information on core life cycle assessment issues, such as identification of environmental and financial hotspots, as well as data on the pilots' Key Performance Indicators.

The open-source suite Grafana [15]. MySQL acts as the primary internal data storage solution and cross platform PHP-frameworks like Laravel [16] enables the creation of seeders to populate the MySQL database with data. A plugin is also used to enable the direct upload of files into Grafana. The toolkit is deployed using Docker containers and hosted on a DRAXIS server. With regard to already existing solutions (analysed in Deliverable 2.2), OpenLCA was prioritised for use within DiMAT. It will be used to conduct the life cycle assessment studies which will produce the data included in MEC-LCA.

MEC-LCA is expected to connect to:

- **CMDB**: Data exchange between CMDB and MEC-LCA will be established (manual data exchange, not necessarily in an automated manner)

3.2.2 Modelling and Design Suite

Tools for material design, in terms of their internal structure, properties and performance, in order to predict the material behavior before manufacturing.

3.2.2.1 MDF

The MDF is an ontology-based open knowledge system to support the material design process, an App running on DiMAT Open Cloud Materials Database to provide data, information, and wisdom relevant for the material design process. The Materials Design Framework will use the knowledge and data stored in other toolkits to provide suggestions for users given their queries.

As such, it will be aware of the APIs from other toolkits, and its main service will use them to query for information. This MDF service will be connected to a frontend to enable a more natural user interaction, but its API will be documented to allow direct communication.

Internally, it will use the CMDB triplestore to store information on the interconnections with the toolkits, as well as allow creation of new materials relations. With regard to already existing solutions (analysed in Deliverable 2.2), MDF will use scikit-learn for data analysis/ machine learning due to its simplistic nature. For enabling advanced CMDB queries, the application of large language models, e.g. Llama, will be evaluated.

The MDF will mainly connect to:

- **CMDB:** As an application running on top of it, to provide additional functionalities. On the contrary, MDF is dependent on CMDB as its main source/ storage of data within DiMAT.

Deployment is done via Docker.

3.2.2.2 MM

The Materials Modeler (MM) toolkit provides an integrated architecture built on sophisticated AI and causal inference methodologies. Its primary function is to predict, deduce, and recommend ideal configurations for a range of materials under various conditions. The toolkit's architecture is segmented into several core components:

- **Frontend**: A user interface for accessing the toolkit’s features, communicating with the backend to perform processing and analysis.
- **Backend**:
- Data Manipulation: Cleans and structures data.
- Querying: Retrieves specific information from the database.
- Analytics/Visualization: Applies algorithms for insights and visual representations of material properties.

User Data / Database: Stores and manages all user-submitted and processed data, serving as the central source for backend operations.The MM toolkit is devised within a multiscale

material framework and is essential for linking the composition of materials with their mechanical properties. Deployment of this toolkit is facilitated via Docker. With regard to already existing solutions (analysed in Deliverable 2.2), MM is using scikit-learn for development of machine learning models (e.g. materials property prediction) and hyperparameter optimization. Additionally to scikit-learn, pytorch is used for machine learning. Furthermore, Hugging Face is used for providing interactive support using a large language model.

The Materials Modeler will interact with:

- **CMDB:** For data acquisition.
- **MD** and **MMS** for exchange of results. The exchange of results will possibly be established via CMDB.

3.2.2.3 MD

The **DiMAT Materials Designer** is a toolkit for designing materials in terms of mechanical properties, light fastness, thermal properties, rheology, etc., and predict their behaviour during and after their processing. The Material Designer (MD) will calculate the mechanical properties of non-homogeneous materials, starting from those of the base components and the chosen microstructure. Actual calculations will be executed by an external Finite Element Method solver. Regarding already existing solutions (analysed in Deliverable 2.2), MD is using Hexagon Digimat as the external Finite Element Method solver that will be integrated into the toolkit workflow.

The necessary data will be retrieved from the CMDB toolkit or directly entered by the user; also, the calculated properties of the new material will be uploaded, if needed, to the CMDB toolkit.

The toolkit will be deployed via Docker.

As mentioned, the Material Designer toolkit will interact with:

- **CMDB:** to query input data and store output data.
- **MM:** the toolkit will expose an API to allow the MM, if needed, to directly ask for the calculation of properties of new materials. The two toolkits will also exchange data, if needed, indirectly via the CMDB toolkit.

3.2.3 Simulation and Optimisation Suite

Developing the adequate tools for determining manufacturing conditions and concepts while simulating their application, results and requirements, in each one of the materials, processes, and processing conditions.

3.2.3.1 MMS

The MMS is a toolkit for determining numerically mechanical properties such as stiffness, tensile strength, plasticity, viscoelastic and viscoplastic properties, damage, fracture, fatigue, etc. The macro mechanical behavior will be linked with the microstructure and the constitutive equations of the components. The Mechanical Material Properties Simulator will link the composition and volume fraction of material with the mechanical properties. The toolkit is formulated in a multiscale material framework and will use the results of MM toolkit.

Deployment will be done via Docker. With regard to already existing solutions (analysed in Deliverable 2.2), MMS uses LAMMPS, CALCULIX and Gmesh for numerical simulation, as well as not listed tools like FREECAD (for generating geometries to simulate) and PREPOMAX (finite element pre and postprocessing). SALOME, Code-Aster, and TexGen were also explored but were not considered for usage. For machine learning, scikit-learn will be used due to its simplicity and LIME (explainable artificial intelligence) will be explored.

The toolkit will connect to:

- **CMDB:** Both for querying stored results to use as inputs and storing outputs.
- **MM:** To generate hybrid models combining data-driven and physical models
- **MPS:** To develop a viscoelastic material to be used in MPS
- **MPS-DT**: To develop a reduced model part of the virtual digital twin

3.2.3.2 MPS

DiMAT Material Processing Simulator (MPS), will use data of material transformation process, geometry, processing conditions and materials properties for recreating a virtual material processing, giving as output data that is important for the correct processing of the materials, such as pressure, temperature, etc. This output will be used to increase the database, and to optimize the transformation process, by taking the new parameters of the manufacturing, and feeding them back into the process, to obtain new data. With regard to already existing solutions (analysed in Deliverable 2.2), MPS will use OpenFoam (due to its characteristics as a freeware, offering the possibility for recoding existing solvers to be adapted according to

the needs) to simulate the curing resin cycle in Pilot 2. To simulate polymer materials in Pilot 1, OpenFoam will be used in combination with FreeCad (free access software). Furthermore, Calculix is used to analyze the model developed for Pilot 3, obtaining more accurate data regarding the behavior of glass deformation during the manufacturing process.

MPS will leverage information from:

- **CMDB:**. The MPS toolkit intends to interact with CMDB to exchange material properties data and information that could fit the common database while being useful for obtaining more accurate simulations.
- **MMS:** The MPS toolkit intends to interact with MMS to exchange material properties data. Pilot 3 will have a common simulation environment to achieve the demands of the pilot and obtain accurate results.
- **DTPC:** DTPC will allow performed simulations to interact with physical devices and observe real-time simulations of selected parameters and materials.

3.2.3.3 DTPC

The Digital Twin for Process Control (DTPC) toolkit allows the creation of digital twins (DTs) acting as abstractions of the physical manufacturing equipment and/or relevant IoT devices. The DTs offer access to virtual functions such as process simulations/emulations and connection with the physical counterparts. Regarding the examined solutions mentioned in Deliverable 2.2, DTPC will use the NEPHELE VO Software Stack for constructing virtual objects as counterparts of relevant devices and materials. Moreover, the virtual functionalities will be developed with popular open-source Python libraries such as Tensorflow, scikit-learn and Pandas. Storage of time-series data will be accomplished by using InfluxDB database. Communication with the other toolkits of the Suite is necessary for accessing specialized simulation software.

The architecture of the toolkit consists of the following major parts:

- The network edge: IoT devices transmit data that may be aggregated through a Gateway and transmitted to the Virtual Object.
- The Virtual Object: Allows temporary storage of device measurements in a time-series database and access to virtual functions either implemented as part of this toolkit or by communicating with the rest of the suite's toolkits.The cloud layer: Offering long-term storage and access to virtual functions that need a multitude of computing resources.

The users can access the toolkit, that is deployed via Docker, either through an interface or through an API.

The expected communications with other toolkits are:

- **KAF:** DTPC will connect to KAF to retrieve data related to the material's structure in order to design more accurate models.
- MPS and MPS: DTPC will connect to the other tools of Suite 3 in order to provide access to specialized simulation software.

4 CONCLUSION

The architecture that the DiMAT project will be based on has been presented, highlighting important features such as communication, security, scalability, and interoperability. Its modular design can be easily replicated to support further solutions in a collaborative and secure way. User management is centralised for an easier user interaction.

This revision focusses on providing more actual details on deployment and collaboration, complementing the initial research and groundwork laid by D3.1. Furthermore, D3.3 is a complement that provides a deeper per-toolkit perspective through the different viewpoints.

Even though this is the final architecture deliverable, the architecture document will be revisited and subject to potential updates based on the needs and requirements of the project.

REFERENCES

[1] ISO/IEC/IEEE 42010 Standard: "Systems and software engineering - Architecture description", available at hiso.org

[2] DiMAT Consortium: Public deliverable D3.1 "DiMAT architecture v1", available at cordis.europa.eu

[3] DiMAT Consortium: Public deliverable D3.3 "DiMAT viewpoints", available at cordis.europa.eu

[4] OYSTER project, available at cordis.europa.eu

[5] CHADA CEN workshop agreement, available at cencenelec.eu

[6] N. Romanos; M. Kalogerini; E.P. Koumoulos; A.K. Morozinis; M. Sebastiani; C. Charitidis (2019): Innovative Data Management in advanced characterization: Implications for materials design. In *Materials Today Communications*. DOI: 10.1016/j.mtcomm.2019.100541.

[7] EMMC-CSA project, available at cordis.europa.eu

[8] MODA CEN workshop agreement, available at cencenelec.eu

[9] Hohpe, G., & Woolf, B. (2004). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.

[10] "International Journal of Hybrid Information Technology, Vol. 1, No. 3, July, 2008."

[11] Zimmerman, O., Doubrovski, V., Grundler, J., & Hogg, K. (2004). Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned. OOPSLA Workshop on SOA & Web Services Best Practices.

[12] Link to https://robofuse.com/

[13] Link to https://www.keycloak.org/

[14] Link to https://github.com/pyupio/safety

[15] Link to https://grafana.com/

[16] Link to https://laravel.com/