Tec4MaaSEs

Technologies for Manufacturing as a Service Ecosystems

Deliverable D3.1

Digital Twin Models and Services for Factories and Value Networks v1

WP3: DT Modelling, Operation and Governance for resilient value networks

Editor:	Marc Leon Haller
Lead beneficiary:	IOSB
Version:	1.0
Status:	Final
Delivery date:	30/06/2025
Dissemination level:	PU (Public)

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101138517.

Deliverable Factsheet

Grant Agreement No.	101138517
Project Acronym	Tec4MaaSEs
Project Title	Technologies for Manufacturing as a Service Ecosystems
Start date	01/01/2024
Duration	36 months

Deliverable Name	D3.1 Digital twin models and services for factories and value networks v1
Related WP	WP3 DT Modelling, Operation and Governance for resilient value networks
Due Date	30/06/2025

Author	Marc Leon Haller (IOSB)
Contributor(s)	Dimitris Verlekis, Eleni Gkinou (MAG), Ina Steinmetz, Ljiljana Stojanovic (IOSB)
Reviewer(s)	Patricia Casla (TEK), Sotiria Antaranian (ATC)
Approved by	All partners

Document History

Version	Date	Author(s)	Organisation	Description
0.1	23/04/2025	Ljiljana Stojanovic	IOSB	Draft ToC
0.2	13/062025	Marc Leon Haller, Ljiljana Stojanovic	IOSB	Provided input for section 1 and section 2
0.3	16/06/2025	Marc Leon Haller, Ljiljana Stojanovic	IOSB	Updated content from section 2 and provided content for section 4 and the executive summary
0.4	18/06/2025	Dimitris Verlekis, Eleni Gkinou	MAG	Provided input for section 3
0.5	18/06/2025	Marc Leon Haller	IOSB	Finalized the document for the internal review
0.6	25/06/2025	Patricia Casla, Sotiria Antaranian	TEK, ATC	Internal review
0.7	27/06/2025	Marc Leon Haller, Dimitris Verlekis	IOSB, MAG	Integration of comments from internal review
0.8	30/06/2025	Armend Duzha	MAG	Quality check
1.0	30/06/2025	Marc Leon Haller	IOSB	Final version ready for submission

Executive Summary

The Tec4MaaSEs (T4M) project embraces key enabling technologies to support Manufacturing as a Service (MaaS) ecosystem. At its core, it develops a network of Digital Twins (DTs) that possess both trustworthiness and cognition, allowing them to work collaboratively within a distributed value network. To this end, this deliverable reports on the progress made in Task 3.1 "DTs at factory level", and Task 3.2 "Collaborative DTs supporting value networks" during their first iteration, up to project month 18. Both tasks have made significant progress towards achieving their respective objectives. Task 3.1 focuses on implementing the AAS Model Generator, as defined in the overall T4M architecture outlined in Deliverable D2.5. To this end, the Fraunhofer Advanced Asset Administration Shell Tool (FA³ST) CreAItor, a novel and user-friendly tool enabling non-expert users to generate standard-compliant DTs from their input data based on Large Language Models, is proposed. Additionally, task 3.2 covers implementing the Supply Chain Monitoring as defined in the overall T4M architecture, for which Multi-level Integration for Real-time Applications (MIRA) serves as the basis.

As part of Task 3.1, a comprehensive literature review was conducted, and example cases were analyzed to identify challenges that need to be addressed during the development of the FA³ST CreAItor architecture. The main challenge arises from the insufficient domain knowledge of the Large Language Models. This knowledge is required for identifying and extracting information from the input data to generate the respective DTs. To address this, a Context Provider component was introduced to the architecture of the FA³ST CreAItor. This component provides the required knowledge based on domain-specific vocabularies and ontologies. However, a first version of the FA³ST CreAItor was implemented based on the proposed architecture as, supporting two out of four intended Submodel Templates (SMTs): the Digital Nameplate for Industrial Equipment, and the Capability Description SMT. This initial version was evaluated by an expert group with a focus on usability. The next steps include refining the user interface of the FA³ST CreAItor before the midterm review and ensuring full support for all SMTs required for the DTs in the Tec4MaaSEs project during the second iteration.

Regarding Task 3.2, the background in modelling supply chain DTs through MIRA is presented, followed by the innovative proposal of the project by combining the monitoring features of MIRA with the matching features of the Tec4MaaSEs marketplace. The proposed approach is discussed, and an overview of the current development status of MIRA is provided, along with a set of screenshots showcasing some of the main features and a thorough explanation of future features that are currently in design. Finally, an analysis on the level of integration with AAS-compliant assets generated on factory level is conducted, accompanied by a listing of submodel (SM) elements that will be supported in MIRA by the end of the project.

Table of Contents

List of Tables

List of Figures

Acronyms and Abbreviations

Acronym	Description
AAS	Asset Administration Shell
AI	Artificial Intelligence
CSS	Capabilities, Skills, and Services
DT	Digital Twin
FA³ST	Fraunhofer Advanced Asset Administration Shell Tools
IEEE	Institute of Electrical and Electronics Engineers
LLM	Large Language Model
MaaS	Manufacturing as a Service
MIRA	Multi-level Integration for Real-time Applications
SM	Submodel
SMT	Submodel Template
T4M	Tec4MaaSEs
UEQ	User Experience Questionnaire
WP	Work Package

1 Introduction

1.1 Purpose and Scope

The purpose of Deliverable 3.1 is to report on the progress and results of Tasks 3.1 and 3.2 within the scope of Work Package (WP) 3. It provides a comprehensive overview of the activities carried out by the project partners in the development of methods and tools aimed at supporting both the instantiation of Asset Administration Shell (AAS) models at the factory level and their on-demand interaction within value networks. This document covers both conceptual foundations and practical insights, highlighting the current implementation status of the developed tools in alignment with the objectives defined for the respective tasks.

1.2 Relation to other Deliverables

This deliverable is closely linked to several other WPs, in particular WP 2 Task 2.4 and Task 2.5, WP 4, and WP 5. WP 2 Task 2.4 provides relevant information for Task 3.1, Task 2.5 outlines the overarching architecture of the Tec4MaaSEs (T4M) system, while WP 4 concentrates on the integration of its components. Additionally, the outcomes of WP 5 support the pilots in developing pilot-specific AAS models. Furthermore, the developed tools enable on-demand interaction within the value networks of the respective pilots. The work conducted in Tasks 3.1 and 3.2 builds a foundation for the application and demonstration activities in the pilots and provides key technical enablers for their implementation.

1.3 Structure of the Document

This document is structured as follows:

- **Section 2** presents the activities and results of Task 3.1, “DTs at factory level”. Task 3.1 focuses on implementing the AAS Model Generator, as defined in the overall T4M architecture outlined in Deliverable D2.5. To this end, the Fraunhofer Advanced Asset Administration Shell Tool (FA^{3}ST) CreAItor is presented. This includes a comprehensive literature review aiming at identifying existing approaches relevant to the generation of DTs at the factory level. Furthermore, the section provides more details on the architecture of the FA^{3}ST CreAItor, followed by an overview of its current implementation.
- **Section 3** is dedicated to Task 3.2, “Collaborative DTs supporting value networks”. This section documents the progress in Supply Chain Monitoring as defined in the overall T4M architecture, beginning from the background upon which the task was designed, covering the proposed approach and finally describing the implementation of current features.
- **Section 4** concludes the deliverable with a summary of key findings and lessons learned. It also provides an outlook on the next steps to be taken within the scope of Tasks 3.1 and 3.2.

2 DTs at the Factory Level

2.1 Introduction

This section reports on activities, conducted work and preliminary outcomes related to Task 3.1. The aim of this task is to develop user-friendly methods and tools for creating AAS models at factory level to be used in Manufacturing as a Service (MaaS) scenario. Therefore, the work builds on the existing AAS *metamodel*[1] and standardized *Submodel Templates* (SMTs)[2] like the *Capability Description*[3] template. During T3.1, the FA³ST CreAItor has been developed. It automatically generates and instantiates AAS-based DTs while ensuring their compliance with the AAS standard. This tool is designed to: 1) assist end users in the AAS creation process via a user-friendly interface, enabling them to create AAS models automatically without any coding; and 2) integrate state-of-the-art semantic models and knowledge graphs to guide end users and to improve the quality of the generated AAS models. Considering recent advancements in Artificial Intelligence (AI), particularly in the field of Large Language Models (LLMs), the core idea underlying the FA³ST CreAItor is to leverage LLMs to support its users during the generation of AAS-based DTs.

Deliverable 3.1 reports on the activities and outcomes of T3.1 up to project month 18. During this first iteration, the focus was on developing the conceptual foundation and implementing the first version of the FA³ST CreAItor. The conceptual work included a comprehensive literature review to identify existing approaches and tools, clarify their advantages and disadvantages, and analyze their implications for the proposed approach. Additionally, the conceptual work included the specification of the architecture of the FA³ST CreAItor, as well as the definition of specific approaches for each relevant AAS SMT.

The remainder of this section is organized as follows. First, a literature review is presented, outlining existing approaches and tools, as well as providing the conceptual bases for the following sections. Next, the conceptual foundation is described in more detail. This is followed by an introduction of the first implementations. Initial evaluation results are then discussed, based on standardized questionnaires assessing overall user experience. Finally, the section concludes with a discussion of the challenges encountered and lessons learned.

[1] Please refer to the IDTA specification of the Asset Administration Shell Part 1: Metamodel.

[2] A list of all SMTs is provided in the Content Hub hosted by the IDTA.

[3] Please note that this SMT is currently under development and is not available to the general public. Once published, it will be accessible via the Content Hub.

2.2 Literature Review

2.2.1 State of the Art

As a first step toward achieving the objectives of T3.1, a comprehensive literature review was conducted to identify AI-based approaches relevant for the instantiation of AAS models. The goal of this review was to explore existing AI methods, particularly those leveraging LLMs, that could serve as a foundation for the activities in T3.1.

To ensure a high-quality literature review, the scope was limited to databases containing only peer-reviewed research papers. Consequently, the databases were restricted to Institute of Electrical and Electronics Engineers (IEEE) Xplore, Scopus, and Web of Science. Each database was queried for articles referencing the term *Asset Administration Shell* in the title, abstract, or keywords. Additionally, to ensure that all relevant AI-related publications are retrieved, the databases were queried with the same terms, e.g., machine *learning*, *artificial intelligence*, *AI*. In total, 144 articles were identified as relevant. After removing duplicates and articles out of scope 36 publications remained. To identify articles related to the scope of T3.1, specifically those that support the modeling and generation of AAS models, each publication was categorized based on its respective focus within the AAS lifecycle.[4] The results are summarized in Figure 1.

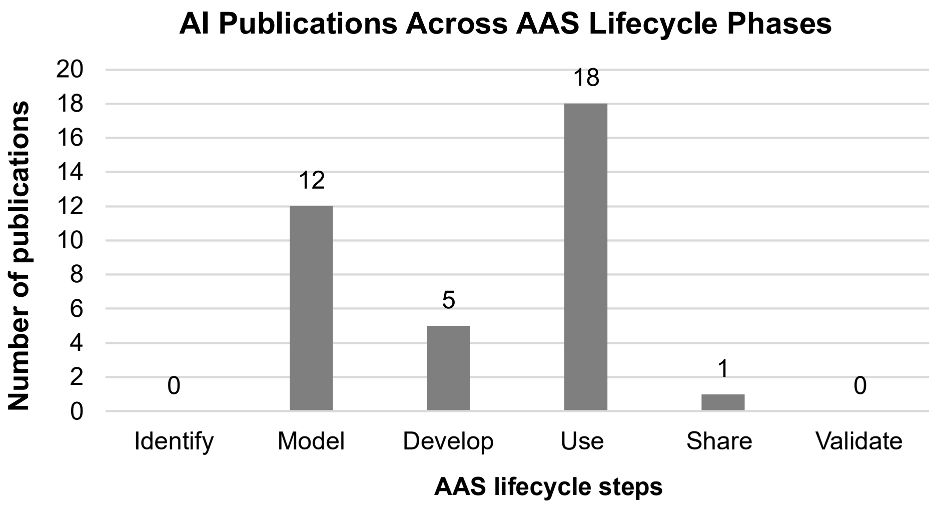

Figure 1: Overview of AI Related Publications Mapped to the AAS lifecycle

As the aim of T3.1 is to support the creation of AAS models, only articles related to the *Model* step of the AAS lifecycle, were considered as relevant. Most of these articles discuss suitable SMTs for modeling AI models or agents with the AAS (cf. [2] for production scheduling, [3] for predictive maintenance, or [4] for a survey on AI integrated DTs across different fields) and are therefore not relevant within the scope of T3.1. Only a few articles focus on using AI models to support the actual AAS modeling process: [5] introduces a Retrieval-Augmented Generation approach that leverages an LLM to produce AAS-compliant JSON outputs. The process begins by populating a database with vectorized AAS JSON snippets derived from product descriptions. Given a new product description, semantic search is used to identify the most relevant entity from the database. In the final step, both the retrieved JSON snippet and the new description are passed to a GPT instance, which generates

[4] The AAS lifecycle is composed of the following phases: Identify, Model, Develop, Use, Share, and Validate. For a detailed description of the AAS lifecycle, please refer to [1].

an AAS JSON representation of the product description. This approach demonstrates the general potential of LLMs to support the instantiation of SMTs. However, it appears not suited for the objectives of T3.1. The method is highly computationally intensive, as it is used to directly generate the JSON representation of an AAS model. As these JSON representations typically contain multiple thousand lines of code, this method is also very costly, as LLMs are typically charged by the amount of generated output. Moreover, the transformation of structured information into AAS-compliant JSON can be performed more reliably and cost-effectively using rule-based approaches.

In contrast, the authors of [6] present an approach to automatically extracting information entities from existing data sources such as PDFs, Excel files, or JSON. These entities are then processed through an LLM-based semantic matching engine in two stages. First, each extracted entity is matched to an AAS element defined in the corresponding AAS SMT. In the next step, these elements are linked to standardized Concept Descriptions from sources like ECLASS[5]. Unlike [5], this approach does not use LLMs to directly generate the JSON representation of AAS models from unstructured input. Instead, LLMs are used to map the input to the best matching AAS SM element. However, the approach has several limitations. Firstly, the authors focus exclusively on the instantiation of SM properties based on existing dictionaries, like ECLASS. They do not cover the instantiation of complex nested structures, like required for the Capability Description SMT. Secondly, the approach does not cover the extraction of relationships (like *generalizedBy*, *constraints*, or *composedOf* relationships) between the respective SM elements which are required for the Capability Description SMT. Lastly, the authors claim that the semantic matching engine is error prone.

While the approaches presented demonstrate the general suitability of LLMs for automated SMT instantiation, they are only partially applicable to T3.1. Nevertheless, they provide initial insights for the activities carried out within T3.1. First, using LLMs to directly convert the extracted information into the AAS format is considered error-prone and computationally intensive. This can be more efficiently addressed using rule-based approaches. Second, external standardized data sources can be used to enrich the information extraction process with additional domain knowledge. However, relying solely on standardized dictionaries such as ECLASS does not provide sufficient information about nested structures and relationships. Finally, LLM-based information extraction itself is prone to errors. Therefore, the extracted data and SMT elements must be validated before the SMT is instantiated. These issues are addressed by approach proposed in T3.1, which will be further detailed in Section 2.3.

2.2.2 Overview and Challenges of the Capability Description SMT

As the descriptions of Capabilities are essential for the registration, searching and matching functionalities in a Manufacturing as a Service platform, the Capability Description SMT is particularly important for T4M. Therefore, one of the main objectives of T3.1, is to support non-expert users during the instantiation of the Capability Description SMT. To this end, this subsection outlines the theoretical foundations of the SMT. Please note this SMT is currently still under development and is therefore not available to the general public. However, as FhG is part of the corresponding working group defining the Capability Description SMT, the preliminary version of the SMT will be applied during the first iteration of T3.1 (Please refer to Deliverable D6.4 for more information about the status and activities conducted by FhG regarding this working group).

The reference model proposed in [7] for the description of Capabilities, Skills, and Services (CSS) of production serves as the basis for the formulation of the Capability Description SMT. Within the context of the CSS-Model

[5] For more information about ECLASS please refer to this webpage.

and thus also in the Capability Description SMT, Capabilities are defined as "An implementation-independent specification of a function in industrial production to achieve an effect in the physical or virtual world" [7, p. 97]. A detailed description of the status and pending challenges of the SMT can be found in D6.4.

In the context of capability modeling, two principal approaches are distinguished: *product-based* and *process-based* capability modeling [8]. Product-based capability modeling represents capabilities as standardized STEP features. This approach allows for a direct mapping between required and provided capabilities, eliminating the need to reference specific processes. While this facilitates one-to-one capability matching and narrows the solution space, it also demands significantly greater modeling effort [8]. In contrast, process-based capability modeling defines capabilities in terms of standardized industry processes, such as drilling as specified in ISO 8636-1:2000. This approach requires less modeling effort and allows for greater flexibility in capability matching. However, the broader solution space can make it challenging to precisely match required and provided capabilities. For instance, different processes—such as form milling and inside milling—may produce equivalent outcomes, complicating the identification of suitable capability providers [8]. Within the context of T3.1 the activities focused on process-based capabilities modeling, to not overwhelm non-expert users with the modeling requirements. Additionally, the authors of [9] propose a workflow for determining capabilities from existing data sources based on the CSS Model which will guide the conceptual activities during T3.1. This workflow is detailed in Figure 2. This process starts the derivation of capabilities relevant for the considered context. To achieve this, data, like product descriptions, could be used. Next, based on this outcome, specific capabilities need to be assigned to the specific manufacturing resources. Once they are assigned, their actual capability descriptions can be instantiated using respective data from the resources. In the last step, the instantiated Capability Description needs to be reviewed and updated if applicable.

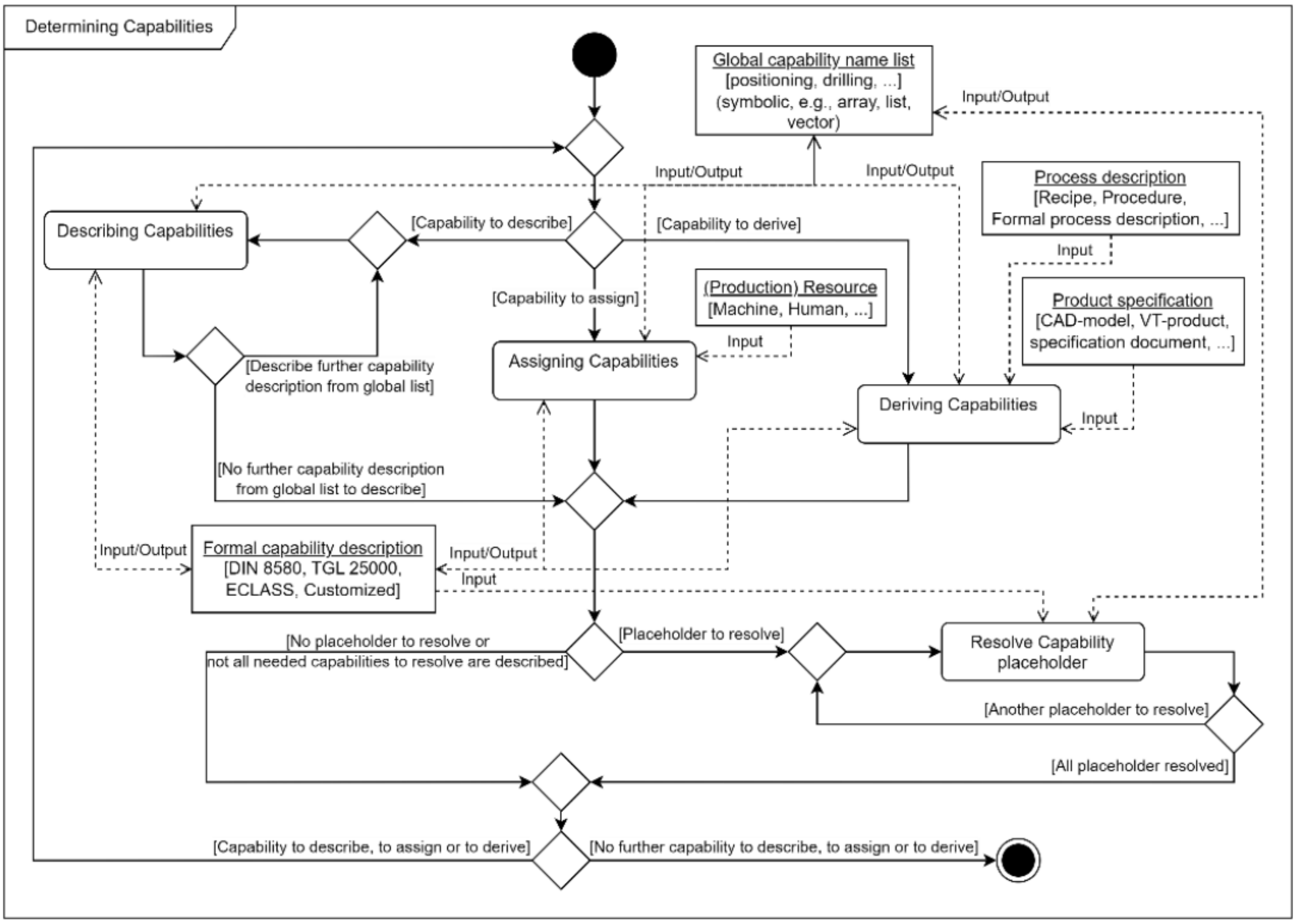

Figure 2: Deriving Capabilities. Taken from [9, p. 4]

2.2.3 Overview of Existing Tools

This section provides an overview of existing tools focusing on the AI-based instantiation of AAS models or Capability Descriptions. As already highlighted in Section 2.2.1, only 12 relevant publications could be identified in the literature. Accordingly, only two existing tools were found that directly address the AI-supported instantiation of AAS elements or Capability Descriptions. Therefore, Table 1 presents the *AASbyLLM*[6] and *LLM4Cap*[7] tools, outlining their core functionalities, known limitations, and their respective relevance within the T4M context. While both tools utilize LLMs, they target different aspects.

[6] For more details about AASbyLLM, please refer to this GitHub repository.

[7] For more details about LLM4Cap, please refer to this GitHub repository.

Table 1: Overview of Existing Tools

Tool	Description	Limitations	Relevance in T4M
AASbyLLM	AASbyLLM leverages LLMs to instantiate the Technical Data SMT from unstructured textual input. The tool includes a user interface that allows users to choose from multiple LLMs for processing textual input.	AASbyLLM faces two major limitations. First, it only supports input texts of up to 600 characters, which restricts its practical usability. Second, the tool is limited to generating instances based on an outdated version of the Technical Data SMT and only includes property elements.	Given these limitations, AASbyLLM is not considered relevant within the T4M context. This approach allows only to instantiate Property SM elements and therefore does not provide the flexibility needed for the Capability Description SMT.
LLM4Cap	LLM4Cap is a tool designed to automatically generate capability ontologies from unstructured textual input using LLMs. The resulting ontology is automatically validated through syntax checks and SHACL validation to ensure structural correctness.	The tool employs predefined prompt templates to structure the input text for the LLM. These templates provide contextual information, including an ontology description. However, the inclusion of rich context leads to a high token count, which increases computational costs.	This tool can support T4M participants in the creation of capability ontologies. This ontology can then be used by the FA³ST CreAItor to improve the automated instantiation of the Capability Description SMT.

The tools identified in this review represent early-stage efforts to use LLMs for the automated creation of AAS models and capability descriptions. The first tool, *AASbyLLM*, was developed to instantiate the Technical Data SMT based on structured textual input, such as product descriptions or data sheets. However, the tool only supports textual input restricted to 600 characters, which limits the practical use since technical descriptions are often longer. Additionally, the tool only supports the instantiation of Property SM elements, but not other element types such as Relationship or Range SM elements. Since the instantiation of more complex SMTs requires the use of these additional elements, this further limits the applicability of the tool. As a result, *AASbyLLM* is not considered relevant for the T4M project.

The second tool, *LLM4Cap*, focuses primarily on the generation of capability ontologies. It is designed to extract capability information from textual input and automatically generate a corresponding capability ontology. The tool provides syntax validation mechanisms to ensure the structural correctness of the generated ontologies. A key feature of *LLM4Cap* is its use of prompt templates that specify the structure of the CaSk ontology[8] in RDF syntax. However, using LLMs to directly generate RDF syntax results in a large amount of output generated. This is associated with high costs or high computational effort. Despite this limitation, *LLM4Cap* is considered potentially relevant in the T4M context, as it can support the generation of domain-specific capability ontologies. These domain specific ontologies can for example be used by the FA³ST CreAItor to better adapt the generated AAS models to specific domains.

2.2.4 Conclusion

This review has highlighted the current state of AI-based approaches for the creation and instantiation of AAS models, with a particular focus on LLM-supported methods. From an initial pool of 144 relevant publications, only a small subset directly addresses the specific objective of T3.1. The approach presented in [5] employs a Retrieval-Augmented Generation pipeline to generate AAS-compliant JSON representations from textual product descriptions. This approach demonstrates the technical feasibility of applying LLMs to AAS model

[8] Please refer to this GitHub repository for more information about the ontology.

generation. However, due to the high volume of generated output, the approach is both prone to hallucinations and computational demanding. Thus, the approach is not considered. A second approach presented in [6] details the extraction of SM properties from unstructured textual input. Additionally, the approach maps the extracted properties to standardized dictionaries, e.g., ECLASS. Although more aligned with T3.1's goals, this approach does not support the instantiation of complex nested SMT structures, or the extraction of relationships between the identified properties.

Nevertheless, the review provided first insights for T3.1. First, directly converting extracted information into the AAS format using LLMs is error-prone and computationally intensive. Second, external data sources can be used to provide guidelines for the information extraction. Finally, the extracted information needs to be evaluated by domain experts to ensure accuracy and to avoid hallucinations from the LLMs. Additionally, the workflow presented in [9] provides a practical and structured approach for determining capability descriptions from existing data sources. It outlines a step-by-step process, from identifying relevant capabilities within the considered domain to instantiate formalized capability descriptions. This workflow can effectively support the conceptual development activities within T3.1.

2.3 Proposed Approach

Using the insights from the literature review and the workflow presented in [9], the FA³ST CreAItor and its architecture outlined in this chapter is proposed to address the objectives of T3.1. Please note, that the FA³ST CreAItor represents the AAS Model Generator, as defined in the overall T4M architecture outlined in Deliverable D2.5. The FA³ST CreAItor is designed to semi-automatically instantiate SMTs from input data, to support T4M participants with the generation of standard compliant AAS models. It provides semi-automated support not only for the typical AAS SMTs like Digital Nameplate[9], but also for SMTs specific to Manufacturing as a Service scenarios like the Capability Description SMT. Table 2 provides an overview of the respective SMTs, their relevance within T4M, and their associated challenges. Additionally, the table specifies their expected availability within the FA³ST CreAItor. As these SMTs strongly differ in their complexity, the instantiation approaches need to be adapted for each SMT. Therefore, within the first iteration of T3.1 the activities focused on supporting the instantiation of the Digital Nameplate SMT as well as the Capability Description SMT. The remaining SMTs listed in Table 2 will be addressed in the second iteration of T3.1. Additionally, the inclusion of further SMTs (e.g., Purchase Order Creation[10]) in the scope of the second iteration will be evaluated.

Table 2: Overview of Relevant SMTs

SMT	Relevance for T4M	Challenge for the FA³ST CreAItor	Version
Digital Nameplate for Industrial Equipment	This SMT describes standardized meta data of specific manufacturing resources and can support the registration functionalities in T4M.	The main challenge for this SMT is to extract information entities from the input data based on the elements and descriptions provided in the SMT.	First iteration

[9] For more information about this SMT please see the *Digital Nameplate for Industrial Equipment* SMT at the IDTA Content Hub or at the respective GitHub repository.

[10] For more information about the Purchase Order Creation SMT please visit this website.

SMT	Relevance for T4M	Challenge for the FA³ST CreAItor	Version
Generic Frame for Technical Data for Industrial Equipment in Manufacturing	The SMT describes the technical data of manufacturing resources, ideally by referencing standardized dictionaries like ECLASS. This information can, e.g., be used for searching and matching within T4M.	There are three main challenges associated with this SMT. First, relevant data must be identified and extracted. Second, this SMT provides only abstract definitions. Next, the extracted data needs to be structured into semantic groups. These groups must then be mapped to the respective AAS elements.	Second iteration
Capability Description	This SMT describes the capabilities of manufacturing resources and will be used for registration, searching and matching functionalities in T4M.	Loosely specified template with a high degree of freedom including relationships between different elements. Missing domain knowledge of LLMs hinders information extraction. Like for Technical Data, direct the identification of relevant information entities is considered as the main challenge.	First iteration
Production Calendar	This SMT describes unified production calendars that are required for the production scheduling and capacity allocation in T4M.	Unlike for the other SMTs, for this SMT dynamic data that require frequent updates need to be included during the instantiation.	Second iteration

In the remainder of this section, a manual SMT instantiation example is first presented using the Capability Description SMT and data provided by T2.4. The aim of this section is to identify challenges for the design of the overall system architecture. Next, the architecture of the FA³ST CreAItor is introduced, highlighting architectural components that address the identified challenges. Lastly, the interactions between the different components of the architecture are presented using the same example as during the description of the manual instantiation process.

2.3.1 Manual Instantiation of the Capability Description SMT

As stated in the introduction of this chapter, this section showcases the manual instantiation process of the Capability Description SMT using an example from Value Network 2 to identify challenges for the FA³ST CreAItor. Within Value Network 2 capability descriptions are, e.g., needed during the onboarding of manufacturing service providers (please refer to deliverable 2.1 for more information about the onboarding process). To showcase the instantiation of the SMT, this section follows the proposed process for determining Capabilities as outlined in [9]. In the first step, before instantiating the Capability Description SMT, relevant capabilities and their respective describing elements must be identified. In the context of Value Network 2, this has largely been provided as part of T2.4. Besides defining Value Network 2's specific capabilities, the outcomes also detail their corresponding describing elements (e.g., axes, travel distance, and tolerance). Using this as a basis for the instantiation of the SMT, the next step is to identify a specific machine for which capabilities should be described. In this context, Tekniker's *Goratu G Dynamic* has been selected. Table 3 provides a short overview of the Goratu G Dynamic machine and its describing elements.

Table 3: Data of the Goratu G Dynamic Machine, Provided by T2.4

Reference	Goratu G Dynamic
Manufacturer	Goratu
Type	Milling Machine

Reference		Goratu G Dynamic
Control mode		CNC
Primary operation		Milling
Secondary operation		Drilling
		Threading
Axes (Number)		5
Travel distances (mm)	X length	700
	Y length	600
	Z length	500
Maximum speed (rpm)		18000
Tolerance (mm)		0,01
Workpiece max weight [kg]		1450
Material compatibility		P, P1, P2, P3, P4, P5, P6, M, M1, M2, M3 M4, M5, K, K1, K2, K3, K4, N, N1, N2, N3, N4, N5, N6, S, S1, S2, S3, S4, S5, H, H1, H2, H3, H4, H5
Explosive materials		No
Inert environment		No
Max available tool length[mm]		105
Min available tool diameter [mm]		6

The next step as outlined in [9] is to assign capabilities to the specific machine. As can be seen in Table 3, the Goratu G Dynamic machine provides a total number of three possible operations, each representing a capability according to the definition introduced in Section 2.2.2. While the main (primary) capability of the Goratu Machine is milling, it provides two additional (secondary) capabilities: drilling and threading. According to the outcomes of Task 2.4, these three capabilities share the same describing elements. Once the capabilities and their describing elements are identified, the actual Capability Description SMT can be instantiated. Therefore, the information from Table 3 is converted into the structure outlined in the SMT. First, according to the Capability Description SMT, for each capability (milling, drilling, threading), a separate capability container was created. According to the SMT these containers include, besides the actual capability, an additional capability comment, a capability property set (containing the describing elements of the capability), and information about capability relations. Additional qualifiers were assigned to each capability. These qualifiers are used to specify whether the capability is a primary or secondary machine operation. Additionally, for better human readability, the capability comments include the semantic description of the respective capability. These semantic descriptions are based on industry standards, e.g., on ISO 8636-1:2000[11]. In line with the SMT, the property set includes information elements (e.g., axes, travel distance, and tolerance) of the capability. For each information element described in Table 3 separate property containers must be created. This property container should include, besides the actual information element, a property comment, and an optional relationship element. Each respective information element was either instantiated as a property (containing only one value), range (specifying an upper and lower boundary for possible values), a multilanguage property (containing textual information translated into different languages), or as a list (containing multiple information elements). Additionally, for better human readability, the respective property comment defines a semantic description derived from industry standards for the information element. The optional relationship element can be used to specify that the information element is already described in the Technical Data SM. Lastly the capability relations were used to describe information about the relationships of the three assigned capabilities. According to the SMT these relations can include hierarchical relationships between the individual capabilities, constraints for

[11] For more information about the ISO 8636-1:2000 please refer to this webpage.

their respective information elements, and references to machine-specific implementations of the capability. Regarding Value Network 2, references to machine-specific capability implementations are not required. Therefore, they are not considered during the instantiation of the SMT. Additionally, as no restrictions are specified for the information provided in Table 3, constraints are also not considered during the instantiation. However, following the output provided by T2.4 milling, drilling, and threading can be generalized by a machining capability. Therefore, this capability also needs to be instantiated. As this capability can neither be considered as a primary nor as secondary capability, a new capability qualifier needs to be introduced to specify the capability as abstract. Once instantiated, hierarchical relationships can be included for each of the three capabilities. The instantiated SM is depicted in Figure 3.

This example showcases the complexity of the instantiation process. The structure of the machine-specific Capability Description needs to be specified (e.g., defining the property set, excluding constraints) based on the abstract structure provided in the SMT. Additional capability qualifiers need to indicate the operation type of the respective capability. Furthermore, during the instantiation process, close collaboration with Tekniker was necessary to ensure that the defined qualifiers, comments, properties, and relationships accurately reflect the manufacturing resource. This demonstrates that the instantiation of the Capability Description SMT requires a deep understanding of the AAS specification and domain expertise.

Figure 3: Part of the Capability Description SM for the Goratu G Dynamic Machine

2.3.2 Architecture of the FA³ST CreAItor

While some of the SMTs, like Digital Nameplate (detailed in Table 2), provide a clear structure with well-defined SMT elements, other SMTs, like Capability Description, only provide an abstract structure that needs to be applied to the specific manufacturing resource. Additionally, these generic SMTs only provide abstract definitions for their respective SMT elements. This poses several challenges for the architecture of the FA³ST CreAItor:

- First, relevant data for the respective manufacturing resource needs to be identified and extracted from the input data. This requires resource-specific domain knowledge.
- Second, the provision of this domain knowledge is considered as challenge. This domain knowledge is required for guiding the information extraction, and for applying the abstract SMT structure to the actual manufacturing resource.
- Third, to ensure that the extracted data reflects the actual manufacturing resource, the extracted information needs to be evaluated by the user before the SMT is instantiated.
- Finally, to ensure that the FA³ST CreAItor can be extended to support additional SMTs in the future, the architecture must be designed to be modular and expandable.

To address these challenges, the following architecture for the FA³ST CreAItor, shown in Figure 4, is proposed.

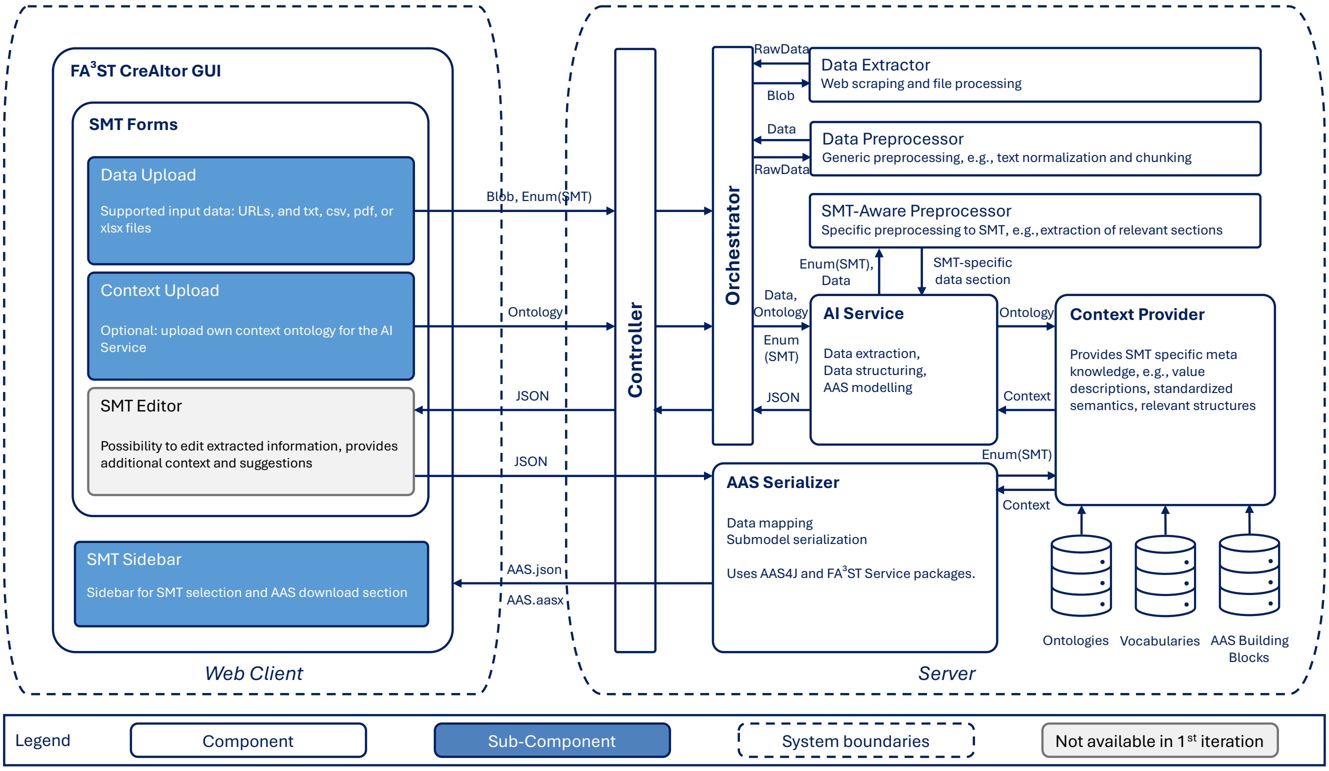

Figure 4: Architecture of the FA³ST CreAItor

To address the identified challenges, the following components were introduced:

- **AI Service**: The AI Service utilizes LLMs to address the first challenge: information extraction. Additionally, it uses information provided from the Context Provider and the SMT-Aware Preprocessor to define the context for the SMT specific information extraction and to reduce hallucinations. Currently, the AI Service supports LLMs that are accessible via the OpenAI API[12], or that are locally hosted via Ollama[13]. The component is designed to be extendable, allowing the integration of further SMT-specific extraction approaches and LLM providers in the future.
- **SMT-Aware Preprocessor**: The SMT-Aware Preprocessor addresses the first challenge. It uses both AI-based and rule-based preprocessing to extract relevant sections from the input data, thereby narrowing the search space for the AI service during information extraction.
- **Context Provider**: This component addresses the second challenge, by providing domain knowledge to other components, if needed. Therefore, the component uses domain-specific vocabularies or ontologies. Additionally, the component provides SMT-specific meta data, e.g., SMT element descriptions, or standardized semantics stored in a respective extendable database.
- **SMT Forms**: This component tackles the third challenge, by allowing users to review, edit, and delete the extracted SMT elements and their respective values. In the first iteration of T3.1, this component does not support user forms for complex SMTs, due to their dynamic structure.

While the components detailed above are designed to address the identified challenges, additional components were introduced to allow flexibility and extendibility of the system:

[12] For more information about the OpenAI API please refer to this website.

[13] For more information about Ollama please refer to this website.

- **Controller**: This component manages communication between the client and the server.
- **Orchestrator**: The component coordinates the SMT instantiation tasks on the server side. It further enables the integration of additional backend components without the need to adapt the interaction between the client and the server.
- **Data Extractor**: This component was included to handle the different input data formats. Currently this -component supports URL, PDF, TXT, and CSV as input formats but can be extended to additional data formats.
- **Data Preprocessor**: This component enables generic preprocessing, e.g., lemmatization, of the input data.
- **AAS Serializer**: The AAS Serializer is used to instantiate the respective SMT using the reviewed data and additional SMT-specific meta data provided by the Context Provider. Additionally, the component includes the required concept descriptions into the AAS model.
- **FA³ST CreAItor GUI**: This component provides the user interface of the FA³ST CreAItor. This user interface includes a respective view for each SMT. Additionally it allows navigating among the available SMTs and downloading the generated AAS model.

2.3.3 Interaction Sequence of the FA³ST CreAItor Components

This section showcases the interactions between individual components of the FA³ST CreAItor architecture based on the example presented in Section 2.3.1. As the system's interactions follow a similar pattern for all supported SMTs, it is considered sufficient to describe them using the Capability Description SMT as a representative example. The interaction flow is described in the sequence diagram shown in Figure 5.

First, the input data is provided, and the instantiation process is started by the user via the FA³ST CreAItor GUI. The input data, as well as information about the respective SMT that should be instantiated is passed from the client to the server using the Controller component. This component redirects the received information to the Orchestrator, which then initiates the preprocessing of the provided input data. Therefore, the Orchestrator passes the input data to the Data Extractor component. The Data Extractor converts the input into an internal representation and returns this raw data to the Orchestrator, which directs it to the Data Preprocessor. The Data Preprocessor applies generic preprocessing, like lemmatization and returns the preprocessed data to the Orchestrator, which initiates the information extraction. Therefore, the preprocessed data together with the SMT type is passed to the AI Service. This component first requests relevant data from the preprocessed data through the SMT-Aware Preprocessor using the SMT type. Next the AI Service requests additional context from the Context Provider based on the provided SMT type. In the context of the Capability Description SMT, the AI Service first requests a list of capabilities from the Context Provider. The Context Provider uses the data stored in the respective vocabularies to provide this information. In the next step, the AI Service uses this context to extract a set of capabilities (e.g., milling, drilling, and threading) present in the relevant data. Subsequently, this extracted capability set is used to request more information about each capability from the Context Provider. This information is retrieved from an ontology and contains, e.g., describing elements like axes or tolerance, and structural information, like generalizedBy relationships. In this example, an ontology specifically developed during T3.1 for Value Network 2 is used by the system.

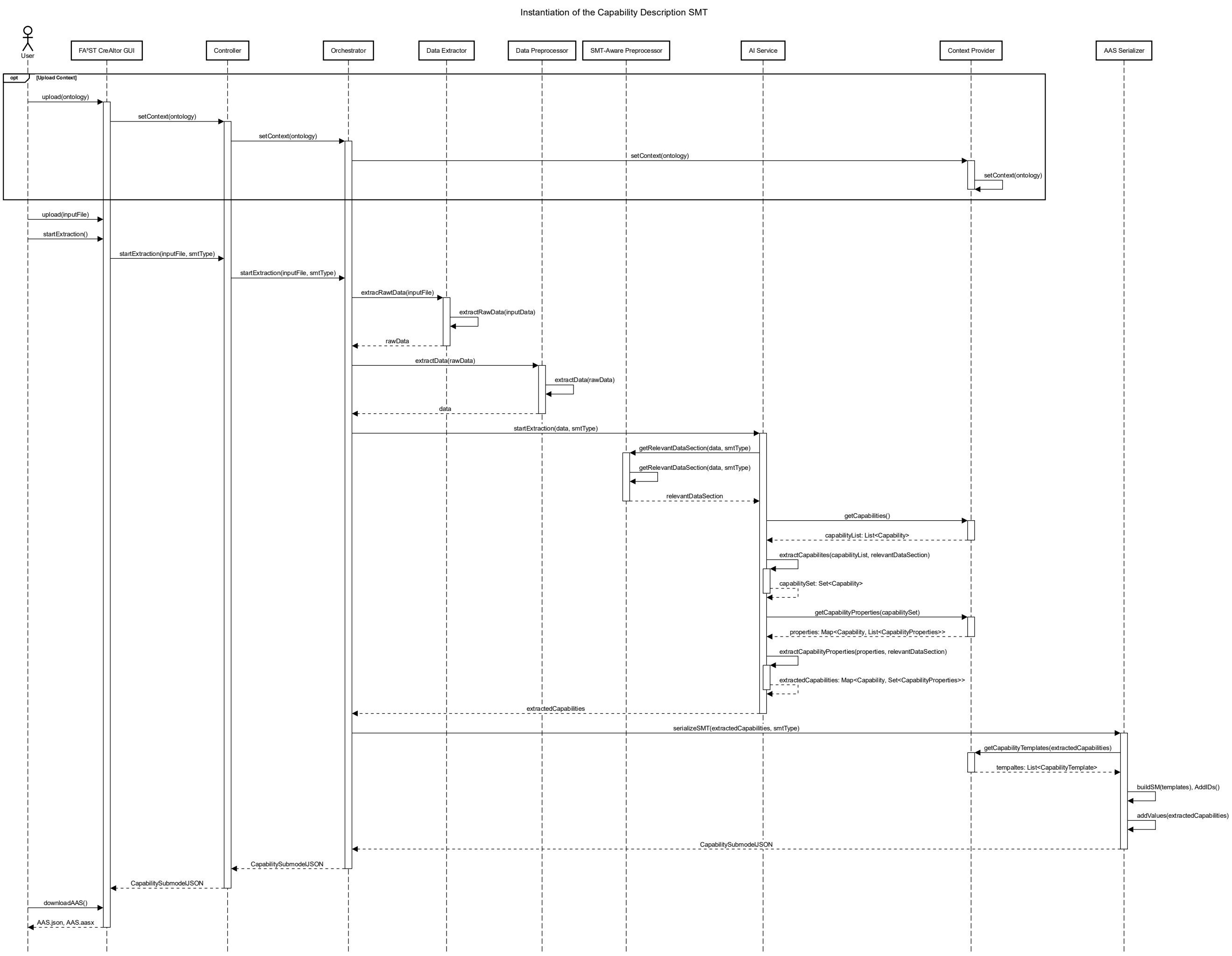

Figure 5: Interactions Between the Components of the FA³ST CreAItor

This ontology is based on the results of T2.4 and defines capabilities and their respective describing elements for the manufacturing resources involved in Value Network 2. Figure 6 provides an overview of this ontology. Please note that by the end of the second iteration of T3.1 users of the FA³ST CreAItor can upload their own ontologies to adapt the requested context to their specific domain. The AI Service uses the additional context provided by the Context Provider to extract specific values for each property of the capability. The resulting capabilities, including the extracted values for their respective properties, are then returned to the Orchestrator, which then starts the actual instantiation of the SMT. Therefore, the extracted information, as well as the SMT type, is passed to the AAS Serializer. The Serializer first requests additional context about the SMT from the Context Provider to build templates for the instantiation of the SMT. Next, the AAS Serializer uses these templates to instantiate the SMT in JSON format and add the corresponding extracted information. This capability SM JSON is returned to the Orchestrator, which subsequently returns it to the Controller component. The Controller returns the capability SM JSON to the client signaling that the instantiation process is finished. The user can now download the fully instantiated SMT using the FA³ST CreAItor GUI.

Besides demonstrating the interactions between the components of the FA³ST CreAItor, this example showcases the system's ability to simplify the instantiation of complex SMTs. Without the FA³ST CreAItor, instantiation is time-consuming and requires both deep domain knowledge and expertise in the AAS specification. By using the CreAItor, users only need to review and edit the extracted values before the SMT is instantiated. This clearly reduces the required effort and minimizes the need for in-depth AAS expertise.

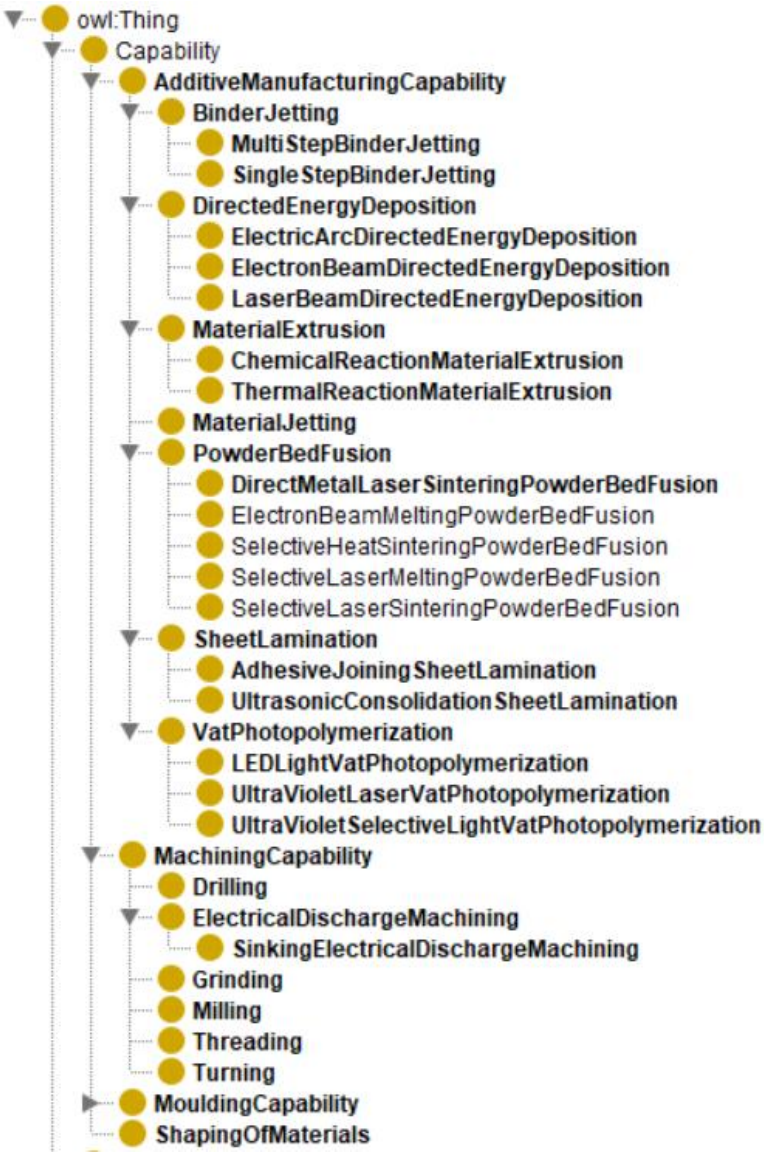

Figure 6: Class hierarchy of the Ontology Developed for VN2

2.4 Implementation of the FA³ST CreAItor

This section presents the current implementation of the FA³ST CreAItor based on the architecture outlined in Chapter 2.3. The implemented version of the FA³ST CreAItor is explained step by step using selected screenshots from the system interface. In each screenshot, the different sections of the system are highlighted in red. Figure 7 shows the start screen of the FA³ST CreAItor. Please note that the FA³ST CreAItor is provided to the project partners as Docker images and can be retrieved from the corresponding T3.1 folder in the project repository.

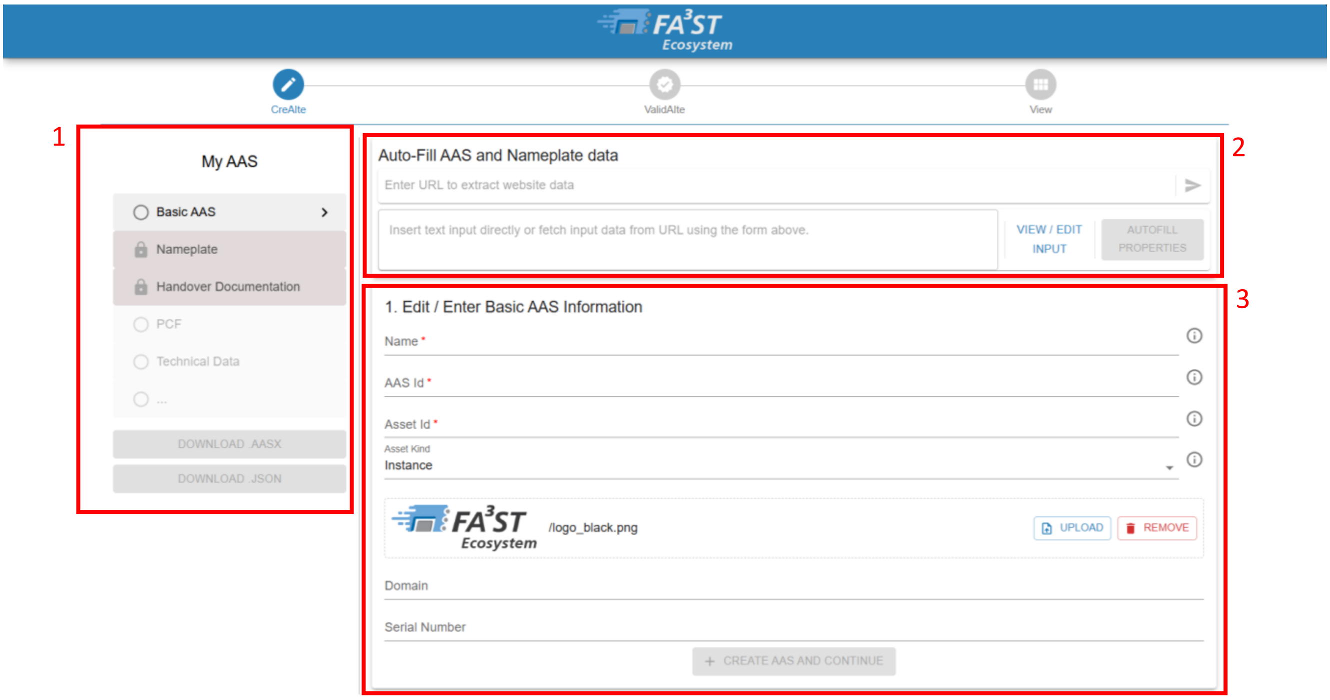

Figure 7: Start Screen of the FA³ST CreAItor

The start screen is divided into three main sections. In the first section on the left-hand side, users can select the SMTs[14] they wish to instantiate. It is important to note that when creating a new AAS model, users must first provide basic AAS information (e.g., the name of the AAS model, or the ID of the asset which is represented) before SMT instantiation becomes available. Until the basic AAS information is provided all supported SMTs are marked as locked.

In the second section, on the right-hand side users are prompted to provide input data, either by providing a URL, textual input, or uploading supported files. The latest version of the FA³ST CreAItor currently supports CSV, PDF, and TXT file formats. If a URL is provided, the FA³ST CreAItor employs a web scraper to extract relevant content from the referenced web page. Otherwise, the information is directly processed through an LLM-based extraction pipeline. Once the extraction is complete, the FA³ST CreAItor notifies the user whether the process was successful. In case of success, the extracted information is not immediately displayed to the user. Instead, it is used to automatically populate the properties of the Nameplate and Handover Documentation SMTs. These auto-filled properties become accessible only after the user has provided the basic AAS information.

Finally, in section three, users can input the required AAS basic information through a structured form. Once this information is provided, the user can proceed with the instantiation of the selected SMT, as described in Figure 8 and Figure 10.

[14] Please refer to Table 2 to retrieve a list of SMTs that are supported during the first iteration of T3.1.

2.4.1 Instantiation of the Digital Nameplate SMT

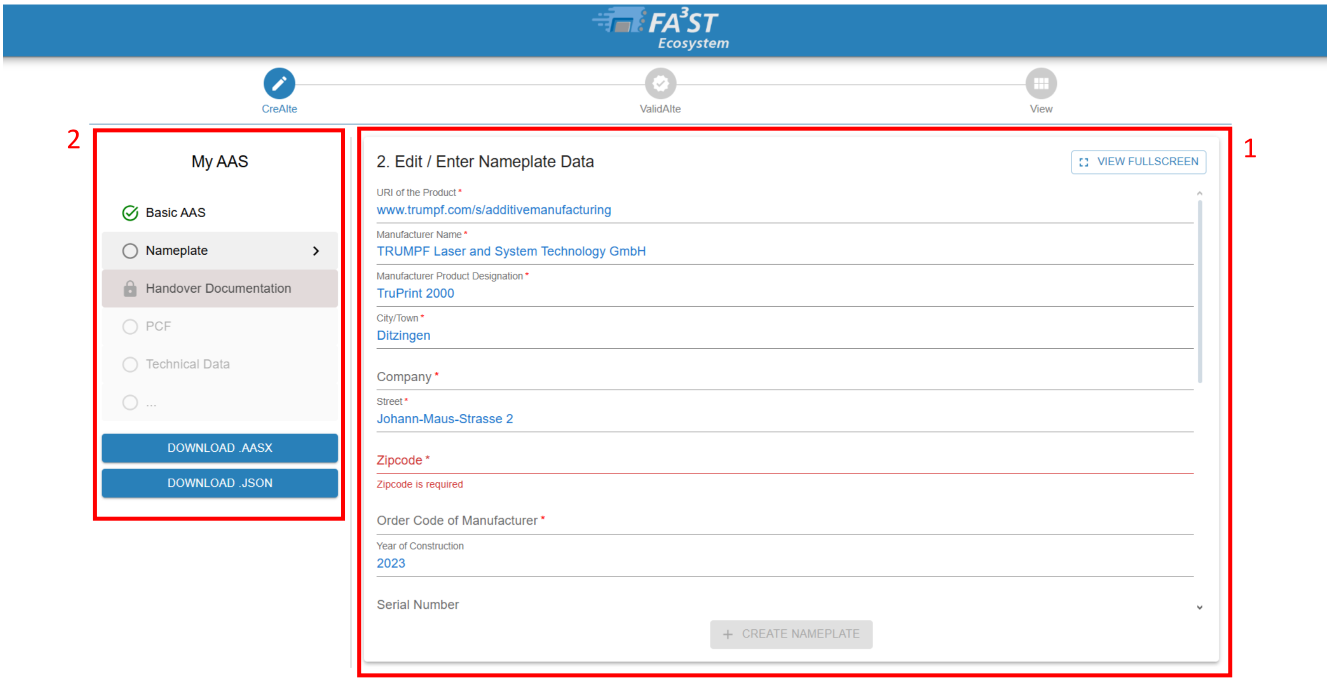

Figure 8: Digital Nameplate SMT Screen of the FA^{3}ST CreAItor

The Digital Nameplate SMT instantiation screen consists of two main sections. On the right-hand side, a structured form is presented, allowing users to provide input for each SM element. As described above this form is pre-populated with information extracted from the input data. Users can review and refine the extracted values, adding missing information, or removing entries that were incorrectly extracted. Certain fields within the form are marked with a red asterisk, indicating recommended elements as defined in the respective SMt specification. Once the user has completed and reviewed the form, the SMT instance can be generated by clicking the “Create Nameplate” button, which adds the instantiated SM to the current AAS model.

The second section, displayed on the left-hand side, provides an overview of all SMs already added to the AAS. After completing the instantiation of the SMT, users can export the AAS model in either AASX or JSON format. The exported model is fully compliant with the AAS metamodel and can be used with any AAS-compatible tools or platforms, such as the FA^{3}ST Service[15] or the AASX Package Explorer[16]. Figure 9 illustrates the generated model in more detail, including its structure and the instantiated SM elements.

[15] The Fa^{3}st Service can be retrieved from this GitHub repository.

[16] The AASX Package Explorer can be retrieved from this GitHub repository.

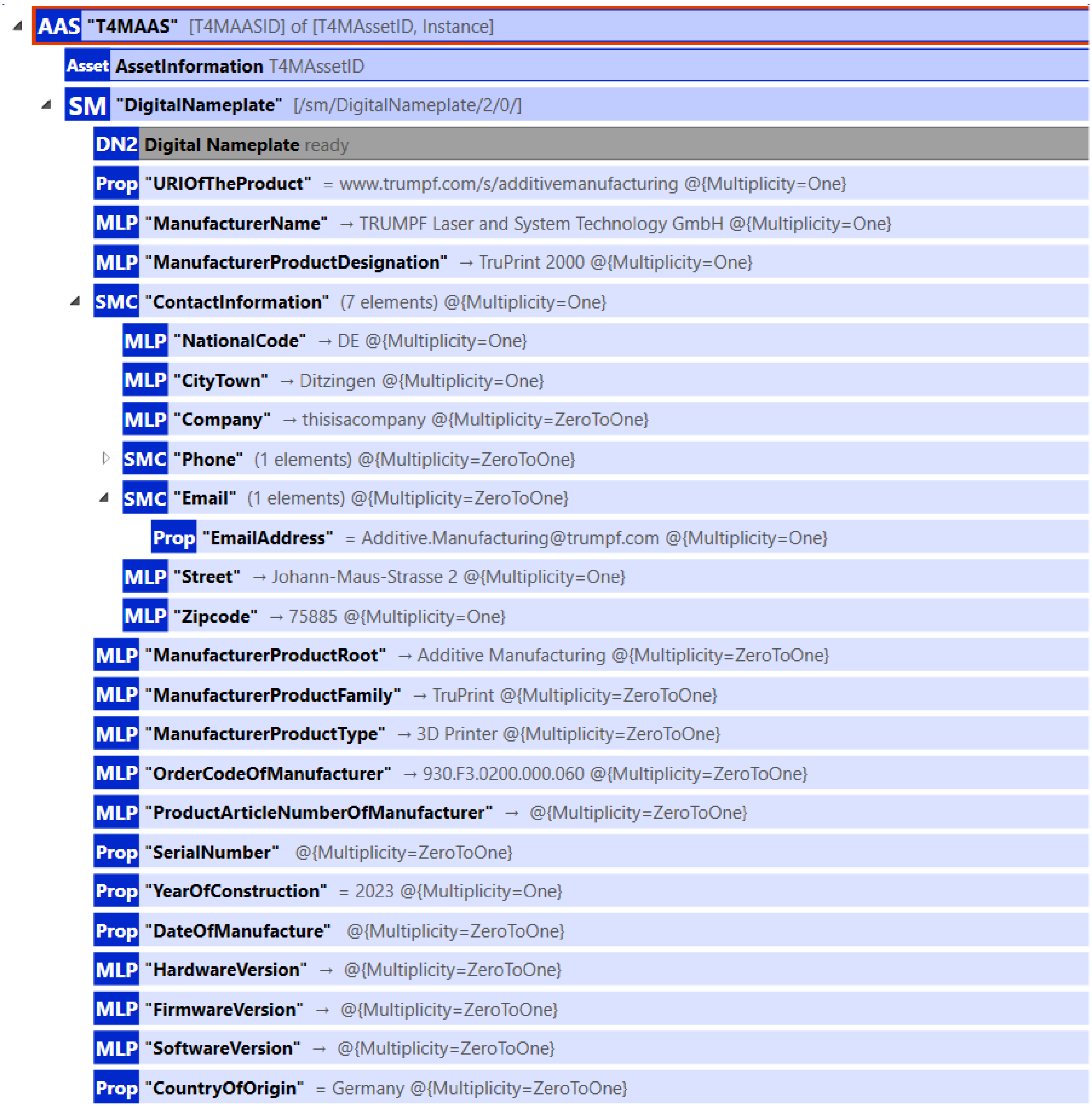

Figure 9: Example of an AAS Model Generated with the FA³ST CreAItor

2.4.2 Instantiation of the Capability Description SMT

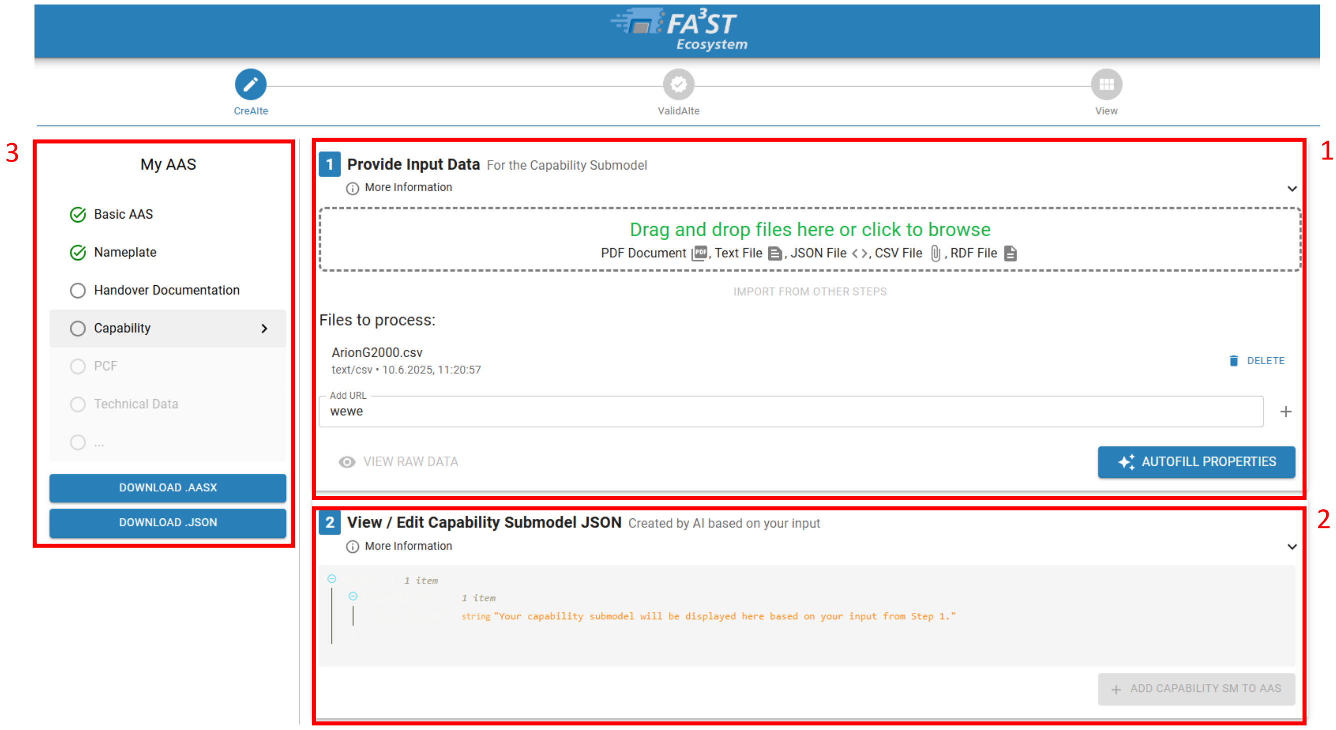

Figure 10: Capability Description SMT Screen of the FA³ST CreAItor

As for the provision of basic AAS information, the interface for the Capability Description SMT is organized into three main areas. On the right-hand side (1.), users can provide input data and start the automated instantiation of the SMT. Users have two options to provide input data. First, they can upload input files that include the actual information for the instantiation, such as PDF or CSV files describing the actual assets. Additionally, they can upload domain-specific ontologies in RDF format, which are used as context to adapt the SMT instantiation to the user's domain. If no additional ontology is provided, the system uses the default ontology described in Chapter 2.3. Once the user has provided the relevant input, the instantiation process is triggered by clicking the "Create Capability SM Button". The instantiated SM is then displayed to the user in the bottom right-hand side (2.). Users can directly annotate and review and annotate the generated AAS JSON representation of the instantiated Capability Description SM. Once satisfied the user can add the instantiated SM to their AAS by clicking on the button on the bottom right corner.

On the left-hand side (3.), the user can download the generated AAS either in AASX or JSON format model once the instantiation of the Capability Description SMT is finished by using the respective buttons at the bottom of the sidebar.

2.5 Initial Evaluation Results

This chapter presents the initial evaluation results of the FA³ST CreAItor. As the main objective in T3.1 was to support non-expert AAS users in generating AAS Models, the evaluation mainly focused on assessing the user experience of the tool. To this end, the User Experience Questionnaire (UEQ)[17] was employed as a structured and well-established instrument for capturing subjective impressions of usability. The official UEQ standardized evaluation and benchmarking tool[18] was used to score and analyze the collected data. The corresponding results are visualized in Figure 11.

Data collection was carried out with an expert group consisting of 10 domain experts, including both domain experts without AAS-specific knowledge and AAS experts. It is important to note that the feedback was gathered in a moderated setting with the evaluation team present, which may have introduced a certain degree of bias to the collected data. Nevertheless, the results provide valuable initial insights into the usability and perceived effectiveness of the FA³ST CreAItor. Figure 11 presents the results of the evaluation of the FA³ST CreAItor. The graph shows scores across the six UEQ categories: Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation, and Novelty. Each bar is segmented into five performance levels (from Bad to Excellent), and the black line represents the mean score for each dimension based on the collected feedback.

The tool received consistently positive ratings, with most categories scoring between above Average and Good, especially in Attractiveness, Efficiency, and Stimulation. Notably, Dependability received the lowest mean score, suggesting a potential area for improvement.

In addition to the quantitative data, qualitative feedback was collected to deepen the understanding of user experience. Participants frequently highlighted “fast,” “efficient,” “easy to learn,” and “valuable” as standout adjectives. The most appreciated features included automatic data extraction and pre-filling, and support for multiple SMTs. These features were credited with significantly saving time and simplifying AAS creation, especially for non-expert users.

Users expressed strong interest in continued use and recommendation of the tool, with most responses indicating they were “very likely” or “somewhat likely” to recommend it. Future enhancements requested include support for PDF uploads, API integration, bulk AAS creation, and clear indicators for missing fields. Several users proposed to expand the tool by additional SMTs (e.g., Product Carbon Footprint or Maintenance).

[17] For a full documentation about the UEQ please refer to this website.
[18] The tool can be accessed on this webpage.

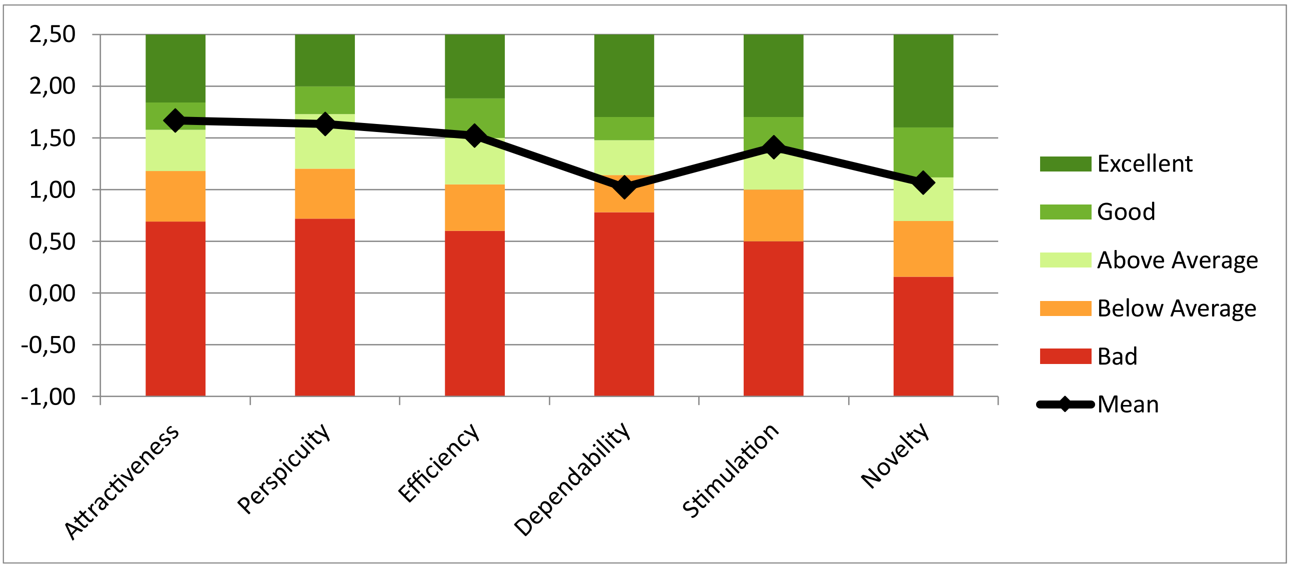

Figure 11: Evaluation Results FA³ST CreAItor

2.6 Challenges and Lessons Learned

The activities conducted in T3.1 revealed several challenges and key lessons regarding the FA³ST CreAItor that will inform the next iteration of work. One of the primary limitations encountered in the current development phase is the pending final publication of the SMT specifications. Although no significant changes to the data model are expected, the lack of a stable standard is considered a risk. Therefore, the tool will need to be revisited once the SMT specifications are finalized. Another open challenge relates to the instantiation of relationships among the Capability Descriptions SMT elements, such as *GeneralizedBy*, *ComposedOf*, or various *Constraint* relations (e.g., property or capability constraints). While the SMT framework allows such relationships, the current version of the FA³ST CreAItor does not yet support the instantiation of these relationships. Enabling this instantiation is a priority for the second iteration of Task T3.1. The flexibility in modeling *PropertySets* and *Relationships* elements also presents a challenge for the user interface. As the Capability Description SMT needs to be adapted dynamically to each instantiated Capability, the user interface must also be generated dynamically to reflect the specific structure and content of each Capability instance. This requires a more flexible and modular design of the SMT Editor component. Currently, the FA³ST CreAItor only enables users to directly review and update generated AAS JSON representation of the instantiated Capability Description SMT. This limitation needs to be addressed during the second iteration of T3.1. Furthermore, in the second iteration the FA³ST CreAItor should be extended to support the SMTs specified in Table 2.

Finaly, the main research contributions resulting from the activities conducted during the first iteration of Task T3.1 are highlighted. First, an initial approach for semi-automate the instantiation of the Capability Description SMT was introduced. The key challenge for this contribution was providing the domain knowledge necessary to identify and extract the information required to instantiate the SMT. Our main contribution lies in the proposed combination of the Context Provider and the AI Service, enabling domain-specific and relation-aware information extraction. Second, a user-centered approach to semi-automate SMT instantiation using AI is proposed. This approach addresses the challenge of hallucinations and incorrect information extracted by LLM. To mitigate this issue, a semi-automated SMT instantiation process with human oversight is introduced, ensuring both correctness and completeness of the instantiated SMTs.

3 Collaborative DTs Supporting Value Networks

3.1 Introduction

In a MaaS environment, consumers who need to outsource a manufacturing process and providers who have the capacity to allocate resources to serve such consumers by accommodating their manufacturing process need to connect in a way that allows them to identify potential partners and evaluate whether their needs and offerings align. To achieve this, they must be able to discover one another, assess compatibility, and engage in a structured interaction that moves from initial interest to concrete collaboration. This process involves not just finding a match but also understanding whether the services offered meet specific requirements, discussing, negotiating and agreeing in clearly stated terms, and monitoring the collaboration process along the duration of the partnership.

The first step of the process is matching potential partners, a process that takes place in the main T4M platform, as described in D4.1. Once a match is identified, the parties enter a negotiation phase involving the exchange of service requests and offers through an iterative feedback loop. As part of MIRA, the Supply Chain Monitoring component of the T4M architecture, this process involves configuring modelling aspects, defining access policies, and aligning other operational details necessary to reach a final agreement. Terms may need to be revised as the negotiation progresses. These interactions require continuous updates and a shared understanding of the negotiation's state and context, which are managed by each organization's DT.

For this entire interaction to work smoothly, there is a critical need for a clear and trusted flow of information between the independent DTs of each party. This is achieved through the collaborative DTs infrastructure which ensures secure and governed communication. Through this infrastructure, messages, status updates, and negotiation events can be exchanged, enabling both parties to stay aligned from the initial discovery phase through negotiation and up to the final execution of the agreement.

3.2 Background and T4M Innovation Proposal

In the beginning of the project, the foundational concept of cognitive DTs across a value network has already been explored through Maggioli's DT platform MIRA, which enables the creation and modelling of digital representations of resources and processes into systemic assets, residing within organisation and shared across them. These digital entities are enriched with information about their capabilities and capacities, allowing them to describe what they can offer and under what conditions.

MIRA handles DTs on supply chain level by supporting the interaction of collaborators across value networks and enabling DTs to communicate in real time. This allows organizations to share information, such as resource availability or delivery options and statuses and helps them stay aligned throughout the collaboration. The platform also provides dedicated sharing tools that enable secure and traceable data exchange. These tools support the delivery of relevant information both throughout and upstream in the flow of business processes, allowing for end-to-end transparency. This traceability is essential for maintaining trust between stakeholders and ensuring compliance with governance requirements. An overview of the MIRA architecture can be found in Figure 12.

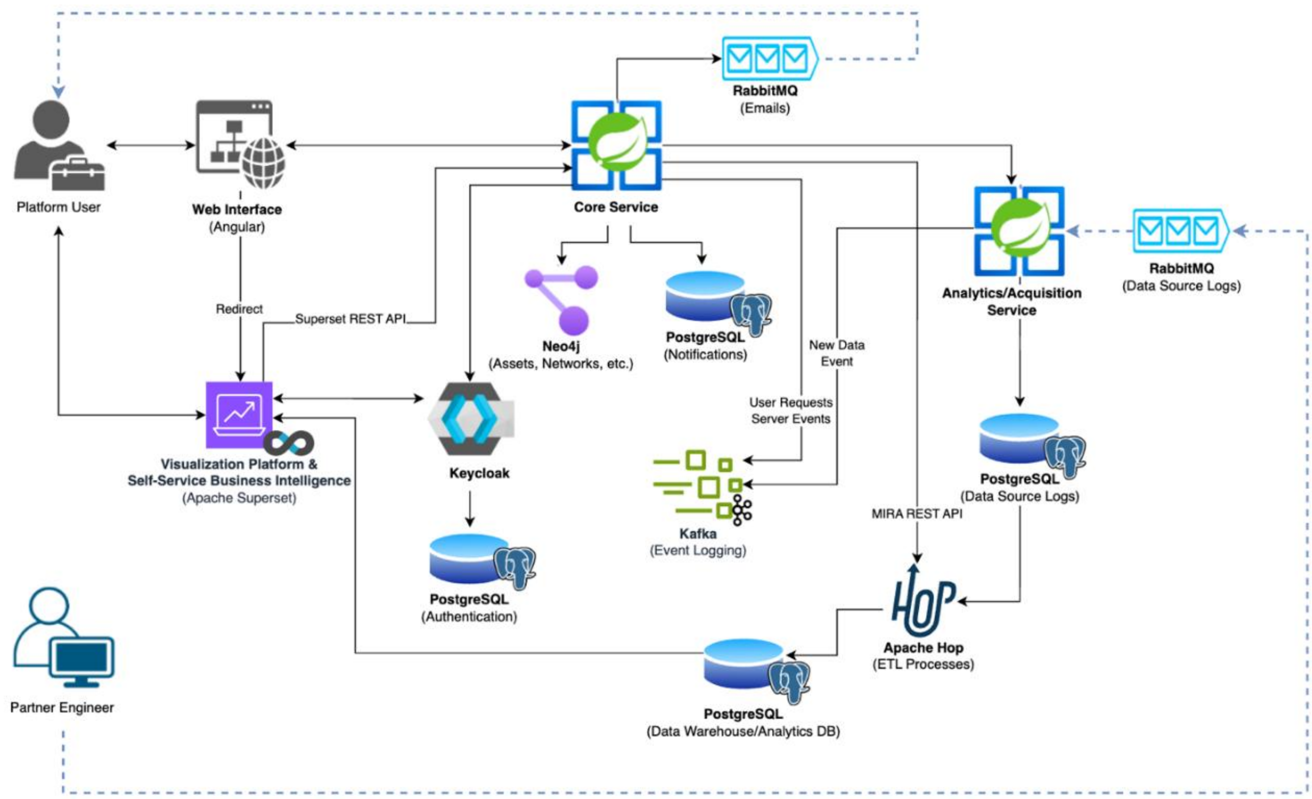

Figure 12: Overview of MIRA Architecture

The main components comprising MIRA are the following:

- **Web interface:** The web application built on Angular, that provides users with the interface to interact with MIRA.
- **Core Service:** The central component of the platform, playing the role of a gateway. All requests pass through this service, get forwarded to the designated services and return a response to the UI. The main functions performed by the core service are user rights filtering depending on their respective roles, management of business logic with a Neo4j database, notification handling, and request routing.
- **Keycloak:** The identity management service, responsible for user authentication, credentials management, and issuing authentication and authorisation tokens for access to several functionalities, depending on each user's role and relationship with registered organisations.
- **Neo4j:** A non-SQL graph database that stores and manages business logic data on relationships between systemic entities like assets, and telemetries that represent properties of AAS compliant DTs.
- **Analytics Acquisition Service:** The component responsible for data consumption through various inputs. The standard input source is a RabbitMQ instance which handles messages posted onto MIRA's API endpoints. Additional input methods can be easily integrated to expand compatibility with external systems, such as the Data Space Connector, which will be integrated in the second iteration of the project.
- **Apache HOP:** An ETL service, which combines input data from the data source logs database, with business relationships from Neo4j, to transform input data into a clear relational format grouped under their assets/telemetries in preparation for system analytics and BI services and loads the output into another PostgreSQL instance for data warehousing.

- **Apache Superset**: The service used for data visualisations and business intelligence operations like predictions or data analytics. It receives pre-processed input from the warehousing PostgreSQL instance, performs its operations and then provides visual output on the web interface. Superset output datasets are extracted through its API and will be used in feature such as the Balanced Scorecard.
- **Kafka:** An event logging service which handles two types of data: a) user events originating from the web interface, used to trigger notifications and b) system events originating from backend services, used to trigger internal processes. In the scope of Tec4MaaSEs, the software used to publish and consume data from Kafka will be integrated with the message bus to support interactions with the marketplace.
- **RabbitMQ:** A popular, lightweight open-source message broker, deployed in the cloud. It supports multiple messaging protocols, ideal for consuming data from diverse external sources. Currently, there are two queues in MIRA. One for data source logs consumption, and one for email notifications forwarding.
- **PostgreSQL:** One of the most robust relational database management systems. There are four PostgreSQL instances in MIRA. One used by Keycloak to store authentication related data, one used by the core service to store and manage all system notifications, one used by the analytics acquisition service to store data source logs, and one used for data warehousing after the involvement of HOP.

Building upon this background work, T4M introduces an innovative approach by seamlessly streamlining the prerequisite step of exploring the market to identify potential partners and establishing collaborations with the negotiation phase between partners, when specific details are discussed and clear agreements are made. By utilizing data spaces, technical schemas and requirements generated on factory level can be shared and compared, enhancing both the partner matching process and the negotiations between established collaborators.

Through the marketplace platform, T4M allows organizations to discover suitable collaborators and make service requests; while using MIRA platform, partners can exchange offers, negotiate services and resources and monitor the status of negotiations, all within a trusted environment that supports interactions across the value network.

3.3 Proposed Approach

The proposed approach builds directly on the DT logic inherited from the MIRA platform, extending it to support more advanced interactions across the value network. MIRA's foundation for modelling resources and enabling collaboration has been preserved, while new components have been introduced to align with emerging standards and to better support negotiation scenarios in MaaS environments.

A key enhancement is the integration of the AAS standard into the way DTs are created. This approach involves the creation of assets by manually importing models based on the AAS structure, making the DTs AAS-compliant. This allows each asset to be built using standardized SMs that define its capabilities and technical characteristics. By aligning with AAS, a uniform way to represent what resources can offer is supported, making it easier for collaborating parties to evaluate shared information.

In the second iteration, data space connectors will be used to consume descriptions of AAS-compliant DTs built on factory level and automatically generate systemic assets in MIRA. Data spaces will be utilised further to receive real time data on dynamic data like production capacities, which will be consumed into MIRA telemetries to monitor and used to determine satisfiability of manufacturing needs for a given order in the context of a negotiation between collaborating parties.

To facilitate such negotiations between MaaS providers and consumers, a messaging system within the collaboration environment has been developed. This system allows users to exchange messages, documents, and predefined templates that include fields linked to the AAS SMs. These templates provide structure and consistency during negotiation loops, making it easier to share offers, respond to requests, and manage communication. Complementing this, customizable negotiation status mechanism will be introduced, where users can define their own set of statuses to reflect the different stages of a process. These user-defined statuses will provide a unified and real-time overview of ongoing interactions, allowing all parties to remain aligned across various requests, negotiations and service deliveries. Together, the messaging system and customizable status tracking make the negotiation process and MaaS service delivery more efficient and coordinated, by helping all parties stay aligned, respond efficiently, and maintain a clear view of progress at every stage.

A set of messages will be issued across the different platforms of T4M to notify other services on relevant updates. The deployment of a message bus implemented with Kafka, will provide the infrastructure for interested services to post and consume the necessary messages to ensure alignment. This way MIRA can establish timely collaborations based on the results of matching implemented in the marketplace and then post updates on negotiations and orders to provide marketplace users with an overview of current status.

3.4 Implementation of MIRA

As part of the ongoing work within the T4M project, significant development has been carried out in the MIRA platform to support the representation of resource capabilities, in line with the proposed AAS-compliant DT structure.

One of the key elements implemented is the Details tab within each Asset. This feature is designed to support the structured representation of **Capabilities**, which are critical when describing what a resource or a service can offer in a MaaS context. The **Details** tab is designed to display data imported from AAS SMs, providing a clear view of a resource's technical features and service information within the DT.

Initially, a flexible functionality has been implemented that allows users to create custom templates that consist of sections and fields. Users can access the **Templates** section, create a new template, as depicted in Figure 13 and make it available for use across all assets, as presented in Figure 14.

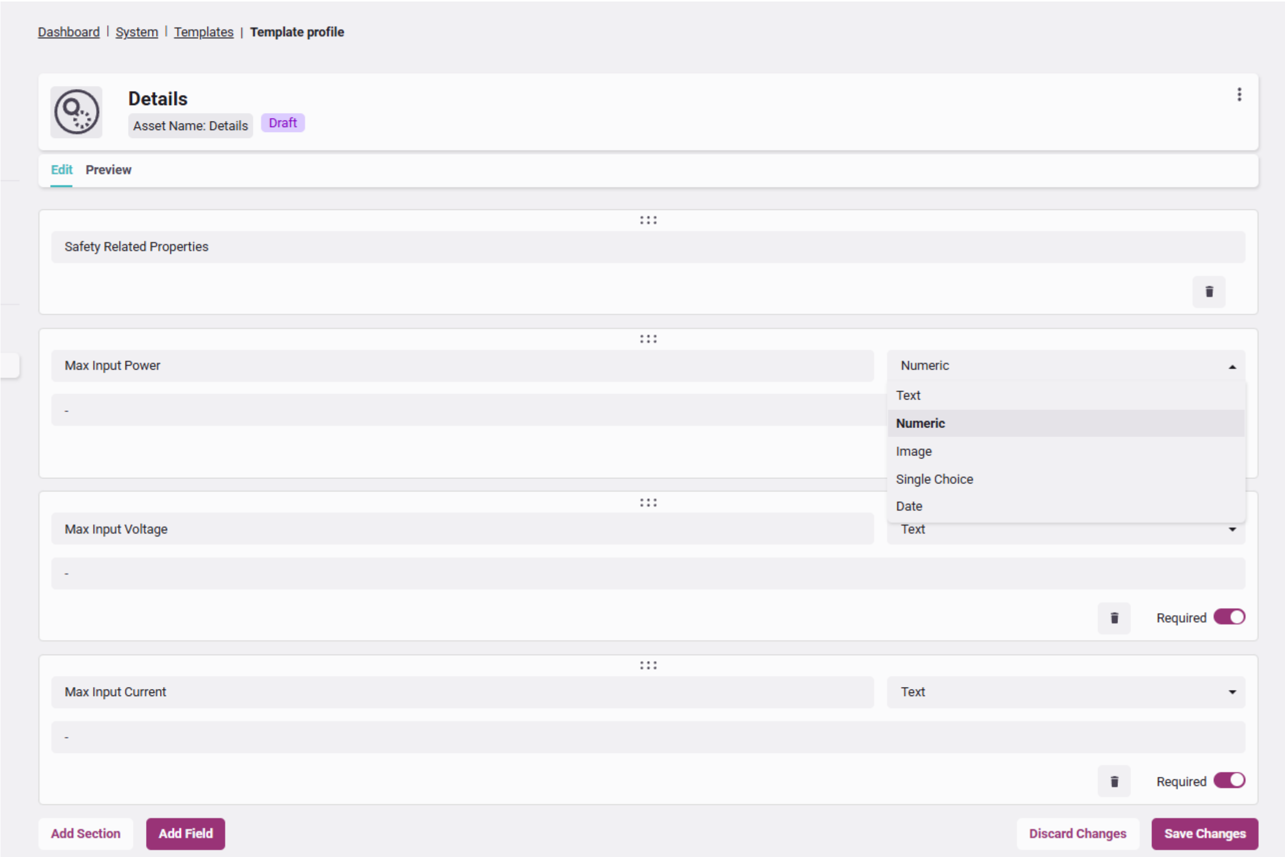

Figure 13: Details Template Creation

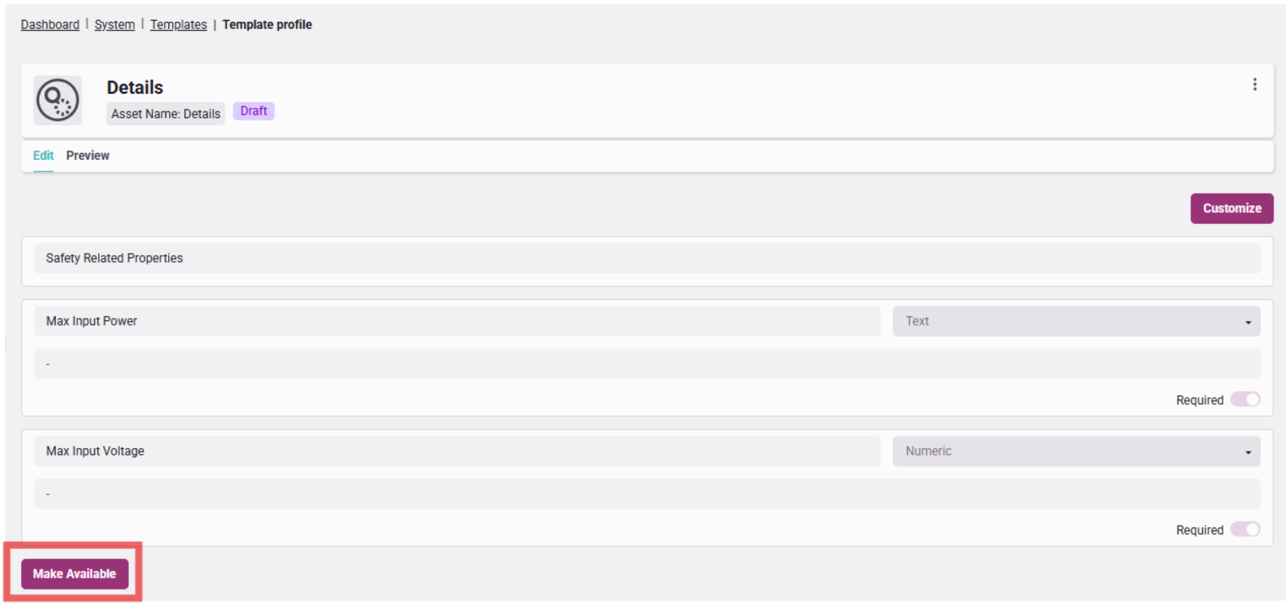

Figure 14: Details Template Activation

Once a template is created, the user can go to an asset's profile page, import the selected template, and fill in the required information directly within the interface (see Figure 15 & Figure 16). This approach supports early-stage manual definition while preparing the system for full AAS integration.

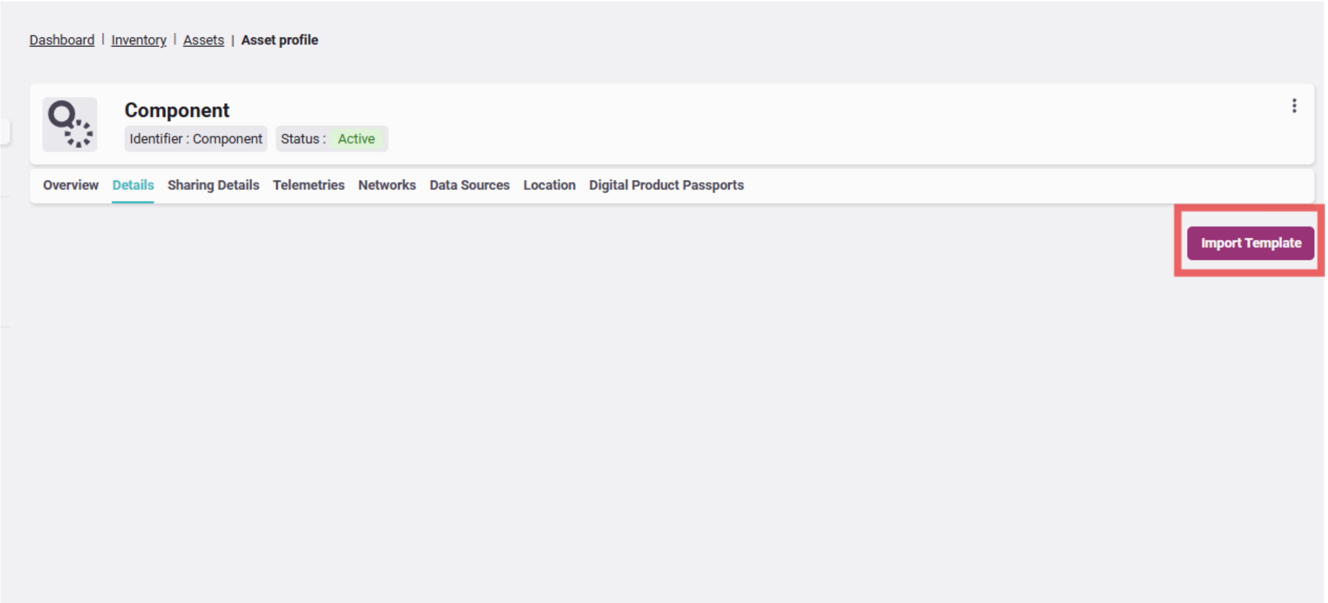

Figure 15: Importing Details Template in Asset Profile

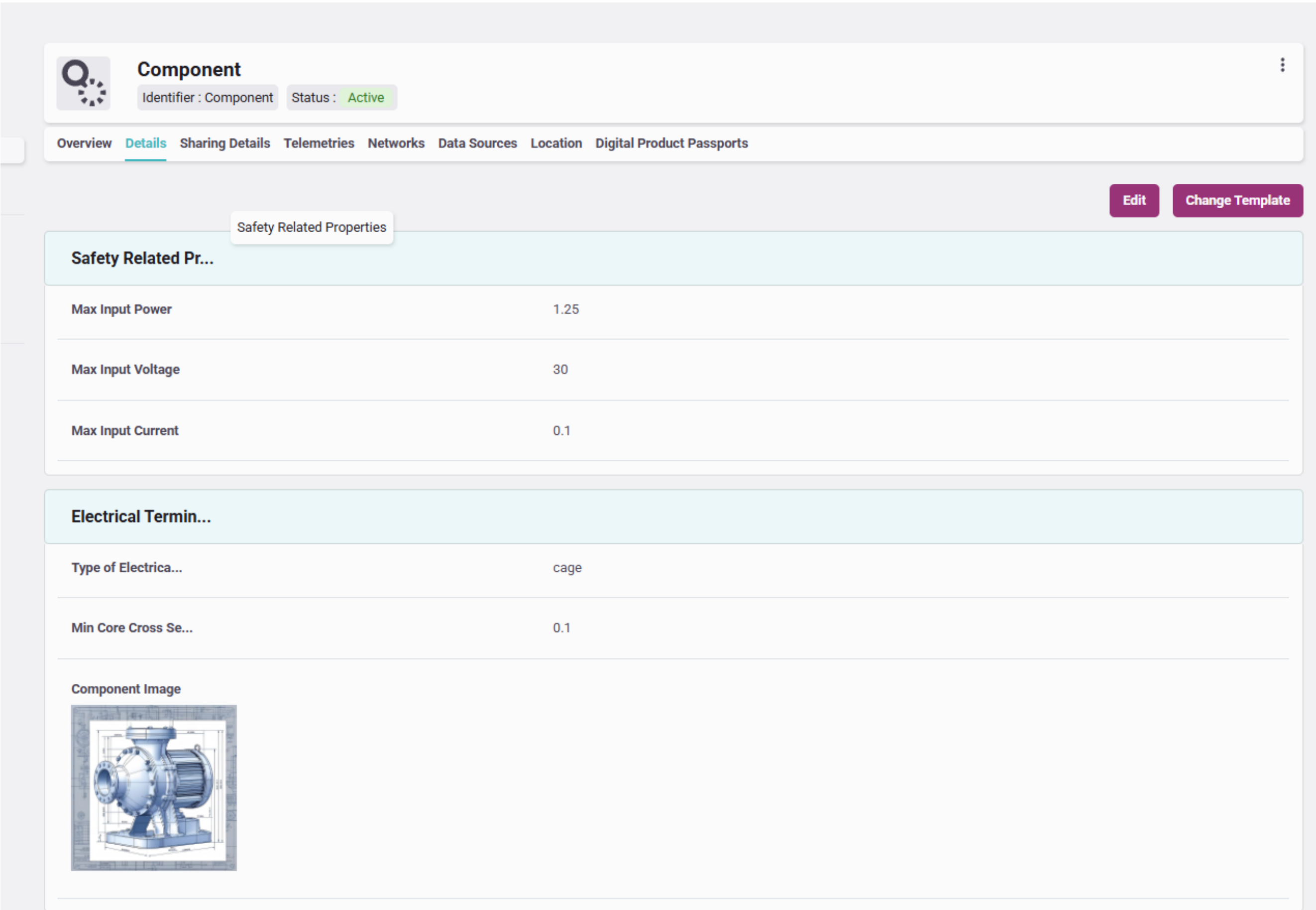

Figure 16: Visualized Template Information

In future versions, the details tab will be renamed based on each organization's preferences using a field customization and localization feature. This will let organizations use their own terms and layout. More importantly, the tab will support automatic import of capabilities from AAS models, making it easier to set up DTs. Each Capability SM element will generate a new entry in the details tab, while each property SM element will additionally trigger the creation of a telemetry under the DT, which will be a variable that can dynamically reflect the current value associated with this property. Such telemetries can be updated in real time through the data space connector, or a post message in a secure API endpoint associated with the asset.

Another key component implemented in MIRA is the **Messaging system**, designed to support negotiation loops between MaaS providers and consumers. This system has been developed to facilitate the exchange of service requests, offers, and supporting information during the negotiation phase, but perhaps also after the agreement phase.

The initial implementation is accessible through the collaborator's profile page, where users can initiate communication by creating and sending blank messages to their selected collaborators as shown below in Figure 17. In addition to plain messages, users can also send predefined templates that include fields derived from AAS SMs related to ordering processes, providing a structured format for exchanging relevant data during negotiation.

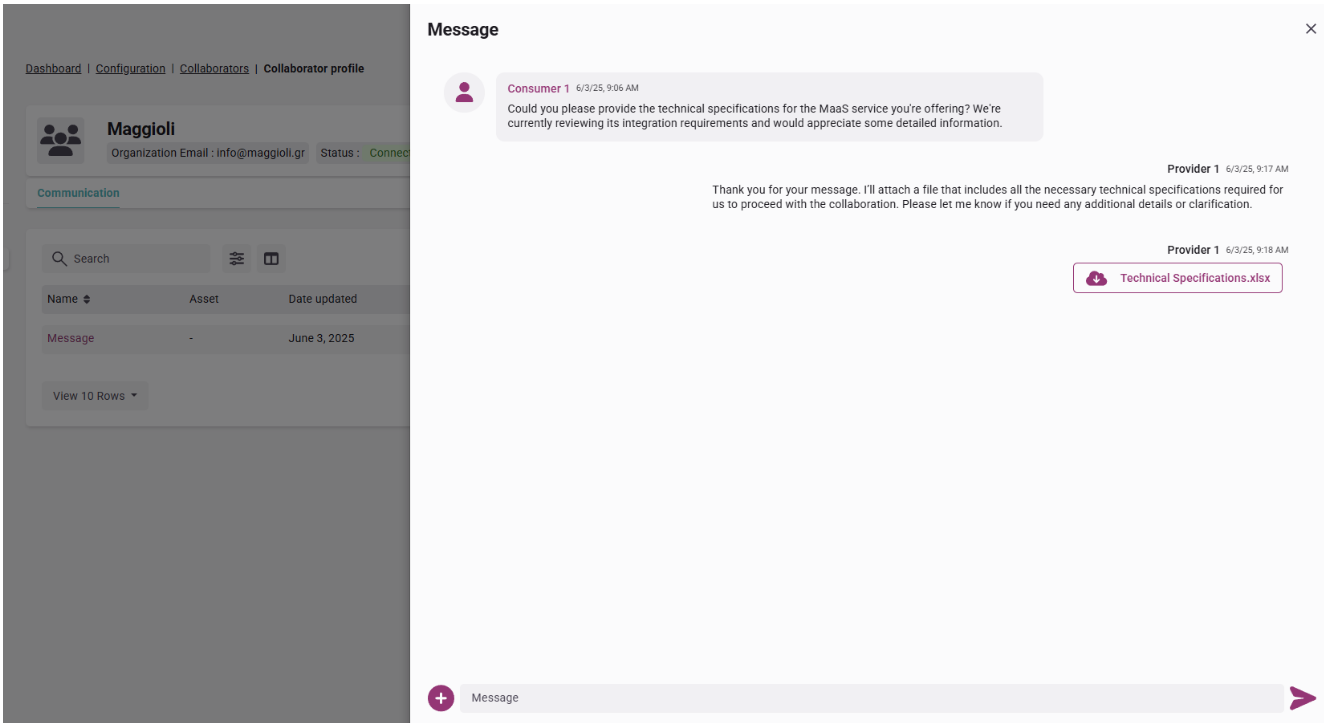

Figure 17: Blank Messaging Exchange

In the first version, a template based on the Purchase Request Notification SM has been integrated. This template enables users to complete predefined fields, such as order quantity, delivery date or currency and send the filled form directly to a collaborator as a message. Such examples are presented in Figure 18 and Figure 19. The messaging system also supports the attachment of files or documents to any message and allows users to link related messages within the same communication thread with a specific collaborator, as depicted on Figure 20.

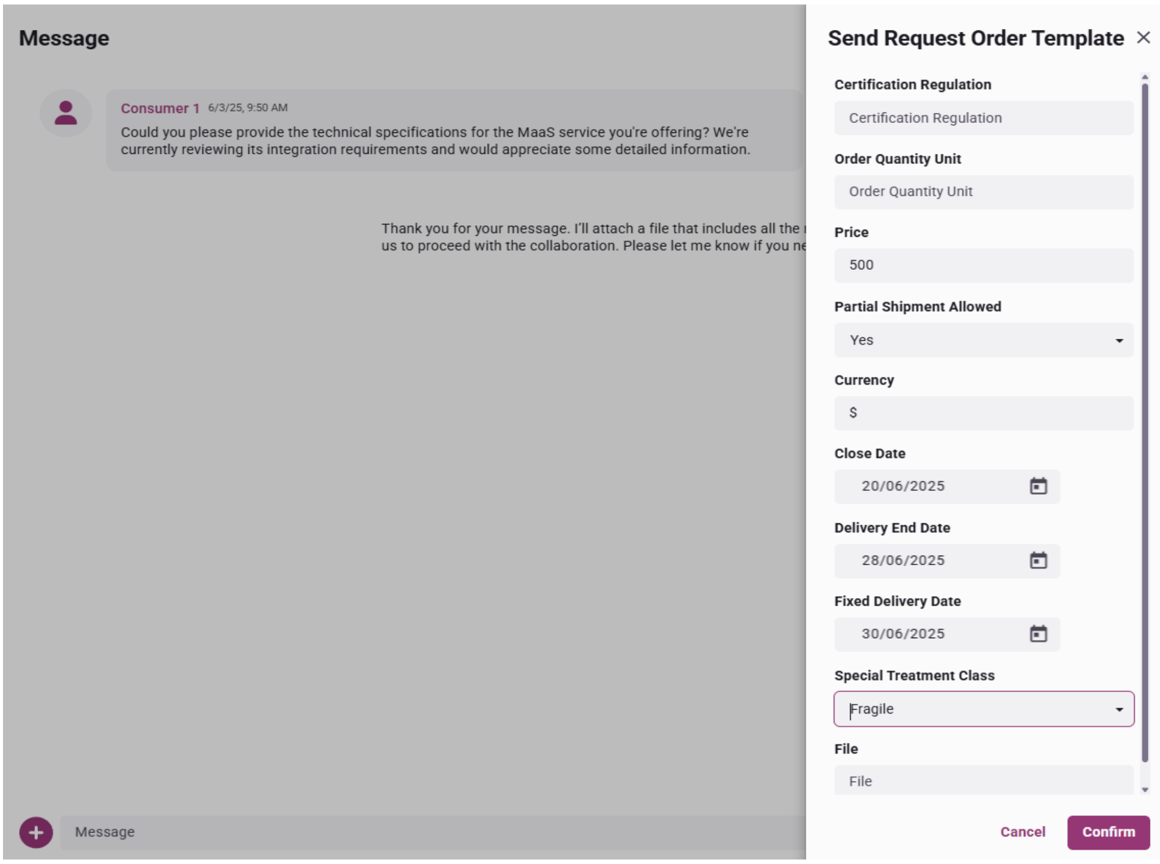

Figure 18: Template Fields Completion

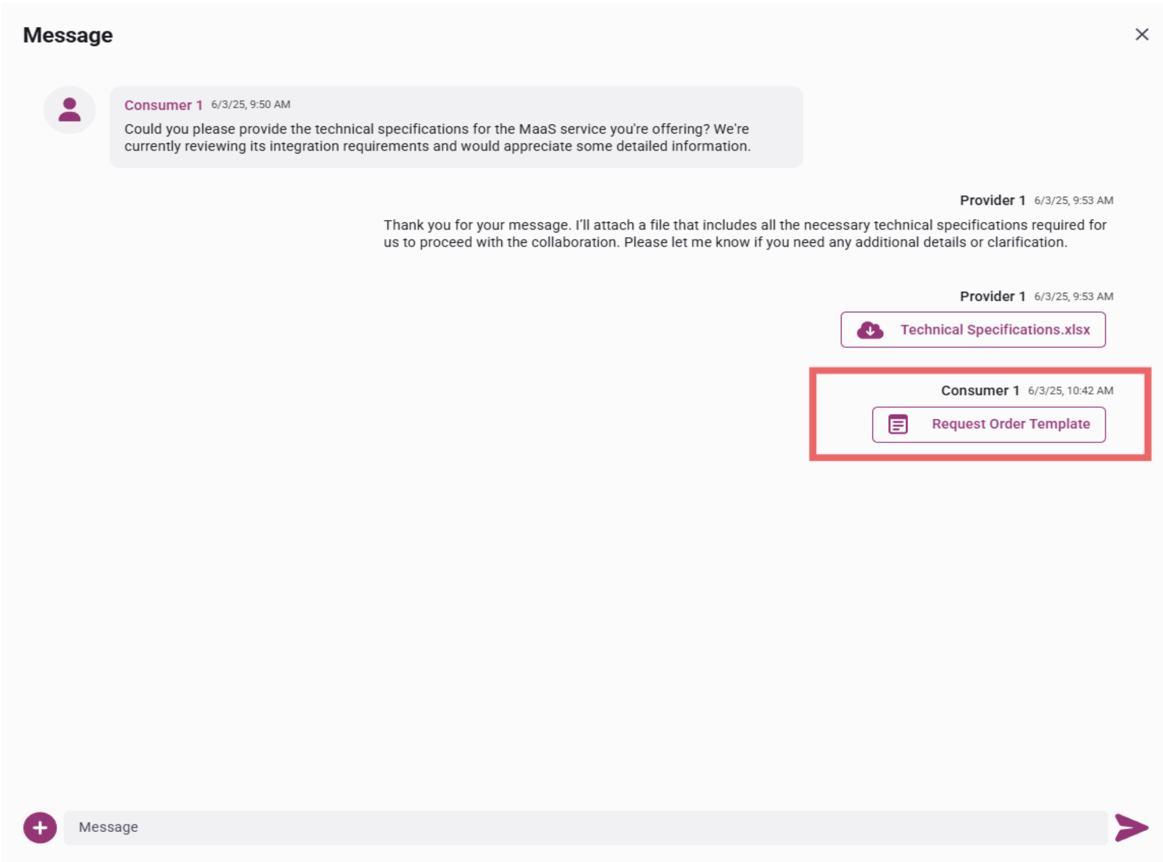

Figure 19: Template Message Exchange

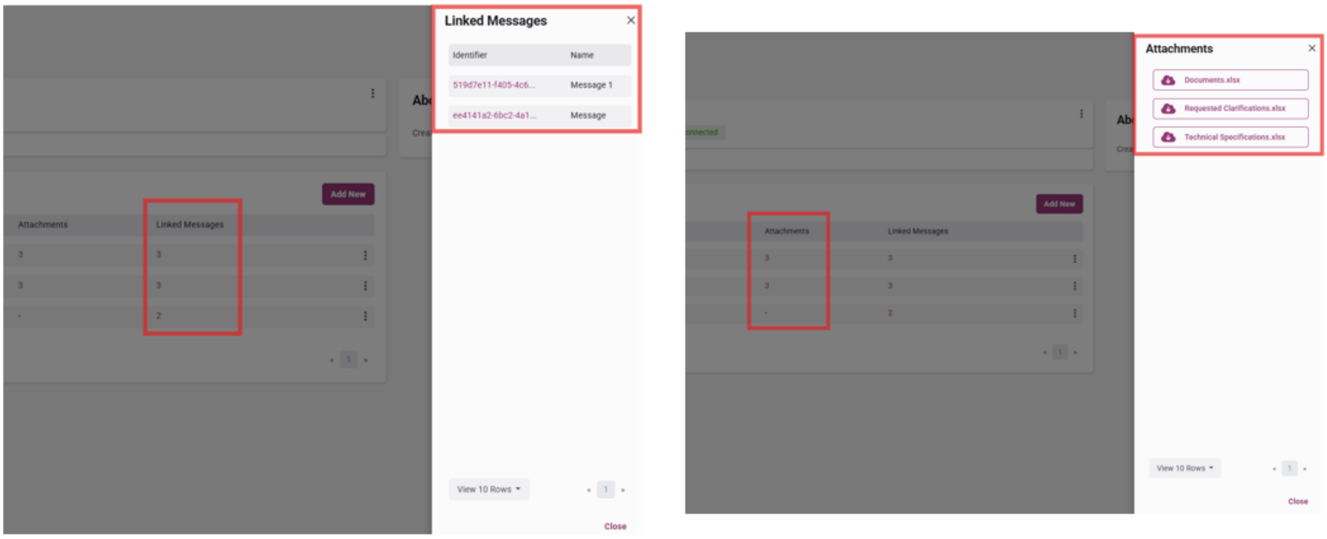

Figure 20: Attachments & Linked Messages in Messaging System

In future versions, this functionality is planned to be extended to the Shared Asset's profile page, providing users with an additional way to create and send messages directly from a specific asset. Additionally, users will be able to choose from a broader selection of templates based on different AAS SMs and will also have the ability to create and send custom templates by selecting specific fields according to their needs. This will enhance flexibility and better support a variety of negotiation workflows across the value network.

The second version of this deliverable will delve further into these features and thoroughly describe the integration of MIRA with the message bus of the project and a data space connector to receive AAS models and real time data from factory level DTs.

3.5 Initial Evaluation Results

By the end of M18, design and development on MIRA focused mainly on three pillars:

- The introduction of a details tab in Assets to provide an overview of manufacturing capabilities.
- The development of a conversation environment to facilitate partner negotiations and transactions.
- The mapping between all main AAS SM elements and MIRA systemic entities.

Each of these pillars can be evaluated independently.

3.5.1 Introduction of Asset Details

The introduction of details tab is an effort to properly visualize information encapsulated in AAS compliant assets, in a user friendly format. In this first iteration the focus was given in customization and manual generation of contents in this tab. The core supported concepts are capabilities, and capacities reflected through properties. The idea was to initially provide a user interface that can accommodate the data modelling needs of the project by allowing users to provide their input manually. Then, after verifying that the user requirements are met, the automatic consumption of such data from ASS models formatted in JSON or AASX file types will be supported. Currently, a minimal adapter which extracts capability and property elements from AAS models in JSON format, and generates an Asset based on the consumed data is being tested. The current interface successfully displays this basic information, but further testing with concrete examples built upon the pilot value networks will ensure the practical compliance between MIRA features and the AAS models implemented in the scope of the project.

Furthermore, the interface will be enriched to properly reflect hierarchical information, such as the grouping of AAS elements under SM element collections, or the relationships between different assets or components. Similarly, the adapter to consume more complex AAS models, or models formatted in AASX will be developed.

3.5.2 Collaborator Conversations

The concept of collaborator conversation is simulating the real-life interactions between collaborating party representatives during the process of a negotiation, or after an agreement has been reached, during the phase of generating an order request and then monitoring the delivery process. The basic idea is to support all potential information needs that users may have. The simplest way to do this was to implement a messaging system that allows the exchange of free text, with the additional feature of attaching a document in a conversation. This way, any exchangeable piece of information over email communication can be processed and exchanged on MIRA, thus substituting email threads, providing the additional benefit of communicating on the same platform which monitors the value network, and allowing an immediate connection of a conversation with the asset it refers to.

Given the fact that most processes regarding negotiations and orders typically have standardized sets of fields that need to be exchanged, the next step is to provide users with structured messages that illustrate such fields, in order to avoid miscommunication or misalignment of ontologies in free text. To do this, MIRA supports two options:

- The generation of customizable message templates with user defined fields.
- The generation of message forms based on relevant AAS SMs.

The initial version of forms based on AAS SMs only supports “Purchase Request Notification”. More SMs such as “Purchase Request Response”, “Purchase Order Creation” will be supported among others beyond the first iteration, to satisfy the needs of all pilot value networks. The additional feature of allowing users to manually import AAS SMT that are not natively registered into the system, is also being considered.

3.5.3 Mapping AAS Submodel Elements into MIRA

The AAS defines a standardized framework for representing DTs of assets. It ensures interoperability across systems and industries, enables structured data exchange, and provides a unified interface to assets. Inside an AAS, there may be one or more SMs, which represent focused sets of structured data describing specific aspects of the asset. Each one contains SM elements, which are the atomic units of information used to describe asset characteristics, behaviors, and relationships in a structured, interoperable, and machine-readable way.

The key types of SM elements identified and their status regarding their representation in MIRA are the following:

- **Capabilities**: Descriptive fields on characteristics of an asset. They mainly reflect what assets can do. Currently, MIRA extracts capability information and lists it in the details tab of an Asset. This way users can have an overview of each asset.
- **Properties**: Mainly quantitative fields that describe metrics and manufacturing capacities of an asset. MIRA provides an overview of them in the details tab, and implements a mapping between them and telemetries, which act as variables to store and monitor dynamic values.
- **Ranges**: Quantitative fields that provide valid ranges for numeric properties. They will be consumed in future versions to define validity checks for telemetry values.
- **References**: Associative fields between interconnected assets, or components. They will be used in future versions to define relationships between assets. The design of how they will be presented in MIRA is under discussion.
- **Operations**: Fields that describe actions that can be done by an asset and provide a set of instructions to run them through the DT. So far, these elements seem to go beyond the scope of the project. Thus, their incorporation in MIRA is an optional feature.
- **Files**: AASX formatted assets can include files. Their consumption and representation in MIRA are currently being designed.
- **BLOBs**: Similarly with files, AASX formatted assets can include BLOBs. Their consumption and representation in MIRA are currently being designed.
- **Collections**: Grouping classes for multiple SM elements under a common semantic context. They are not elements, but a grouping mechanism to gather related information in the same space. MIRA will utilise them on its user interface, to visually group related information.

3.6 Challenges and Lessons Learned

During the development of collaborative DTs for value networks for the first iteration, several challenges were encountered. One of them was the design of a negotiation approach that could flexibly support the diverse requirements of all pilot scenarios. Each pilot involved different processes, and interaction models, making it difficult to define a generic solution that remained adaptable across use cases.

The introduction of AAS models into MIRA also proves to be a challenge both in terms of integration and design. The various components and elements inside AAS SMs must be properly mapped into systemic entities, while simultaneously get classified under visual components to achieve a user friendly and comprehensive interface. The understanding of all elements, their incorporation into MIRA database schemas and the creation of a roadmap and a strategy to continuously expand the compatibility of the platform with the AAS SMTs needed for the project, especially the ones that are being in concurrent development in the scope of T4M is a difficult task, which is occasionally updated and refined along the development process and the identification of additional, value network specific peculiarities.

A further issue is the variation in status indicators used by different organizations. For example, one organization might use “order closed”, while another uses “order delivered” for the same step. Even though a listing of differentiated status variations is not available in the current iteration, the foresight of this issue resulted in designing customizable, generic status definitions within the negotiation processes. Thus, each organization will be able to use terms and flows that matched their internal practices while still ensuring consistency across the different value networks.

As a result, multiple lessons revolving around the same concept of reusability have been learned. The incorporation of AAS models that have a wide variety of formats or involved elements, the design of a negotiation and ordering environment through conversations that need to accommodate multiple contexts and ontologies, and the setup for custom tab names or statuses in different transactions, are all features which need a very generic approach, in order to accommodate differentiating needs per pilot value network. The design and development pipeline for MIRA has adapted to this, by building features in an iterative fashion, beginning with the simplest possible generic format – for example the free text conversations – and based on the feedback from testing each feature, more specific enhancements are being developed with the help of templates, in order to use a common, yet customizable structure that can be easily shaped per value network or custom need, even by the end users themselves, through the features of MIRA interface and tools.

Conclusions

During the first iteration, T3.1 has made significant progress towards achieving the respective objectives. The aim of T3.1 is to develop user-friendly methods and tools for creating AAS models at factory level to be used in Manufacturing as a Service (MaaS) scenarios. To this end, a first version of the FA³ST CreAItor, allowing users to semi-automatically generate AAS models from their input data, was developed. In a first step, a comprehensive literature review was conducted and exemplary instantiations of SMTs were analyzed to identify challenges and additional requirements for the system. Based on these insights, the architecture of the FA³ST CreAItor was developed. It should be briefly highlighted that the main components of the architecture addressing the identified challenges are the AI Service and the Context Provider. The AI Service uses LLMs and additional context to extract relevant information from the input data, while the Context Provider provides the additional domain knowledge required by the system. The first version of the FA³ST CreAItor supports two of the four intended SMTs: the Digital Nameplate for Industrial Equipment, and the Capability Description SMT. While the Digital Nameplate SMT is already fully supported, the Capability Description SMT is currently supported with some limitations. First, the FA³ST CreAItor does not yet support the evaluation of extracted capability properties in the user interface. Second, the system is limited to support only a subset of the available relationship elements. However, during the first iteration of T3.1 the usability of the FA³ST CreAItor was evaluated by an expert group. These results further highlight the progress made during the first iteration of this task. Lastly, next steps involve enabling the evaluation of the extracted values in the user interface for the Capability Description SMT. Additional steps for the second iteration of T3.1 include supporting the full set of relationship elements as provided by the Capability Description SMT and supporting all SMTs specified in Table 2.

Regarding T3.2, MIRA has implemented crucial features to accommodate AAS-compliant assets in its database and visualize their characteristics properly on a user-friendly interface. The methodology and approach of integration with the T4M marketplace and the respective factory level DTs has been decided, and some preliminary steps towards its implementation have taken place. User conversations have been designed and developed to cover needs on negotiations and orders in agreed collaborations. Also, monitoring tools have been enhanced to align with the aim of the project. For the second iteration, the integration with factory level DTs and the T4M marketplace will be finalized, while all the MIRA features will be finalized and enhanced to cover all necessary pilot needs. A complete alignment with AAS-compliant assets generated in the scope of the project is expected by the end of development.

References

[1] L. Stojanovic *et al.,* "Methodology and Tools for Digital Twin Management—The FA3ST Approach," *IoT*, vol. 2, no. 4, pp. 717–740, 2021, doi: 10.3390/iot2040036.

[2] V. Siatras, E. Bakopoulos, P. Mavrothalassitis, N. Nikolakis, and K. Alexopoulos, "On the Use of Asset Administration Shell for Modeling and Deploying Production Scheduling Agents within a Multi-Agent System," *Applied Sciences*, vol. 13, no. 17, p. 9540, 2023, doi: 10.3390/app13179540.

[3] S. Cavalieri and M. G. Salafia, "A Model for Predictive Maintenance Based on Asset Administration Shell," *Sensors (Basel, Switzerland)*, vol. 20, no. 21, 2020, doi: 10.3390/s20216028.

[4] T. Kreuzer, P. Papapetrou, and J. Zdravkovic, "Artificial intelligence in digital twins—A systematic literature review," *Data & Knowledge Engineering*, vol. 151, p. 102304, 2024, doi: 10.1016/j.datak.2024.102304.

[5] C. Tinnes, M. Ristin, U. Hohenstein, K. Fathi, and H. W. van de Venn, "From Unstructured Product Descriptions to Structured Data for Industry 4.0 with ChatGPT," in *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*, St. Louis, MO, USA, 2024, pp. 1–8.

[6] D. Shi, O. Meyer, M. Oberle, and T. Bauernhansl, "Dual data mapping with fine-tuned large language models and asset administration shells toward interoperable knowledge representation," *Robotics and Computer-Integrated Manufacturing*, vol. 91, p. 102837, 2025, doi: 10.1016/j.rcim.2024.102837.

[7] A. Köcher *et al.,* "A reference model for common understanding of capabilities and skills in manufacturing," *at - Automatisierungstechnik*, vol. 71, no. 2, pp. 94–104, 2023, doi: 10.1515/auto-2022-0117.

[8] M. Volkmann, *Modellierung von Dienstleistungen, Capabilities und Skills zur Einbindung von Unternehmen in eine marktplatzbasierte Fertigung*: VDI Verlag, 2024.

[9] M. Winter *et al.,* "Methods to describe, assign and derive capabilities from the capability, skill and service (CSS) model," 2024.

