RAASCEMAN

Resilient and Adaptive Supply Chains for Capability-based Manufacturing as a Service Networks

Grant Agreement No. 101138782

Deliverable 2.2

Models for Product Digital Twin

Funded by
the European Union

Project title	**RAASCEMAN - Resilient and Adaptive Supply Chains for Capability-based Manufacturing as a Service Networks**
Grant Agreement number	101138782
Funding scheme	Call: HORIZON-CL4-2023-TWIN-TRANSITION-01 Topic: HORIZON-CL4-2023-TWIN-TRANSITION-01-07
Project duration	1 September 2024 – 31 August 2027 (36 months)
Project coordinator	DFKI – Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Deliverable number	**D2.2**
Title of the deliverable	**Models for Product Digital Twin**
WP contributing to the deliverable	**WP1, WP2, WP3, WP4, WP5**
Deliverable type	R – Document, Report
Dissemination level	PU – Public
Due submission date	**30 April 2026**
Actual submission date	**28 April 2026**
Partner(s)/Author(s)	**CEA** (Quang-Duy Nguyen, Kunal Suri, Guéréguin Der Sylvestre Sidibe)
Internal reviewers	**FM** (Michel Schaffers, Camilo Velazquez, Apoorva Oak)
Final approval	**Yes**

History of changes		
When	**Who**	**Comments**
06/07/2025	CEA	Initial Table of Content and structure
17/07/2025	CEA	Version 0.1: Section 1, 2, 3, Appendix B, C are fulfilled. Section 4.1 and 4.2 are fulfilled.
31/07/2025	CEA	Version 0.2: Section 5 adds the Model Lake description. Appendix A and Appendix B.1 are fulfilled.
27/08/2025	CEA	Version 0.3: All Universal-Applicable submodel templates of Section 4.3 are fulfilled.
30/10/2025	CEA	Version 0.4: Section 4.3 is filled with specific submodels for some product types.
19/01/2025	CEA	Version 0.5: Add a sample of PDT information model in Section 4.3. Move Glossary from top to bottom after Conclusion.
30/02/2025	CEA	Version 0.6: Add and fulfill Section 5.2. Update Section 3.3 with T3.3O2 and T3.3O3. Update Section 4.3 with submodel Quality and Commercial Information. Fulfill Conclusion.
17/03/2025	CEA	Version 0.7: Add and fulfill Section 5.3 and Appendix B.2.
04/04/2026	CEA	Version 0.8: CEA internal review phase before submission to the designated consortium reviewer.
24/04/2026	CEA, FM	Version 0.9: Modify the document according to reviews.
28/04/2026	CEA	Version 1.0: Final approval and ready to submit.

Confidentiality	
Does this report contain **confidential** information?	Yes ☐ No ☑
Is the report **restricted** to a specific group?	Yes ☐ No ☑ *If yes, please precise the list of authorized recipients*:

Table of Contents

Funded by
the European Union

Executive Summary

Deliverable 2.2 (D2.2) focuses on describing and developing the concept related to Product Digital Twin (PDT), which includes the PDT information model, along with the method to apply it on the applications (and use cases) from the RAASCEMAN project from the relevant consortium partners. This deliverable provides the approach and a tooling system that supports both the theoretical propositions and methodology developed as part of task 2.2.

To achieve the above-mentioned objectives, this document is organized as follows: Section 1 introduces the methodology for deploying Task 2.2 to achieve its pre-defined goals. Section 2 presents a state-of-the-art related to PDT, Digital Product Passport (DPP), and the prior tools and methods to design and develop PDT and DPP. Section 3 analyses the applications and use cases from RAASCEMAN to determine the requirements related to PDT, which is based on the work done by the RAASCEMAN consortium during WP1. Section 4 proposes a unified PDT information model and design approach. Section 5 proposes a new tooling system that supports PDTs deployment. Section 6 concludes this deliverable and opens a discussion.

Furthermore, as part of Task 2.2, several scientific works were developed. As of April 2026, one of the peer-reviewed papers has been published and two have been accepted to be published in well-established international conferences. We have provided more info in the Scientific Works section after Section 6.

Funded by
the European Union

1 Introduction

Task 2.2 (T2.2) is one of the four major tasks defined in Work Package 2 (WP2) of the project RAASCEMAN. WP2, with its objective of creating a collection of knowledge, methods, and tools to support applications and use cases of RAASCEMAN, has a strong interdependency with the other WPs. Figure 1-1 illustrates a part of the Program Evaluation and Review Technique (PERT) chart related to WP1, WP2, WP3, WP4, and WP5. In detail, WP2 relies only on WP1, but WP3, WP4, and WP5 rely on WP2. T2.2 focuses on defining models for Product Digital Twins (PDT), which serve as inputs for applications involved in WP3 and WP4, and use cases deployed in WP5.

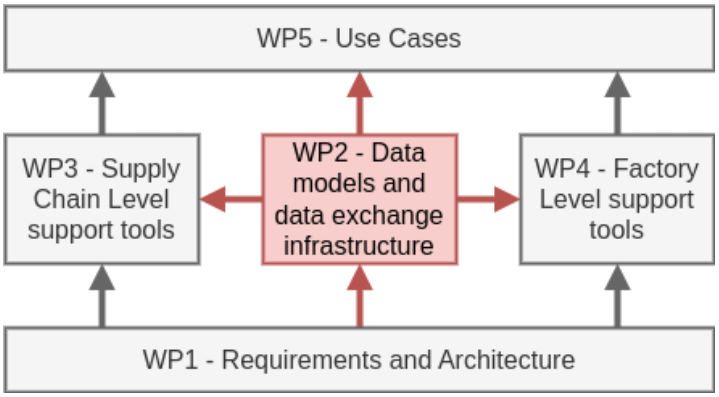


Figure 1-1: PERT chart illustrating the position of WP2

These two following sub-sections present (1) the methodology to deploy the T2.2, and (2) the milestones and objectives of the deployment.

1.1 Methodology

The selected methodology for deploying T2.2 is Agile Scrum with multiple sprints. This approach was chosen because T2.2 spans a long development period and runs in parallel with tasks in WP2, WP3, WP4, and WP5. It requires the T2.2 development team to engage in frequent interactions, as well as regularly update and adjust their work to stay aligned with the progress of other tasks. According to [RAASCEMAN, 2024], T2.2 is developing parallel with:

- T2.1 for nine months
- T2.3 and T2.4 for 13 months
- T3.1 for 15 months
- T3.2, T3.3, and T3.4 for 12 months
- T4.1, T4.2, and T4.3 for 14 months
- T5.1 and T5.2 for three months
- T5.3 for seven months

While retaining the spirit of Agile Scrum, the following principles are considered:

- Meeting the T2.2 team weekly, while meeting other stakeholders from other tasks biweekly.
- Collecting ideas and recommendations from other stakeholders and applying changes.

- Defining a sprint with a time-box of one month. Thus, after 15 months of development, there would be about 10 to 15 sprints.

Figure 1-2 illustrates the principles of the development methodology. The major inputs of T2.2 are from WP1—D1.1, D1.2, and D1.3—which should be analyzed in the first sprint [RAASCEMAN, 2025], [RAASCEMAN, 2025a], [RAASCEMAN, 2025b]. Moreover, the development team of T2.2 needs to continue to study from other sources to understand other tasks which have not yet provided any official documents at T2.2 development time.

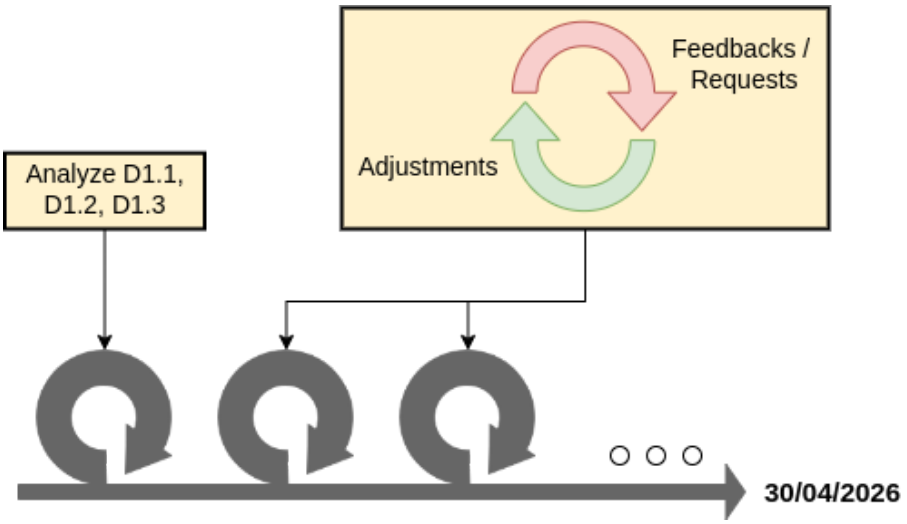


Figure 1-2: Agile Scrum methodology to deploy T2.2

1.2 Objectives

The final aim of T2.2 is to provide knowledge of PDT models to stakeholders involved in the project, which can be divided into the following sub objectives:

a. Defining PDT and clarifying it with other related concepts, such as Digital Product Passport (DPP).
b. Defining a PDT information model based on the data collected from RAASCEMAN and from other credible sources. This step includes the definition of the PDT information model's content, and the vocabulary and format used to encode them.
c. Defining an architecture that enables the design of the PDT information model.
d. Providing a tool that supports the design of the PDT information model.
e. Supporting the deployment of PDT information model in applications and use cases.

Note that most of the requirements defined in D1.3 addressing T2.2 relate to Objective 2.2.b and Objective 2.2.d. However, Objective 2.2.a, Objective 2.2.c, and Objective 2.2.e are also important to make sure the knowledge transfer and T2.2 development are correct.

Funded by
the European Union

2 State of the Arts

The prior arts related to T2.2 can be grouped into three groups: (1) the concepts related to PDT, (2) the vocabulary and standards used for modeling, and (3) the tool that supports the design and implementation of the PDT information model. These following sub-sections are present consecutively in the three mentioned groups.

2.1 Digital Twin, Product Digital Twin, and Digital Product Passport

In the past decade, DT has been emerging as a high-demand topic in both public and private sectors. Thus, there are many definitions available on the market. The following are remarkable ones:

- Micheal Grieves in 2002 coined the original idea which later became the DT concept [Grieves, 2023]. In 2003, the author defined a DT as "*a virtual representation of what has been produced*". In 2011, Micheal Grieves reinforced the idea with more details: "*The Digital Twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level*".
- Digital Twin Consortium (DTC) defines a DT as "*a virtual representation of real-world entities and processes, synchronized at a specified frequency and fidelity*" [DTC, 2020].
- International Electrotechnical Commission (ISO/IEC) defines a DT as "*a digital representation of a target entity with data connections that enable convergence between the physical and digital states at an appropriate rate of synchronization*" [ISO/IEC, 2023].
- French Alliance Industry of Future (AIF) defines a DT as "*a set of digital models representing a real-world entity and is equipped with advanced operations tools, including the ability to understand, analyze, predict, and optimize the operation management of the real entity*" [AIF, 2023].
- Digital Data Chain Consortium (DDCC) defines PDT as "*the collection of all information of a product including the ones of the DPP*" [DDCC, 2024].

Table 2-1 summarizes the DT definitions and compares them. All agree that a DT is a digital representation of an asset. Note that an asset can be categorized as a product, a process, or a resource as classified in the Product-Process-Resource (PPR) paradigm. In which the definition of [Grieves, 2023] and [DDCC, 2024] focuses on product, and only the latter links to DPP.

Table 2-1: Comparison of DT definitions

	Data representation	Connection requirement	Advanced tools Integration	Product-focused	Link to DPP
[Grieves, 2023]	x			x	
[DTC, 2020]	x	x			
[ISO/IEC, 2023]	x	x			
[AIF, 2023]	x		x		
[DDCC, 2024]	x			x	x

As DPP is a growing trend in current manufacturing and sometimes is considered a type of PDT, there exists also some remarkable definitions of DPPs:

- Psarommatis et al. define a DPP as "*a digital representation of a product's information through its lifecycle, from creation to disposal*" [Psarommatis et al., 2024].
- CIRPASS defines a DPP as "*structured with a collection of product-related data with a pre-defined scope and agreed data management and access rights conveyed through a unique identifier*" [CIRPASS, 2024].
- United Nations Transparency Protocol (UNTP) defines a DPP as "*a comprehensive data structure that encapsulates various details pertaining to a product, including its identification details, who issued it, batch information, provenance information, circularity information and a set of verifiable product conformity & sustainability claims*" [UNTP, 2025].
- DDCC defines a DPP as *"DPP is the collection of all product information relevant for circularity and authorities"* [DDCC, 2024].

Table 2-2 summarizes the DT definitions and compares them. Like PDT, a DPP is a digital representation of a product. Moreover, it focuses on the data during the product lifecycle and possesses a unified identifier. This latter feature defines DPP as a PDT instance. Among the definitions, only one of [DDCC, 2024] has an association with PDT.

Table 2-2: Comparison of DPP definitions

	Data representation	Lifecycle	Unified identifier	Link to PDT
[Psarommatis et al., 2024]	x	x		
[CIRPASS, 2024]	x		x	
[UNTP 2025]	x	x	x	
[DDCC, 2024]	x	x	x	x

2.2 Modeling Vocabularies

There exist several vocabularies used to model PDT & DPP as follows:

- Asset Administration Shell (AAS) is a standard first proposed by Platform Industry 4.0 in 2017 and has been continued to be maintained and developed by Industrial Digital Twin Association (IDTA) [IDTA, 2023]. The standard includes five major specifications to develop AAS DTs, of which the first one—Part 1: Metamodel—provides the metamodel and vocabulary to form AAS information models for assets. In detail, an AAS information model is a collection of submodels. Each submodel represents an aspect of its target asset and includes submodel elements containing information related to the aspect. A submodel element can be varied: operation, property, file, collection, etc. Note that an asset can be a product, process, or resource, and the AAS modeling language can be used to model all of them; however, T2.2 focuses only on the product. The AAS ecosystem—including the AAS metamodel and submodel templates provided by the IDTA, its partners, and other organizations and consortia—offers a rich vocabulary for modeling.

- Ontologies and knowledge graphs are well-known tools to represent knowledge and information in computer science. While ontology is a formal, explicit specification of concepts and relationships in a specific domain that provides schema to organize knowledge [Studer et al., 1998], a knowledge graph represents data instances of entities and their relationships following the ontology [Ehrlinger et al., 2016]. CIRPASS, the European project dedicated to the DPP development, promotes the DPP information model implementation using ontologies [CIRPASS, 2024]. In detail, CIRPASS provides a literature review of ontologies for products. They include ontologies and vocabularies for general products (e.g. TOVE, PRONTO, and the GoodRelations schema), for batteries (e.g. BattINFO, Battery Pass, and BVCO), for textiles (e.g. Textile Exchange, Global Organic Textile Standard, and Global Textile Scheme), and for electronic products (e.g. eReuse.org, RePlanIT, and ETIM). Moreover, some ontologies, such as Digital Product Passport Ontology (DPPO) (https://w3id.org/dppo/). the EU DPP Core Ontology (https://w3id.org/eudpp/), are specifically designed to model DPPs. Regarding digital twin modeling, the WoT DT is an ontology designed to model DTs in general [Gozalez-Gerpe et al., 2024], and can therefore be customized to model PDTs.
- ECLASS is a global ISO-compliant data standard that enables classifying and describing products and services in a standardized way. Its classification system consists of four levels: Segment, Main Group, Group, and Subgroup. The classes proposed by ECLASS are at the Subgroup layer, and they are specified by properties. ECLASS Release 15.0 offers 48490 classes and 23910 unique properties. The other elements of the standard, including keywords, values, and units, improve the semantics of properties. ECLASS is accepted and used by over 4000 companies, including some well-known ones such as Siemens, SAP, to name just a few.
- UNTP is a semantic model and ontology proposed by the United Nations Centre for Trade Facilitation and Electronic Business [UNTP, 2025]. Its current version 0.6.0 is rich, well-defined, and focuses on DPP. Moreover, UNTP supports the linked data mechanism and is compatible with other widely accepted standards, such as W3C's Verifiable Credentials Data (VC) and GS1 (Electronic Product Code Information Services) EPICIS-LD.
- Other vocabularies helpful in clarifying one or some aspects of PDT and DPP are: (1) the dataset attributes of GS1 Global Data Model (GSM) (https://www.gs1.org/standards/gs1-global-data-model), and (2) Thing Descriptions [Kaebisch et al., 2023] and Capability Schemas [WebThingsIO, 2025] of W3C Web of Things (WoT).

Table 2-3 summarizes modeling vocabulary: their strengths, weaknesses, and focus. There are five factors that can be used to evaluate them. First, vocabulary contains enough materials to model DPP or has a clear motivation to model DPP. Second, vocabulary contains enough materials to model PDT and/or has a clear motivation to model PDT. Third, the vocabulary was well-developed and maintained correctly. Fourth, vocabulary is widely accepted by the industrial community. Fifth, vocabulary is easy to use in the concept and/or provides enough materials to support users to use them correctly and easily. The evaluation is made by the T2.2 development team.

Table 2-3: Comparison of vocabularies to model PDT & DPP

	DPP supported	PDT supported	Well-developed and maintained	Industry-recognized	Easy and open to use
AAS ecosystem	x	x	x	x	x

Ontologies for PDT/DPP	x	x	Depends	Depends	Depends
ECLASS	x	x	x	x	
UNTP	x		x		x
GS1 GDM	x		x	x	Depends
WoT		x	x		x

2.3 Modeling Tools

PDT information model designers require a graphical tool to create artifacts and communicate their designs to others. Some tools currently available on the market and actively maintained are as follows:

- **Eclipse AASX Package Explorer** (https://projects.eclipse.org/projects/dt.aaspe) is an open-source AAS editor supported and recommended by IDTA [Admin-shell-io, 2025]. It offers a user-friendly graphical interface that provides all the necessary features to effectively model an AAS. The tool supports the AASX format, with the AAS data encoded in either JSON or XML. It also integrates viewers for additional resources such as PDF, PNG, and STL files.
- **Papyrus4Manufacturing** (https://eclipse.dev/papyrus/components/manufacturing/) is an open-source AAS development framework based on model-driven engineering principles [Dhouib, 2023]. It is developed and maintained by CEA List and is a member of the Papyrus's ecosystem. It provides features that allow PDT information model designers to create AAS information models using UML. The tool supports mainly UML format and includes a plugin to convert a UML file to AASX.
- **Protégé** (https://protege.stanford.edu/) is an open-source ontology editor developed and maintained by Stanford University [Musen et al., 2015]. It is widely used in the knowledge engineering community, not only for its user-friendly interface but also for its rich support of reasoning engines (e.g. HermiT, Pellet, and ELF). The tool is easy to use and works with various ontology languages, RDF(S) and OWL, and serializations, including Turtle, RDF/XML, JSON-LD, and N-Triples.
- **Vocbench** (https://vocbench.uniroma2.it/) is defined as a web-based tool for developing and managing vocabularies such as ontologies, thesauri, lexicons, and RDF datasets [Stellato et al., 2020]. One of its remarkable features is its collaborative editing environment supporting multilingual content.
- **FluentEditor** (https://www.cognitum.eu/semantics/fluenteditor/) is an ontology editor that allows users to write axioms in Controlled Natural Language (CNL) and converts them into OWL/RDF. Its main advantage is that CNL provides a more human-readable syntax.
- **TopBraid Composer** is an ontology editor which is part of the semantic technology suit provided by TopQuadrant (http://www.topquadrant.com/topbraid/). The editor combined with other tools in the suite provide an enterprise-grade solution to real-world semantic use cases. Although they are known as paid tools, a free edition with limited features is also available.

Table 2-4 summarizes the tools. The user experience regarding a tool depends not only on the tool itself, but also on the user's preferences. Thus, the comparison criteria primarily focus on the tool's

support for the target model, its platform or operating system compatibility, its licensing model (free or paid), and the supported formats.

Table 2-4: Comparison of some modeling tools for AAS or ontologies

	Target model	Platform/OS support	Desktop or Web	Licensing	Format
AASX Package Explorer	AAS	Windows	Desktop	Free	AASX, JSON, XML
Papyrus4Manufacturing	AAS	Windows, MacOS, Linux	Both	Free	UML (XMI), JSON
Protégé	Ontology	Windows, MacOS, Linux	Both	Free	RDF/XML, Turtle, JSON-LD, OWL
Vocbench	Ontology	Windows, MacOS, Linux	Web	Free	RDF/XML, Turtle, JSON-LD
FluentEditor	Ontology	Windows	Desktop	Free	CNL, RDF/XML, Turtle, OWL
TopBraid Composer	Ontology	Windows, MacOS, Linux	Desktop	Free & Paid	RDF/XML, Turtle, JSON-LD, OWL

3 Requirement Analysis

Since the tasks in WP3, WP4, and WP5 depend on WP2 in general and on T2.2 in particular, analyzing their requirements is essential. This section is not a duplication of the results from WP1 but builds upon them, with a specific focus on the PDT. In other words, in this section we analyze WP1 results from PDT viewpoint. The inputs for this analysis are taken from four main sources: (1) D1.1, D1.2, D1.3 of WP1, and D2.1 of WP2, (2) communication documents exchanged between WP2 and other WPs, (3) knowledge gathered from meetings with stakeholders, and (4) additional academic and enterprise resources. The following five sub-sections provide five analyses of WP1, WP2, WP3, WP4, and WP5.

3.1 WP1 Analysis

D1.1, D1.2, and D1.3 are respectively the results of T1.1, T1.2, and T1.3 in WP1. While the first two will be presented in this sub-section, D1.3 will be presented in Section 3.3 and Section 3.4 since it contains mainly the information for tasks of WP3 and WP4.

D1.1 reports the requirements and specifications of RAASCEMAN [RAASCEMAN, 2025]. Dedicated to WP2, they are the requirements for applying the Capability-Skill-Service (CSS) model and PDT concept in manufacturing. The CSS model extends the well-known Product-Process-Resource representation paradigm (PPR) by introducing new concepts of capabilities, skills, and services. Since the product is a fundamental part of the PPR, PDT relates to CSS. Moreover, D1.1 briefly points out the expected contributions of PDT in recapitulated and analyzed in Table 3-1.

Table 3-1: PDT-related points from D1.1

Code	Statement	Analysis
D1.1P13.1	PDT contains "relevant information to perform step-by-step quotation generation, prepare production plans, and execute manufacturing at the field level".	Information contributed to a **quotation** and **production order** including **product identifier**, **product characteristics**, and Bill of Materials (**BOM)**.
D1.1P13.2	PDT contains "additional information such as usage information to enable companies to evaluate if remanufacturing is possible, and information related to other lifecycle steps of the product including decommissioning".	Information for PDT includes **usage information**, and **lifecycle history**.
D1.1P49.1	PDT must be extended with production information in terms of manufacturing goals, which will be "*used in the trade-off analysis*" to help a service requester define what-if scenarios when using a decision support tool.	Should NOT include manufacturing goals in the PDT information model. Instead, it should include the Bill of Services (**BOS**) to align with the CSS model.
REQ1.2	PDT shall allow a "*service requester/provider participating in the MaaS network to describe*	No detailed information is provided.

Code	Statement	Analysis
	information related to their product, so that the product can be used easily over its full lifecycle".	
REQ1.2.1	PDT shall "*include the CSS model of the product that encompasses the standardized AAS models and submodels*".	Should include a part of the CSS model, such as **BOS**. Other information about capabilities and skills should be associated with resources or processes, instead of products.
REQ1.2.2	PDT shall "*include editable models that can store values related to relevant features like process duration, cost and carbon footprint, to name a few, based on the needs of the product*".	Should include information about the **carbon footprint** and **cost**. Information about duration is included in a **quotation** or **production order** but should NOT be directly related to the product.
REQ1.2.3	PDT shall "*include editable models to store information about skills and values related to relevant features like process duration, cost and carbon footprint, to name a few, based on the needs of the product*".	Information about skills should be associated with resources.
REQ1.2.4	PDT shall "*have a GUI or editable models to specify products based on standards, such as ECLASS*".	Address **Objective 2.2.d** of T2.2. Also, it addresses T2.3 and T2.4.
REQ1.2.5	PDT shall "*provide mechanisms to provide an aggregated view of different information such as BoM, BoP, quality control, to name a few*".	Address T2.3 and T2.4.
REQ1.2.6	PDT shall "*include editable models to store information about the different steps used in the manufacturing of a product, in order to create a holistic DPP*".	In CSS, the product should not include **BOS** to replace the manufacturing steps information.
REQ1.2.7	PDT shall "*include editable models to aggregate information about the different digital twins (such as system, process models) to track the entire lifecycle of a product*".	Information for DPP for product **lifecycle history**.
REQ3.2.1.4	PDT data shall be extended with manufacturing goal metrics to feed the decision support tool.	Should NOT include manufacturing goals in the PDT information model. Instead, it should include the **BOS**.
REQ4.1.2.4	PDT data should be updated with data (e.g. carbon footprint, environmental, and health impact) from the recommendation engine.	Information for DPP including **carbon footprint**, **environmental**, and **health impact**.

Some information from Table 3-1 should be clarified as follows:

- Regarding D1.1P13.1, it is necessary to distinguish quotation and production order. A quotation is sent from the sales department of a factory to a client to clarify a product's characteristics together with the information related to a transaction such as the price of product and term of sales. With such information, the client can send a customer order back to the sales department. These activities are at the supply chain level. The factory having a customer order will generate a production order to the manufacturing plant for a concrete production. The production order includes information about production quantity, the target product's characteristics, and Bill of Materials (**BOM**). This activity is at the factory level. Figure 3-1 illustrates the positions of quotation, customer order, and production order in a typical manufacturing request.

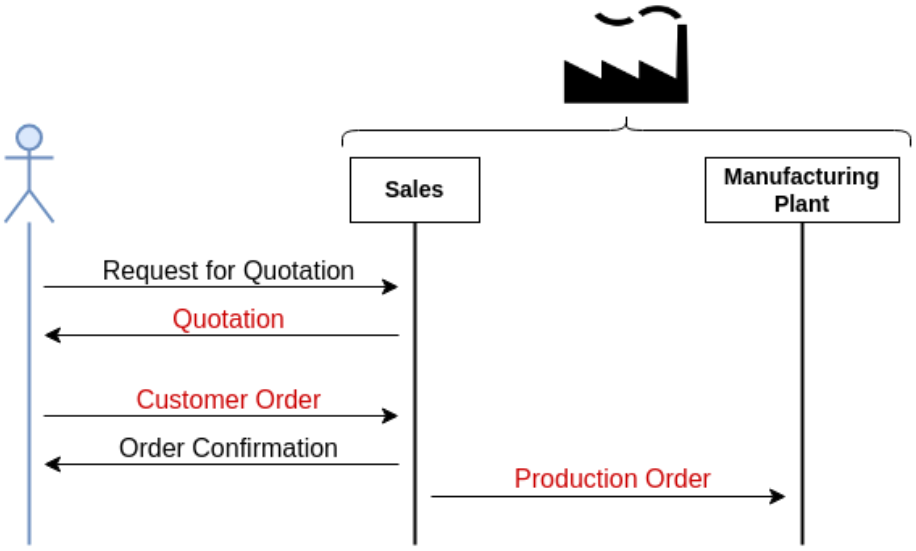


Figure 3-1: Sequenced steps of Quotation, Customer Order, and Production Order

- Regarding D1.1P49.1 and REQ3.2.1.4, the Bill of Services (**BOS**) is a new term aligned with the CSS model. In detail, a product can have a list of required services needed to match offered services from service providers.

D1.2 presents briefly the use cases in WP5, their elements, and the Key Performance Indicator (KPI) to evaluate them [RAASCEMAN, 2025a]. The deliverable indicates some requirements related to PDT as recapitulated and analyzed in Table 3-2.

Table 3-2: PDT-related points from D1.2

Code	Statement	Analysis
D1.2P16.1	DPP can "*document component serial numbers, tracker details, and lock traceability (e.g. in case of lost keys)*"	Information for DPP includes **BOM**, **lifecycle tracking**, and **location tracking**.
D1.2P16.2	DPP can "*track assembly data, such as BOM and quality information, used tools*"	The Information for DPP includes **BOM**. The **quality information** should also be included

		to reflect the quality of manufacturing but not only the assembly. The information related to assembly resources should NOT be associated with the product.
D1.2P16.3	DPP can provide "*assembly and maintenance information*"	Information for DPP includes **maintenance information.**
D1.2P16.4	DPP can provide "*lifecycle data to support remanufacturing processes*"	Information for DPP includes **lifecycle history**.

3.2 WP2 Analysis

WP2 consists of three other tasks in addition to T2.2. In which T2.3 and T2.4 are rather on technical contributions, software, hardware, infrastructure) which are out of the scope of this analysis.

T2.1 focuses on the CSS model, which has a strong relation to PDT, as mentioned in Section 3.1. Figure 3-2 illustrates the relations between components of PPR and CSS. It is sourced from the discussion paper from [Platfform Industrie 4.0, 2022].

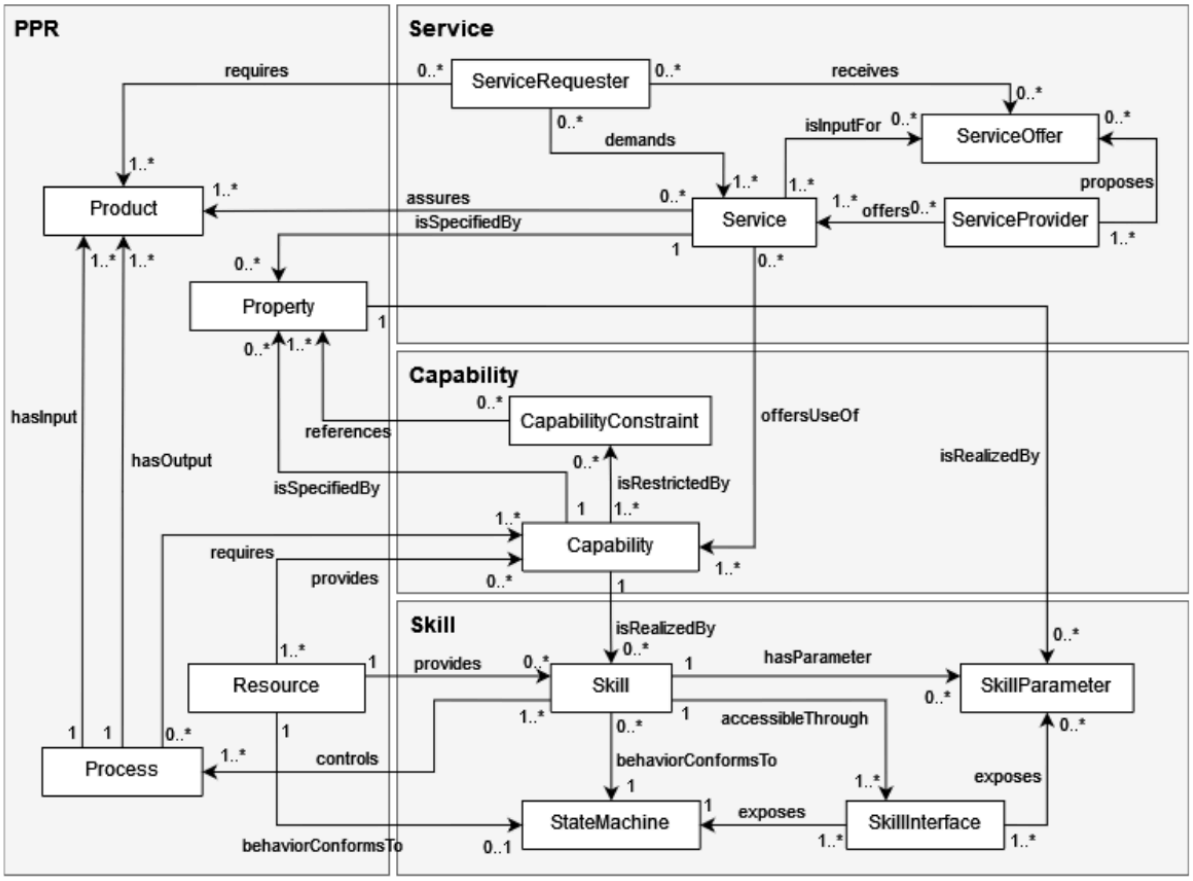


Figure 3-2: Relation between PPR paradigm and CSS model

Product and its digital version, i.e. the PDT, have relations between service and process. Simply put:

- One or multiple services ensure the production of one or multiple products. In other words, a product requires a list of services to be produced successfully.
- One process takes one or several parts as input and produces one or multiple products. Note that part is a type of immediate product.
- Service requester requires a product prototype to form its customer order.

3.3 WP3 Analysis

WP3 is composed of four tasks corresponding to four applications at the supply chain level: ***impact prediction tool*** (T3.1), ***decision support tool*** (T3.2), ***audit tool*** (T3.3), and ***recommendation engine*** (T3.4). They are described in D1.3 [RAASCEMAN, 2025b]. Some of them also provide input and output information.

The aim of the impact prediction tool is to calculate the probability and impact of unexpected events happening in the supply chain. The information proposed by the T3.1 stakeholders shows that the input (e.g., T3.1I1) and output (e.g., T3.1O1) of this application vary and can have some indirect requirements from PDT. Table 3-3 analyzes the requirements.

Table 3-3: PDT-related points from T3.1

Code	Input	Analysis
T3.1I1	Product delivery delays logs	Logs of **customer orders** for completion include **product identifiers**.
T3.1I2	Machine downtime	No information related to PDT.
T3.1I3	Operator availability	No information related to PDT.
T3.1I4	Product cost	Information on **customer orders** includes **product costs**.
T3.1I5	Order history and upcoming orders	Logs of **customer orders** include **product characteristics**.
T3.1I6	Raw material delivery time or delay	Logs of raw material delivery time include raw material information (**product identifier**).
T3.1I7	Frequency of raw material delay	Logs of raw material delivery time include raw material information (**product identifier**).
T3.1I8	Supplier location	No information related to PDT.
T3.1I9	Missing parts	List of missing parts includes part information (**product identifier**).
T3.1O1	Type of impact	No information related to PDT.
T3.1O2	Possibility of impact occurrence	No information related to PDT.
T3.1O3	Actual impact	No information related to PDT.

The aim of the decision support tool is to provide a trade-off evaluation and simulation based on context and manufacturing goals defined by a client. The tool takes as input the output from T3.1 and the results of planning and capability matching from T4.1 and T4.2. Table 3-4 analyses some defined input of T3.2 found in D1.1 and D1.3.

Table 3-4: PDT-related points from T3.2

Code	Input	Analysis
D1.1P60	Embedded manufacturing goal metrics	**BOS**
D1.3P46	Product parts	**BOM**

The aim of the audit tool is to evaluate a new participant based on the profiles and histories of all participants. The information proposed by the T3.3 stakeholders shows that the input and output of this application vary and can have some indirect requirements from PDT. Table 3-5 recaps the information and analyzes the requirements.

Table 3-5: PDT-related points from T3.3

Code	Input	Analysis
T3.3I1	Product properties / feature *(To search in database for similar properties/feature from historic parts)*	List of properties and features of historical parts include **product identifiers** and product **characteristics**.
T3.3I2	Amount *(To check if the production capacity is the for)*	Logs of production capacity are associated with **product identifiers**.
T3.3I3	Business properties *(For example: Cost, Delivery date, etc.)*	No information related to PDT.
T3.3I4	Constraints *(To check if the production capacity is the for)*	No information related to PDT.
T3.3I5	Data (orders) *(Orders to see what was ready done)*	Logs of **customer orders** include **product identifiers** and **product characteristics**.
T3.3I6	*Code (machine code)* *(Orders to see what was already done)*	No information related to PDT.
T3.3I7	History *(To extract the capability of suppliers)*	No information related to PDT.
T3.3O1	Rating (evaluation score) *(Rate each possible factory inside the platform)*	No information related to PDT.
T3.3O2	The AAS/PDT/DPP product must also collect information on the quality of the finished product in three categories. (A-grade goods, B-grade goods, Rejects)	**Finish product quality** can have value “A-grade”, “B-grade”, or “Reject”.

T3.3O3	The AAS/PDT/DPP product must also get information about the Date of the Delivery of the product, from whom it was requested	Information about **Requested Delivery Date** and **Actual Delivery Date** in **Commercial Information**.

The aim of the recommendation engine is to propose a list of suitable service providers to a client based on the request. The information proposed by the T3.4 stakeholders shows that the input and output of this application vary and can have some indirect requirements from PDT. Table 3-6 recaps the information and analyzes the requirements.

Table 3-6: PDT-related points from T3.4

Code	Input	Analysis
T3.4I1	Steps to produce the product *(Steps that are necessary for the production)*	**BOS**
T3.4I2	*Ratings* *(The result from audit tool to find the best result)*	No information related to PDT.
T3.4I3	Business constraints *(For example: Cost, CO2, time, etc.)*	Constraints include various information about products such as **product cost** and **carbon footprint**. The unit production time should be in **production order**.
T3.4O1	Capability routes *(Customer can choose between these results)*	**BOS**

3.4 WP4 Analysis

WP4 is composed of three tasks corresponding to three applications at the factory level: procedure and capability matching (T4.1), dynamic planning & scheduling (T4.2), and dynamic execution (T4.3). They are described in D1.3 [RAASCEMAN, 2025b].

The aim of the procedure and capability matching tool is to assign operations to resources based on offered and required capabilities matching. Table 3-7 analyses some defined input and output of T4.1. found in D1.3.

Table 3-7: PDT-related points from T4.1

Code	Input	Analysis
T4.1I1	MaaS or MES Service requests	**Production order** includes **product identifiers** and **product characteristics**.
T4.1I2	Live shop floor data *(Machine states, tool wear, operator availability)*	No information related to PDT.
T4.1I3	Trust and risk signals *(Supplier/manufacturer trust scores from Audit Tool, impact prediction risk flags)*	No information related to PDT.

T4.1O1	Capability match Manifest (JSON) *(Task ID & required process steps; ranked candidate resources with match scores; constraint/feasibility diagnostics, such as missing skills, overload, etc.)*	No information related to PDT.
T4.1O2	Alerts *(Example: "no feasible match" or "data missing")*	No information related to PDT.
T4.1O3	Metrics Steam *(Matching latency, query statistics to monitoring dashboard)*	No information related to PDT.
T4.1O4	Human-readable summary *(NL explanation of top matches for UI)*	No information related to PDT.

The aim of the dynamic planning and scheduling tool is to (re)program a new production schedule based on the current state of the shop floor and the new production order. The information proposed by the T4.2 stakeholders shows that the input and output of this application vary and have indirect requirements from PDT. Table 3-8 recaps the information and analyzes the requirements.

Table 3-8: PDT-related points from T4.2

Code	Input	Analysis
T4.2I1	Orders in Factory (current state) *(The already planned orders)*	**Production orders** already planned include **product identifier**, **product characteristics**, and **BOM**.
T4.2I2	Model of the factory (current state) *(Containing available machines and resources)*	No information related to PDT.
T4.2I3	List of available resources *(List of available resources)*	No information related to PDT.
T4.2I4	New order *(The order that needs to be planned in)*	**Production order** includes information about the **product identifier**, **product characteristics**, and **BOM**.
T4.2I5	Planning goal/constraints *(To lead the planning)*	**Product identifiers**, **product characteristics**, and quantity of products.
T4.2O1	Production plan/schedule *(The calculated plan to apply)*	Plan includes the **product identifier**, **product characteristics**, and quantity of products.
T4.2O1	KPI *(The newly determined KPIs for the current schedule)*	No information related to PDT.
T4.2O3	Planning error (flag) *(To signal that planning is not possible)*	Error report includes **product identifier**, **product characteristics**, and quantity of products.

Funded by the European Union

The aim of the dynamic execution tool is to process the production following a production schedule/planning. The information proposed by the T4.3 stakeholders shows that the input and output need some indirect requirements from PDT. Table 3-9 analyzes the requirements.

Table 3-9: PDT-related points from T4.3

Code	Input	Analysis
T4.3I1	Parameter set *(This could be various information to fulfill the execution)*	**Product characteristics** can be used as parameters for execution.
T4.3I2	Monitoring data *(To feed PDT/DPP about production process)*	**Production trace** is obtained from production processes.

3.5 WP5 Analysis

The analysis of WP5 is organized by three use cases: **ASKA Bike**, **AUMOVIO**, and **Interconnected Pilot Lines**. The information presented in this section derives from D2.1 which is input provided by the three use cases' members.

ASKA is an innovative Belgian *Speed-Pedelec* (pedal electric bicycle) manufacturer. The ASKA Bike case expects new technologies from RAASCEMAN to realize resilient and sustainable production, especially in supporting bike assembly/disassembly, maintenance, and traceability. The use case has several requirements as presented and analyzed in Table 3.10.

Table 3-10: PDT-related points from the ASKA Bike use case

Code	Requirements	Analysis
UC1.R1	Need flexible capacity production partners. Flexible production rate, easily introducing second source	No information related to PDT.
UC1.R2	Independent from one supplier	No information related to PDT.
UC1.R3	No standard for documentation	This non-featured requirement can be resolved using AAS as this standard itself is a kind of standardized documentation method. No information related to PDT.
UC1.R4	Long lead time to get quotation	No information related to PDT.
UC1.R5	Difficult to find one supplier doing all	No information related to PDT.
UC1.R6	Difficult to evaluate supplier/compare quotation	No information related to PDT.
UC1.R7	Sharing tooling not possible	No information related to PDT.

Funded by the European Union

UC1.R8	Rework needs to be possible	**BOM** and **BOS** are required by a rework. Any failure in the supply chain should be recorded in **traceability** to support rework.
UC1.R9	A platform to describe production processes for each product	Production processes are replaced by **BOS** in CSS.
UC1.R10	Employ low skilled people	**Operational Instructions** should include clear and detailed guidelines at different phases.

AUMOVIO is a global supplier of automotive electronics that provides production facilities for electronics and mechanical fuel transport units. The AUMOVIO case demands the use of new RAASCEMAN technologies for proper production planning, dispatch, and invoicing for adaptable and sustainable supply chains. The use case has several requirements as presented and analyzed in Table 3.11.

Table 3-11: PDT-related points from the AUMOVIO use case

Code	Requirements	Analysis
UC2.R1	Electronic Data Interchange (Call-Offs)	No information related to PDT.
UC2.R2	Customer Plants	No information related to PDT.
UC2.R3	Customer Evaluation	No information related to PDT.
UC2.R4	Bill of Materials (BOM/MBOM)	**BOM**
UC2.R5	Manufacturing Routings	The information about production processes is not related to PDT in CSS. However, **traceability** includes information referenced to process routing.
UC2.R6	Cycle Times and Processes	No information related to PDT.
UC2.R7	Release Process	No information related to PDT.
UC2.R8	Manufacturing Version	No information related to PDT.
UC2.R9	Manufacturing Line Change (MLC) Status	No information related to PDT.
UC2.R10	Scheduled Production	No information related to PDT.
UC2.R11	Packing Information	**Traceability** should include information related to packaging.
UC2.R12	Quality Block	Information related to **quality** of a product.
UC2.R13	Minimum Production Quantity	No information related to PDT.

UC2.R14	Warehouse Value 1	No information related to PDT.
UC2.R15	Warehouse Value 2	No information related to PDT.
UC2.R16	Supplier information	**Nameplate** should include information related to supplier.
UC2.R17	Delivery Information	**Traceability** should include information related to delivery.
UC2.R18	Expiration Data	No information related to PDT.
UC2.R19	Sub-Assembly Warehouse Value	No information related to PDT.
UC2.R20	Sub-Assembly Expiration Time	No information related to PDT.
UC2.R21	Warehouse Value for Specific A3C	No information related to PDT.
UC2.R22	Customer Release Information	No information related to PDT.
UC2.R23	Overall Equipment Effectiveness (OEE)	No information related to PDT.
UC2.R24	Work In Progress (WIP)	No information related to PDT.
UC2.R25	Production Orders	**Production order** includes information about the **product identifier**, **product characteristics**, and **BOM**.

Interconnected Pilot Lines is a connected network of testbeds applying Manufacturing-as-a-Service (MaaS). The members of the network are **DFKI**, **Flanders Make**, **CIIRC CTU**, and **RPTU**. The requirements of this use case are listed and analyzed as in Table 3.12.

Table 3-12: PDT-related points from the Interconnected Pilot Lines use case

Code	Requirements	Analysis
UC3.R1	Cells	No information related to PDT.
UC3.R2	AddOns	No information related to PDT.
UC3.R3	Tools	No information related to PDT.
UC3.R4	ScrewDrivers	No information related to PDT.
UC3.R5	Grippers	No information related to PDT.
UC3.R6	Clamps	No information related to PDT.
UC3.R7	Warehouse	No information related to PDT.
UC3.R8	Workspace	No information related to PDT.

UC3.R9	Robot	No information related to PDT.
UC3.R10	Cell Skills	No information related to PDT.
UC3.R11	Modules	No information related to PDT.
UC3.R12	Cameras	No information related to PDT.
UC3.R13	Worker	No information related to PDT.
UC3.R14	3D Printer	No information related to PDT.
UC3.R15	Transport system (intra)	No information related to PDT.
UC3.R16	Autonomous Mobile Robot (AMR)	No information related to PDT.
UC3.R17	Projector	No information related to PDT.
UC3.R18	cellA (Cell)	No information related to PDT.
UC3.R19	cellB (Cell)	No information related to PDT.
UC3.R20	robot1 (Robot)	No information related to PDT.
UC3.R21	addOn1 (Warehouse)	No information related to PDT.
UC3.R22	Productionisland (modules)	No information related to PDT.
UC3.R23	Manual (Dis-)Assembly Station	**Traceability** should include information about different stations.
UC3.R24	Capability catalog	No information related to PDT.
UC3.R25	Instant cost estimation	No information related to PDT.
UC3.R26	Lead-time projection	No information related to PDT.
UC3.R27	Specifications logger	No information related to PDT.
UC3.R28	Supplier quality rating	No information related to PDT.
UC3.R29	Quality requirement	No information related to PDT.
UC3.R30	Execution-time confirmation	No information related to PDT.
UC3.R31	Purchase-order quotation	**Production order** and **quotation** includes information about the **product identifier**, **product characteristics**, and **BOM**.
UC3.R32	Job execution status	**Traceability** should include information about the current status at different phases.

UC3.R33	Issue gateway	No information related to PDT.
UC3.R34	Deviation management	No information related to PDT.
UC3.R35	Change control	No information related to PDT.
UC3.R36	Quality report	No information related to PDT.
UC3.R37	Customer audit gateway	No information related to PDT.
UC3.R38	Digital passport	Part of PDT.
UC3.R39	Assembly (flexible)	No information related to PDT.
UC3.R40	Disassembly (flexible)	No information related to PDT.
UC3.R41	3D Print	No information related to PDT.
UC3.R42	Quality Control	**Quality** should include information that support quality control.
UC3.R43	Transport	**Traceability** should include information about transport and logistics.
UC3.R44	Assembly (manual)	**Operational Instructions** should include assembly instructions to do assembly.
UC3.R45	Disassembly (manual)	**Operational Instructions** should include disassembly instructions to do disassembly.

4 Product Digital Twin Model Design

This section presents the main contribution of T2.2. It first defines the major and minor concepts that build up the understanding of PDT and its related application. Second, this section proposes a design and implementation method used for collective use of RAASCEMAN. Third, the list of submodels constituting the information model of PDT will be detailed.

4.1 Concepts Definition

Instead of redefining every concept, D2.2 reuses those appropriate to the direction of RAASCEMAN. The selected concepts should be used and understood by all stakeholders of the project. Some concepts are clarified when putting them next to one another. In this sense, the PDT and DPP are defined following [DDCC, 2024]:

> **Product Digital Twin (PDT)** is the collection of all information about a product including the ones of the DPP. **Digital Product Passport (DPP)** is the collection of all product information relevant to circularity and authorities.

This definition focuses on the data and information conveyed in PDT and DPP. It highlights the two mandates of a DPP, which are (1) the lifecycle of a product and (2) unified identifier. It can also clarify that PDT should contain information for other applications (e.g., information for quotation generation) and is not limited to data from DPP. Figure 4-1 illustrates the idea that the PDT information model contains the DPP information model.

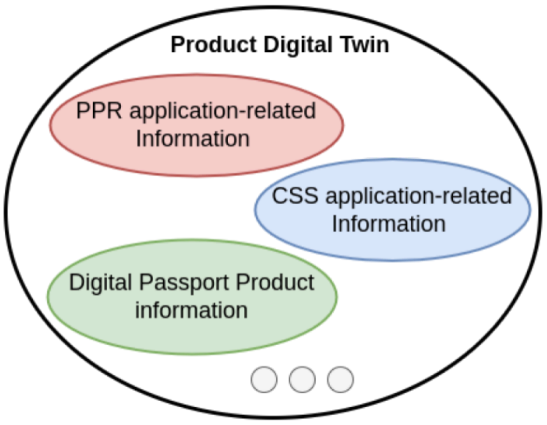


Figure 4-1: DPP information model as a subset of PDT information model

It is worth mentioning that in some applications, such as product tracking, a DPP is actually a PDT instance. However, a PDT instance is not always the same as a DPP instance, for example, a PDT used in virtual commissioning doesn't need to define or include a related DPP [Nguyen, 2024].

The second point that needs to be clarified is based on the difference between a ***Product Instance*** and a ***Product Prototype***. Indeed, while a product instance is a physical product having a unified identifier, a product prototype can exist physically or only in virtual form to represent the characteristics of all product instances. For example, a factory can create a prototype of a pen, which is a blueprint to produce pen instances. A pen instance has a unified identifier to distinguish it from another pen instance and will be put into use after production. In this sense, a PDT of a product instance and a PDT of a product prototype should also be distinguished as following [Grieves, 2023]:

> **Product Digital Twin Prototype** is a digital model of a product prototype that represents all possible generic configurations, features, and behaviors of the product, without being tied to a specific physical item and identification. **Product Digital Twin Instance** is the digital representation of a specific product instance that has been manufactured and exists in the real world, associated with a unified identifier.

While some applications are interested in PDT instances, such as product tracking, some require PDT prototypes, such as quotation generation.

The third concept to discuss is the ***PDT Template***. It can be considered as a data schema consisting of multiple AAS submodel templates. This template can be used to instantiate either a PDT prototype or a PDT instance using actual data from design or production. The following definition is dedicated to the use of AAS and clarifies its relations with the AAS submodel template and instances:

> **Product Digital Twin Template** is composed of AAS submodel templates, which can be used to instantiate Production Digital Twin instance/prototype composed of AAS submodel instances.

4.2 Design and Implementation Method

The design and implementation of PDT can be separated into one for PDT template and one for PDT instance. Figure 4-2 illustrates the two different flows corresponding to the two activities.

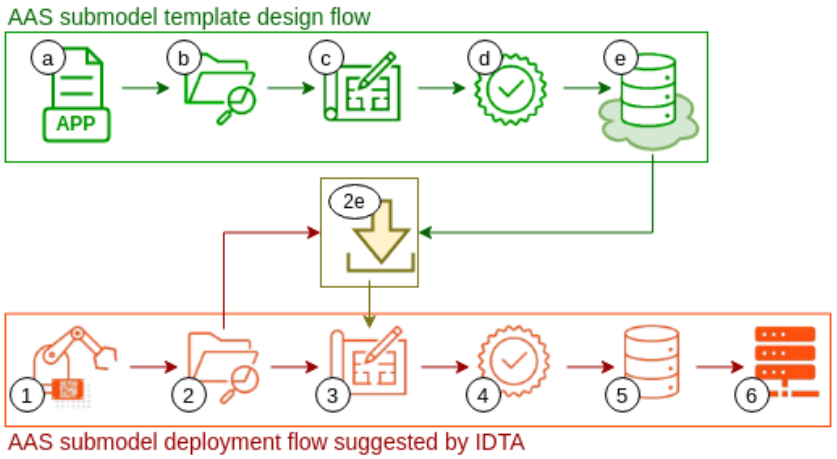


Figure 4-2: Flows for AAS submodel template and instance development

The upper flow describes the steps to develop an AAS submodel template for PDT template. The actors in this flow are mainly engineers in WP2, and they will be considered ***AAS Template Designers***. They need to understand applications and design application-driven submodel templates for collective use. Then, each factory or application user can download and import the template for its specific use case.

For example, the same AAS submodel template **Carbon Footprint** can be downloaded and used by a factory which manufactures pens and another factory which manufactures tires. The template development has five basic steps:

- Step (a): Analyzing application to retrieve requirements.
- Step (b): Searching for one or some existing AAS submodels (from the IDTA hub or another source) that satisfies the requirements.
- Step (c): Designing an AAS submodel template based on the found AAS submodel or making custom designs from scratch. In the latter case, AAS template designers should study and use vocabularies from credible sources, such as from ECLASS or any widely accepted ontologies.
- Step (d): Validating the newly designed AAS submodel template.
- Step (e): Storing the AAS submodel template on a **Model Lake** (see Section 5.1).

The lower flow describes the steps to develop an AAS submodel instance proposed by IDTA [IDTA, 2024]. The actors in this flow should be called **Designers**. The flow includes the steps to (1) determining a target asset, (2) finding an AAS submodel template from the IDTA hub, (3) using a modeling tool to model an AAS submodel, (4) validating the AAS submodel model, (5) deploy the AAS submodel on an AAS server, and (6) register the AAS submodel into an AAS registry.

Step (2e) is added to connect the two flows. It allows the AAS instance designers to download and import one or several AAS submodel templates from Model Lake for their designs.

4.3 Information Model

The information model of a product is logically composed of multiple submodel instances, each specified by a submodel template. A submodel template designer should consider (1) covering all involved products, and (2) not redesigning an existing submodel template.

Regarding (1), the challenge is that while many submodel templates remain unchanged across different products (e.g. the Carbon Footprint submodel template is needed by all kinds of products and its related information should be standardized by experts for a uniform evaluation), some are product specific (e.g. cup requires a property indicating its container volume, while another product such as an e-bike doesn't need such information).

Regarding (2), the idea is that if a submodel template has existed and is widely accepted by the community, it should be wholly or partly reused. While some submodel templates are designed by several organizations (e.g. Product Nameplate), some demand newly designed from scratch.

Table 4-1 lists all AAS submodels for PDT and DPP. These submodels are essential for RAASCEMAN as presented in the analysis of Section 3. In this table, beside the three first columns—Code, Name, and Description—which are basic, the two final columns are used to classify submodel templates based

on (1) and (2). In detail, a cell in column **U** is marked (x) when a submodel template can be **universally applicable** for all kinds of products. However, it is not preventing the template user from extending it if necessary. A cell in column **R** has value when a submodel template is **reused** from an existing submodel template and is blank when it is newly created from scratch. The value is **IDTA** means that the reused submodel template is proposed by IDTA, and is **RPTU** means that it is from our consortium partners at RPTU.

Table 4-1: Submodel templates required by the applications of RAASCEMAN project

Code	Name	Description	Note	
			U	R
S1	**Product Nameplate** *(Product Identification or Digital Nameplate)*	Information describes the digital label or identification of a product. Required by: D1.1P13.1, T3.1I1, T3.1I6, T3.1I7, T3.1I9, T3.3I1, T3.3I2, T3.3I5, T4.1I1, T4.1I1, T4.2I4, T4.2I5, T4.2O1, T4.2O3, UC2.R16, UC2.R25, UC3.R31	x	RPTU
S2	**Characteristics** *(Product Characteristics)*	Information describes the characteristics, features, and behaviors of a product. Depending on the nature of the product, the information could be different. Required by: D1.1P13.1, T3.1I5, T3.3I1, T3.3I5, T4.1I1, T4.2I1, T4.2I4, T4.2I5, T4.2O1, T4.2O3, T4.3I1, UC2.R25, UC3.R31		
S2.1	Automotive Characteristics	Characteristics adhere to automotive.		
S2.2	Electric Bike Characteristics	Characteristics adhere to electric bike.		
S2.3	Battery Characteristics	Characteristics adhere to battery.		
S3	**Bill of Materials** *(BoM)*	Information about components constituting a product. Required by: D1.1P13.1, D1.2P16.1, D1.2P16.2, D1.3P46, T4.2I1, T4.2I4, UC1.R8, UC2.R4, UC2.R25, UC3.R23, UC3.R31, UC3.R32, UC3.R43	x	IDTA
S4	**Bill of Services** *(BoS)*	Information about services (manufacturing, transport, sales, etc.) required to make a product available to a client. Required by: D1.1P49.1, REQ1.2.1, REQ1.2.6, REQ3.2.1.4, D1.1P60, T3.4I1, T3.4O1, UC1.R8, UC1.R9	x	
S5	**Carbon Footprint** *(Product Carbon Footprint)*	Information about carbon footprint calculation and exchange of a product. Required by: REQ1.2.2, REQ4.1.2.4, T3.4I3	x	IDTA

S6	**Usage** *(Usage Information)*	Information describes how a product has been used during its lifetime. Required by: D1.1P13.2	x	
S7	**Traceability** *(Lifecycle Tracking)*	Information to trace a product in its lifecycle. Required by: D1.1P13.2, REQ1.2.7, D1.2P16.1, D1.2P16.4, UC1.R8, UC2.R5, UC2.R11, UC2.R17	x	
S8	**Location** *(Location Tracking)*	Information about the geographical location of a product. Required by: D1.2P16.1	x	IDTA
S9	**Quality** *(Quality Information)*	Information related to the evaluation of the product quality level. Required by: D1.2P16.2, UC2.R12, UC3.R42, T3.3O2	x	
S10	**Maintenance** *(Maintenance Information)*	Information required for the maintenance of a product. Required by: D1.2P16.3	x	
S11	**Environmental Impact** *(Social Impact)*	Information about the impact of a product on the environment. Required by: REQ4.1.2.4	x	
S12	**Health Impact** *(User Health Impact)*	Information about the impact of a product on user health. Required by: REQ4.1.2.4	x	
S13	**Commercial Information** *(Purchase-Related Information or Commercial Properties)*	Information required by commercial activities related to a product. Most of the fields are reused from the RPTU version with some additions. Required by: REQ1.2.2, T3.1I4, T3.4I3, T3.3O3	x	RPTU
S14	**Operational Instructions** *(Instructions)*	Information supporting operational activities (e.g., assembly, disassembly) Required by: UC1.R10, UC3.R44, UC3.R45		

Note that all the submodel templates presented below in this document are designed using the AAS Design Diagram (***ADD***) developed by CEA. Technically, they are ***Class diagrams*** of Unified Modeling Language (***UML***) [Muller et al., 2004] specified with the AAS profile of **Papyrus4Manufacturing**.

> Product Nameplate

Reuse the ***Product Identification*** submodel template proposed by DFKI & RPTU & SF.

> Bill of Materials

Reuse the ***Hierarchical Structures enabling Bills of Material*** proposed by IDTA.

> Bill of Services

Figure 4-3 presents the structure of the submodel Bill of Services using UML. Details of the submodel elements are as follows.

- **ServiceChain**: a reference to an entity that can describe the relation of the services. For example, the referenced entity can be a BPMN file that describes all services' relations.
- **ServiceSet**: a list of services.
- **Service**: a collection of information about a specific service. Each service has a labeled number between 0 and 99.
- Service/**PositionInServiceChain**: a reference to the relative position of the service in the service chain.
- Service/**Name**: a string to present the name of the service.
- Service/**Identifier**: a string to present the identification of the service.
- Service/**Mandatory**: a boolean to present whether the service is mandated or optional.
- Service/**ManufacturingServiceReference**: a reference to a Manufacturing Service. Note that Manufacturing Service is a new term defined in RAASCEMAN

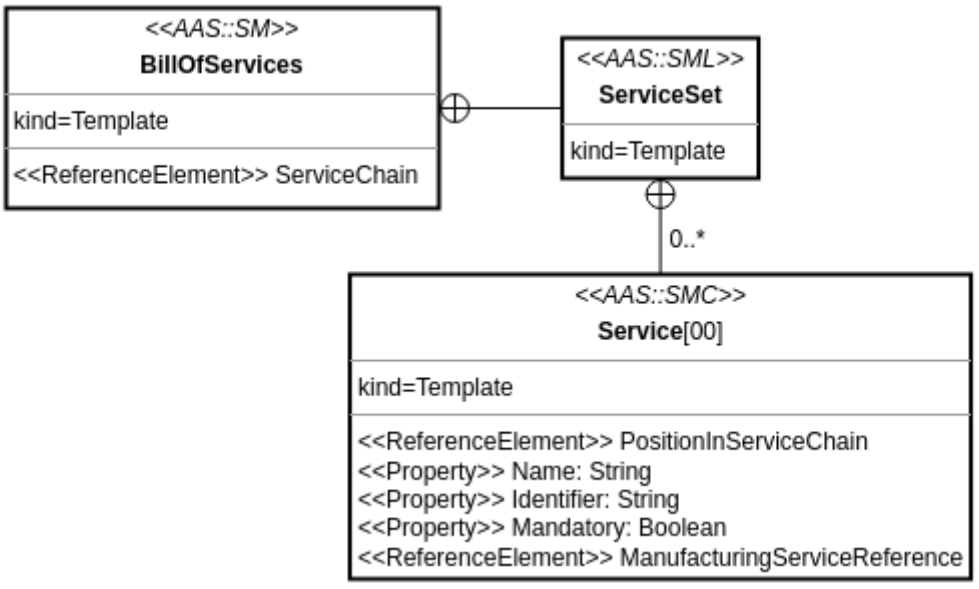


Figure 4-3: Bill of Services submodel template

> Carbon Footprint

Reuse the ***Carbon Footprint*** submodel template proposed by IDTA.

> Usage

Figure 4-4 presents the structure of submodel Usage using UML. Details of the submodel elements are as follows.

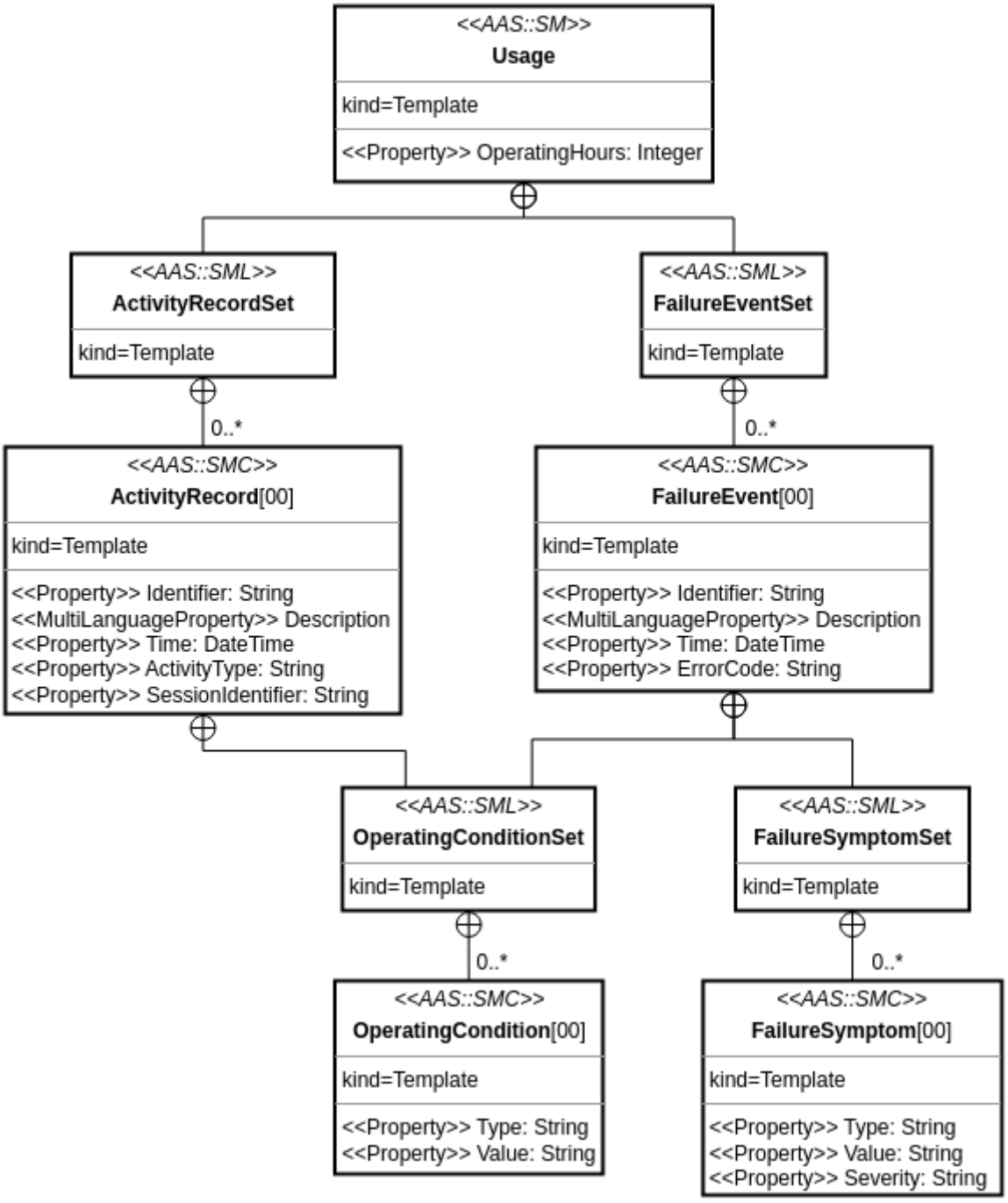


Figure 4-4: Usage submodel template

- **OperatingHours**: an integer number recording the duration which the product is used and/or in operation.
- **ActivityRecordSet**: a list of activity records during the product's lifetime.

- **ActivityRecord**: a collection of information about an activity record. Each activity record has a labeled number between 0 and 99.
- ActivityRecord**/Identifier**: a string to present the identification of the activity record.
- ActivityRecord**/Description**: a multi language string for the description of the activity record.
- ActivityRecord**/Time**: a date time string to present the time the activity is recorded.
- ActivityRecord**/ActivityType**: a string to present the activity type.
- ActivityRecord**/SessionIdentifier**: a string to present the identification of the session providing the context of the activity.
- **FailureEventSet**: a list of the failure events during the product's lifetime.
- **FailureEvent**: a collection of information about a failure event. Each failure event has a labeled number between 0 and 99.
- FailureEvent**/Identifier**: a string to present the identification of the failure event.
- FailureEvent/**Description**: a multi language string for the description of the failure event.
- FailureEvent/**Time**: a date time string to present the time the failure event is recorded.
- FailureEvent/**ErrorCode**: a string to present the error code associated to the failure event.
- **OperatingConditionSet**: a list of operating conditions related to the context of an activity or a failure event.
- **OperatingCondition**: a collection of information about an operating condition. Each operating condition has a labeled number between 0 and 99.
- OperatingCondition/**Type**: a string to represent the name of the condition.
- OperatingCondition/**Value**: a string to represent the value or information to specify the quantitative or qualitative of the operating condition.
- **FailureSymptomSet**: a list of failure symptoms related to the failure event.
- **FailureSymptom**: a collection of information about a failure symptom. Each failure symptom has a labeled number between 0 and 99.
- FailureSymptom/**Type**: a string to represent the name of the failure symptom.
- FailureSymptom/**Value**: a string to represent the value or information to specify the quantitative or qualitative of the failure symptom.
- FailureSymptom/**Severity**: a string to represent the critical level of the failure symptom.

> Traceability

Figure 4-5 and 4-6 present the structure of submodel Maintenance using UML. Details of the submodel elements are as follows.

- **CurrentPhase**: a string to present the current phase of a product.
- **MachineReadableCode**: file or reference towards the carrier (e.g. QR Code) pointing to the identifier of a product.
- **Design**: a collection of information involved in phase Design.
- Design/**CurrentStatus**: a string to present the real-time status the product design.
- Design/**DesignProvider**: a string to present the design provider (e.g. company, department).
- Design/**CurrentVersion**: a string to present the current version of the design.

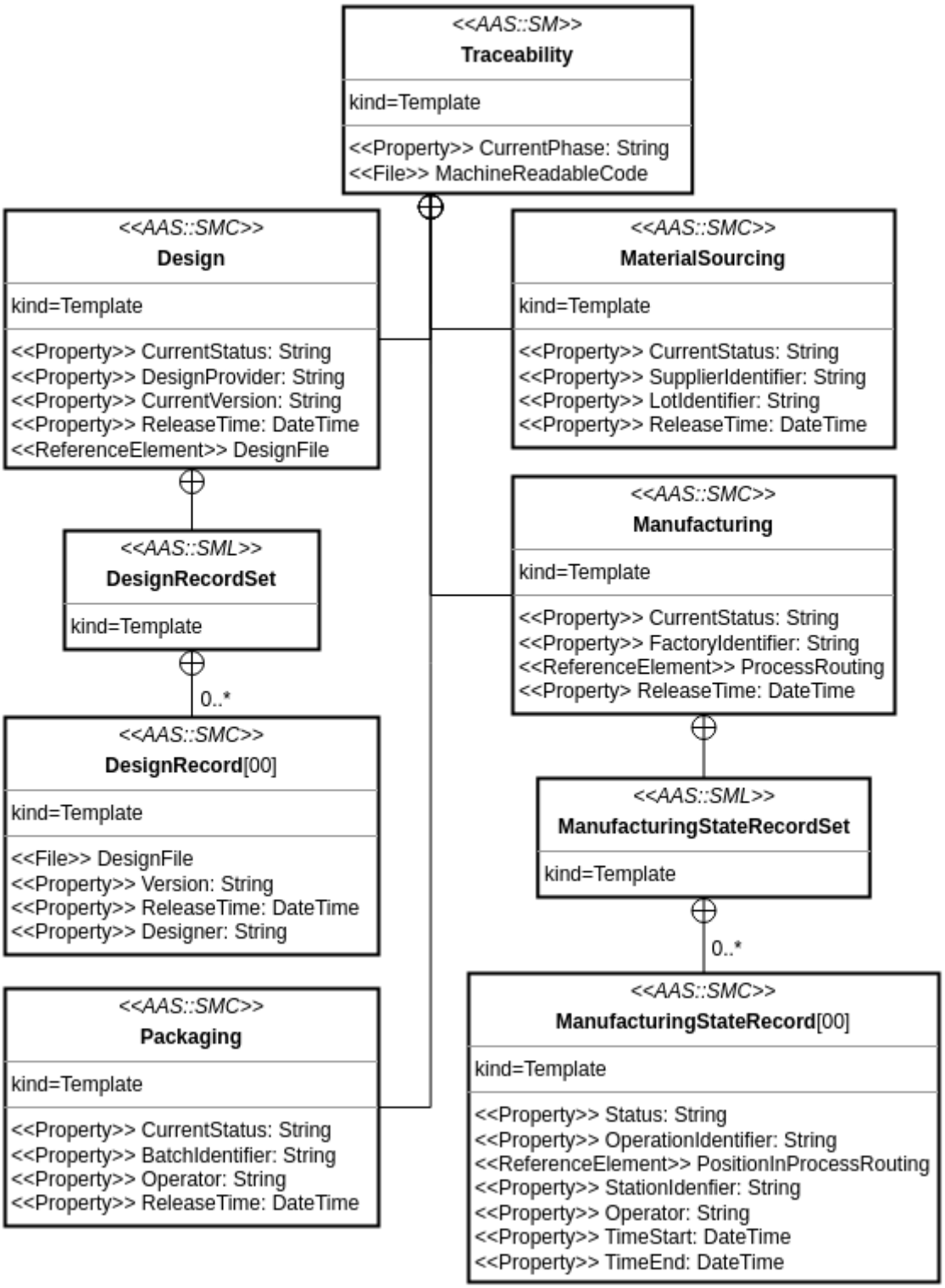


Figure 4-5: Part 1 of the Traceability submodel template

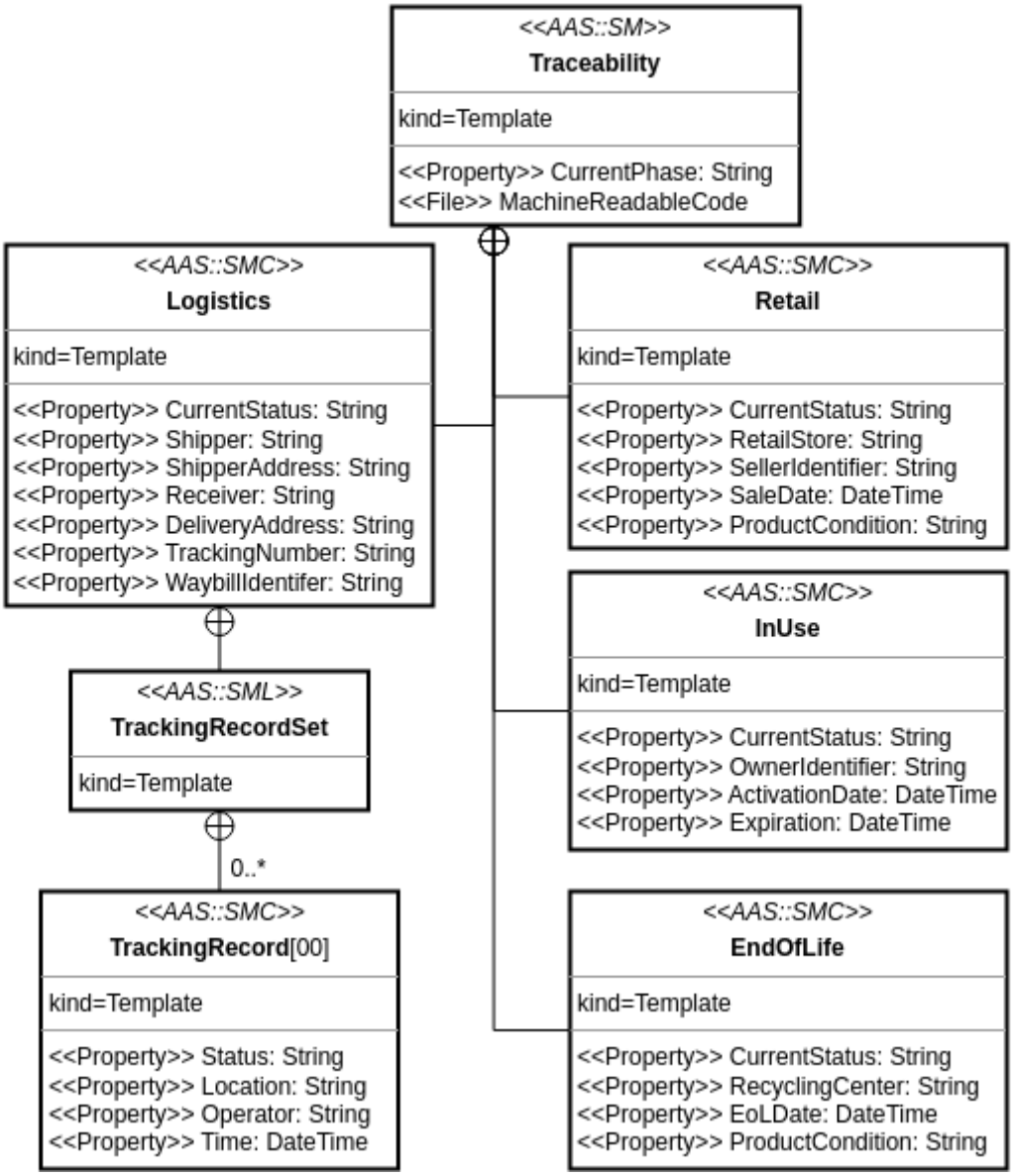


Figure 4-6: Part 2 of the Traceability submodel template

- Design/**ReleaseTime**: a date time string to present the time that the design is released
- **DesignRecordSet**: a list of design records.
- **DesignRecord**: a collection of information of a design record. Each record has a labeled number between 0 to 99.
- DesignRecord/**DesignFile**: a file or reference towards the design file of the design record.
- DesignRecord/**Version**: a string to present the version of the design record.
- DesignRecord/**ReleaseTime**: a date time string to present the time that the design version is released.
- DesignRecord/**Designer**: a string to present the name of the designer.
- **MaterialSourcing**: a collection of information involved in phase Material Sourcing.

- MaterialSourcing/**CurrentStatus**: a string to present the real-time status of material.
- MaterialSourcing/**SupplierIdentifier**: a string to present the material supplier identifier.
- MaterialSourcing/**LotIdentifier**: a string to present the lot comprising the material.
- MaterialSourcing/**ReleaseTime**: a date time string to present the time that the material and its lot is finished and/or released.
- **Manufacturing**: a collection of information involved in phase Manufacturing.
- Manufacturing/**CurrentStatus**: a string to present the real-time status of product in the process routing.
- Manufacturing/**FactoryIdentifier**: a string to present the identifier of the factory.
- Manufacturing/**ProcessRouting**: a reference to an entity (e.g. BPMN diagram) presenting the process routing.
- Manufacturing/**ReleaseTime**: a date time string to present the time the product is released from phase Manufacturing.
- **ManufacturingStateRecordSet**: a list of product state records in phase Manufacturing.
- **ManufacturingStateRecord**: a collection of information of a manufacturing state record. Each record has a labeled number between 0 to 99.
- ManufacturingStateRecord/**Status**: a string to present the status of the product in phase Manufacturing.
- ManufacturingStateRecord/**OperationIdentifier**: a string to present the identification of the the operation.
- ManufacturingStateRecord/**PositionInProcessRouting**: a reference to the relative position of the operation in the operation routing.
- ManufacturingStateRecord/**StationIdentifier**: a string to present the identification of the station that the operation occurs.
- ManufacturingStateRecord/**Operator**: a string to present the name or identifier of the operator at the station.
- ManufacturingStateRecord/**TimeStart**: a date time string to present the start time of the operation.
- ManufacturingStateRecord/**TimeEnd**: a date time string to present the end time of the operation.
- **Packaging**: a collection of information involved in phase Packaging.
- Packaging/**CurrentStatus**: a string to present the real-time status of the product in phase Packaging.
- Packaging/**BatchIdentifier**: a string to present the batch comprising the product.
- Packaging/**Operator**: a string to present the name or identifier of the packaging operator.
- Packaging/**ReleaseTime**: a date time string to present the time that the package is released.
- **Logistics**: a collection of information involved in phase Logistics.
- Logistics/**CurrentStatus**: a string to present the real-time status of the product in logistics.
- Logistics/**Shipper**: a string to present the name or identifier of the expedition.
- Logistics/**ShipperAddress**: a string to present the address of the expedition.
- Logistics/**Receiver**: a string to present the name or identifier of the destination.
- Logistics/**DeliveryAddress**: a string to present the address of the expedition.

- Logistics/**TrackingNumber**: a string to present the identification used for monitoring the delivery.
- Logistics/**WaybillIdentifier**: a string to present the identification of the waybill.
- **TrackingRecordSet**: a list of tracking records.
- **TrackingRecord**: a collection of information to present the tracking record. Each record has a labeled number between 0 to 99.
- TrackingRecord/**Status**: a string to present the status of the tracking record.
- TrackingRecord/**Location**: a string to present the location related to the tracking record.
- TrackingRecord/**Operator**: a string to present the name or identifier of the operator.
- TrackingRecord/**Time**: a date time string to present the time of the tracking record.
- **Retail**: a collection of information involved in phase Retail.
- Retail/**CurrentStatus**: a string to present the real-time status of the product in phase Retail.
- Retail/**RetailStore**: a string to present the identifier of the retail store.
- Retail/**SellerIdentifier**: a string to present the name or identifier of the seller.
- Retail/**SaleDate**: a date time string to present the time of the product is sold.
- Retail/**ProductCondition**: a string to present the condition of product (e.g. new, second hand, or refurbished).
- **InUse**: a collection of information involved in phase In-Use.
- InUse/**CurrentStatus**: a string to present the real-time status of the product in phase In-Use.
- InUse/**OwnerIdentifier**: a string to present the name or identifier of the client (owner) that owns the product.
- InUse/**ActivationDate**: a date time string to present the date that the product is activated by the client for the first time.
- InUse/**Expiration**: a date time string to present the date that the product is expired.
- **EndOfLife**: a collection of information involved in phase End-of-Life.
- EndOfLife/**CurrentStatus**: a string to present the real-time status of phase End-of-Life.
- EndOfLife/**RecyclingCenter**: a string to present the identifier of the recycling center.
- EndOfLife/**EoLDate**: a date time string to present the retirement date of the product.
- EndOfLife/**ProductCondition**: a string to present the condition of the product.

> Location

Reuse the *Data Model for Asset Location* submodel template proposed by IDTA.

> Quality

Figure 4-7 presents the structure of submodel Quality using UML. Details of the submodel elements are as follows.

- **CertificateSet**: a list of certificates.
- **Certificate**: a collection of information about a certificate. Each certificate has a labeled number between 0 and 99.
- Certificate/**CertificateIdentifier**: a string to present the identification of the certificate.
- Certificate/**CertificateCategory**: a string to present the category of the certificate (e.g. performance, safety).
- Certificate/**Description**: a multi language string for the description of the certificate.

- Certificate/**IssueDate**: a date time string to present the time the certificate is issued.
- Certificate/**ValidityPolicy**: a string to present the validity policy of the certificate.
- **StandardSet**: a list of standards.
- **Standard**: a collection of information about a standard. Each standard has a labeled number between 0 and 99.
- Standard/**StandardIdentifier**: a string to present the identification of the standard.
- Standard/**ClassificationValue**: a string to present the score of a product in the standard scale.
- Standard/**Description**: a multi language string for the description of the standard.
- Standard/**TestReport**: a reference to the test report that gives the score.
- Standard/**ValidityPolicy**: a string to present the validity policy of the standard.

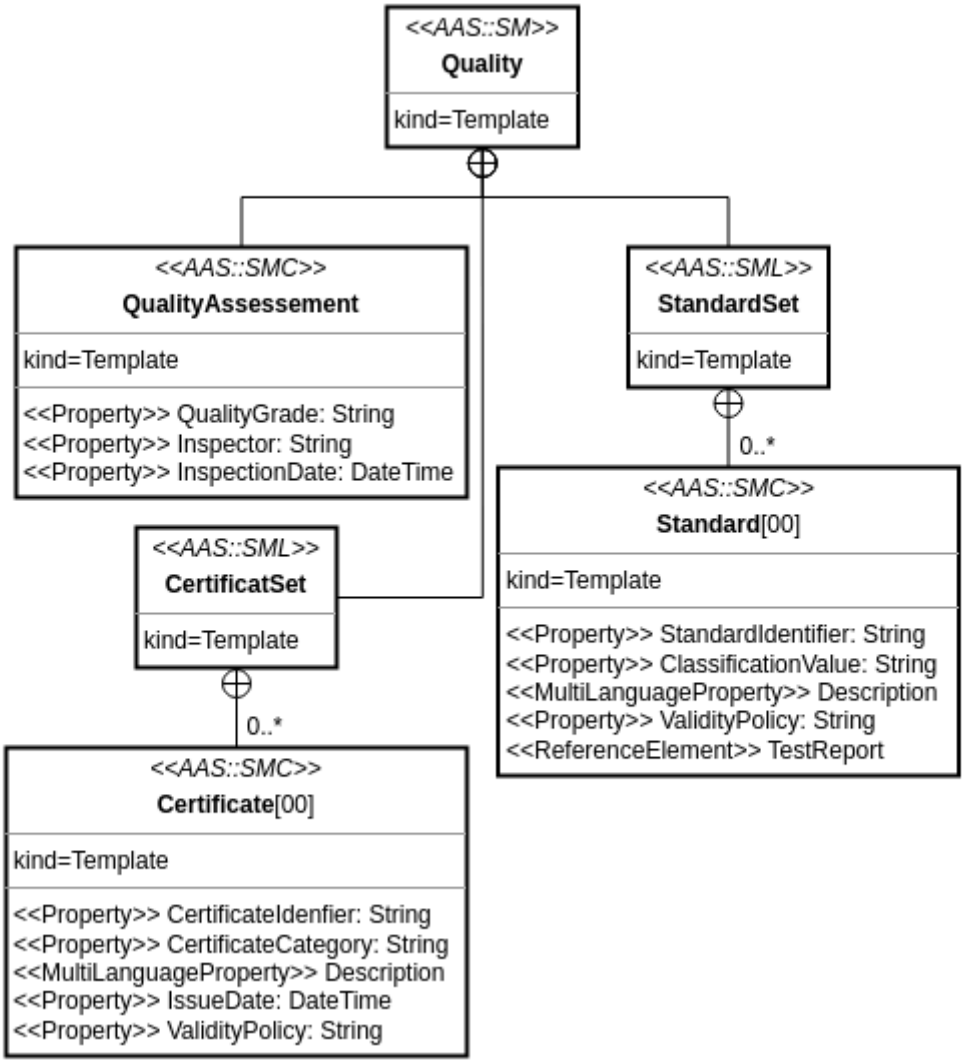


Figure 4-7: Quality submodel template

- QualityAssessement/**QualityGrade**: a string to present the grade of the quality of a product which can be A-Grade, B-Grade, or Reject.

- QualityAssessment/**Inspector**: a string to present the identifier of the person/device that executes the inspection.
- QualityAssessement/**InspectionDate**: a datetime when a product is inspected or evaluated.

> Maintenance

Figure 4-8 presents the structure of submodel Maintenance using UML. Details of the submodel elements are as follows.

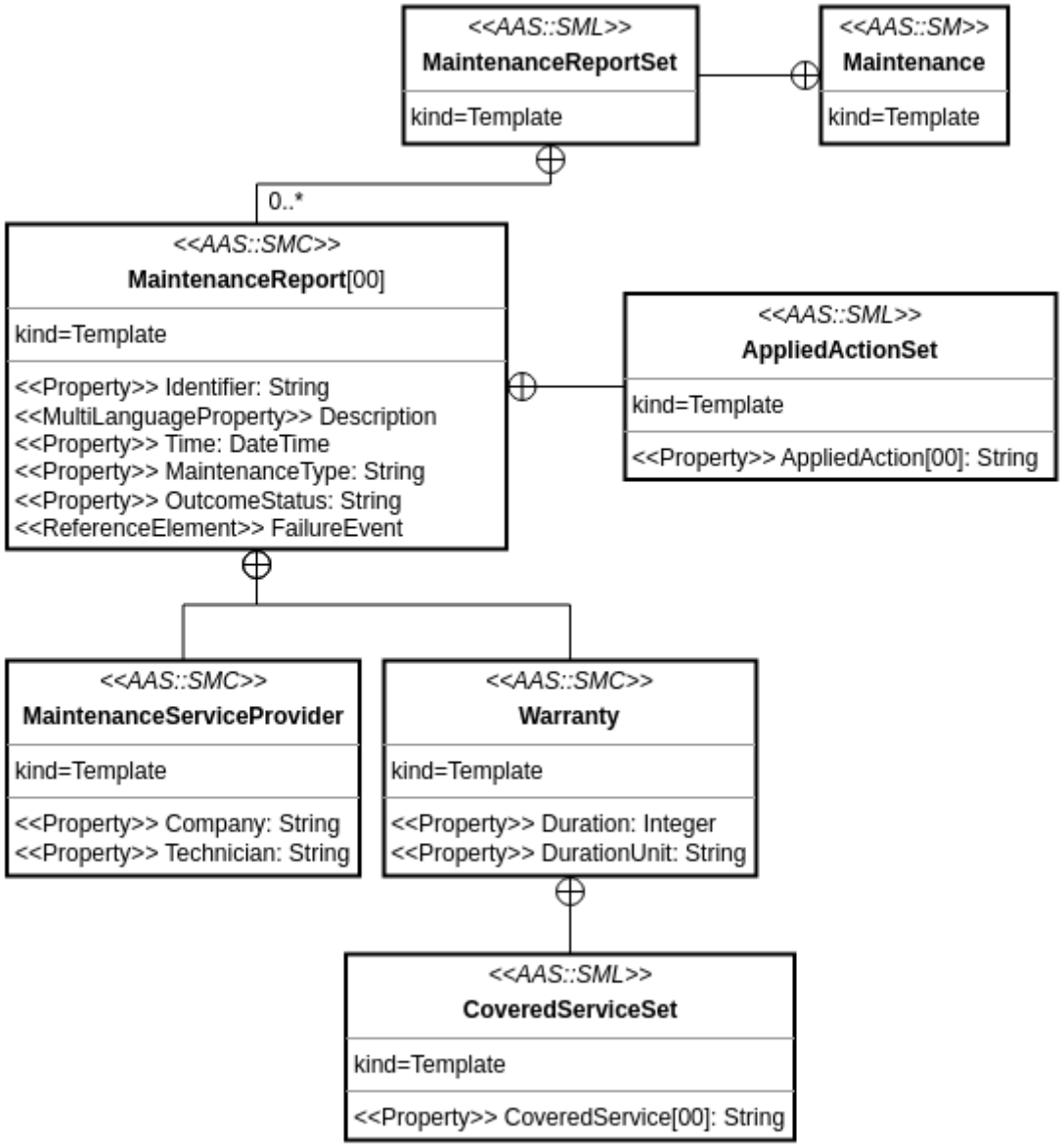


Figure 4-8: Maintenance submodel template

- **MaintenanceReportSet**: a list of maintenance reports.
- **MaintenanceReport**: a collection of information from a maintenance report. Each maintenance report has a labeled number between 0 and 99.

- MaintenanceReport/**Identifier**: a string to present the identification of the maintenance report.
- MaintenanceReport/**Description**: a multi language string for the description of the maintenance report.
- MaintenanceReport/**Time**: a date time string to present the time the maintenance occurs.
- MaintenanceReport/**MaintenanceType**: a string to present the maintenance type.
- MaintenanceReport/**OutcomeStatus**: a string to present the status after the maintenance.
- MaintenanceReport/**FailureEvent**: a reference to the failure event related to the maintenance.
- **MaintenanceServiceProvider**: a collection of information about the maintenance provider.
- MaintenanceServiceProvider/**Company**: a string to present the name of the company.
- MaintenanceServiceProvider/**Technician**: a string to present the name or identifier of technician that is in charge of the maintenance.
- **Warranty**: a collection of information about the warranty of the maintenance.
- Warranty/**Duration**: an integer number about the validity duration of the maintenance.
- Warranty/**DurationUnit**: a string to present the unit of the duration (e.g. month, quarter, year) of the maintenance.
- **CoveredServiceSet**: a list of maintenance services covered by the warranty.
- CoveredServiceSet/**CoveredService**: a string to present the name of the covered service. Each covered service has a labeled number between 0 and 99.
- **AppliedActionSet**: a list of actions applied in the maintenance.
- AppliedActionSet/**AppliedAction**: a string to present the name of the applied action. Each applied action has a labeled number between 0 and 99.

> Environmental Impact

Figure 4-9 presents the structure of submodel Environmental Impact using UML. Details of the submodel elements are as follows.

- **RegulationSet**: a list of regulations.
- **Regulation**: a collection of information about a regulation. Each regulation has a labeled number between 0 and 99.
- Regulation**/Methodology:** a string to present the identification of the methodology.
- Regulation/**Description**: a multi language string for the description of the methodology.
- Regulation/**Version** a string to present the version of the methodology.
- **IndicatorSet**: a list of indicators.
- **Indicator**: a collection of information about an indicator. Each indicator has a labeled number between 0 and 99.
- Indicator/**IndicatorIdentifier**: a string to present the identification of the indicator.
- Indicator/**Description**: a multi language string for the description of the indicator.

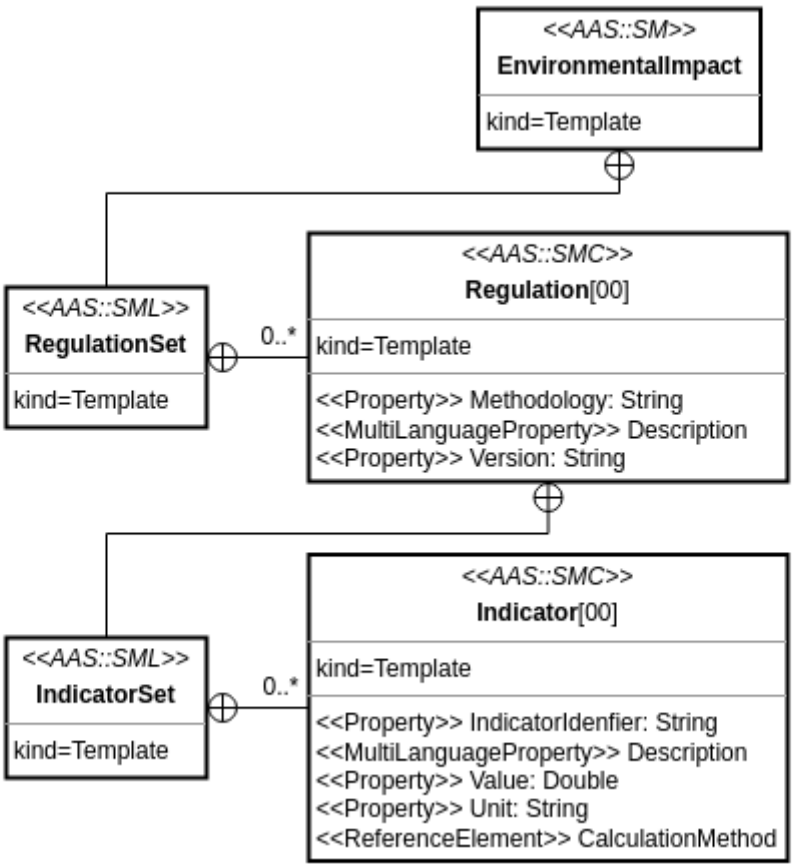


Figure 4-9: Environmental Impact submodel template

- Indicator/**Value** a double to present the score value following the indicator.
- Indicator/**Unit**: a string to present the unit of the value.
- Indicator/**CalculationMethod**: a reference to the method to calculate the score.

> Health Impact

Figure 4-10 presents the structure of the submodel Health Impact using UML. Details of the submodel elements are as follows.

- **RegulationSet**: a list of regulations.
- **Regulation**: a collection of information about a regulation. Each regulation has a labeled number between 0 and 99.
- Regulation**/Methodology:** a string to present the identification of the methodology.
- Regulation/**Description**: a multi language string for the description of the methodology.
- Regulation/**Version**: a string to present the version of the methodology.
- **IndicatorSet**: a list of indicators.

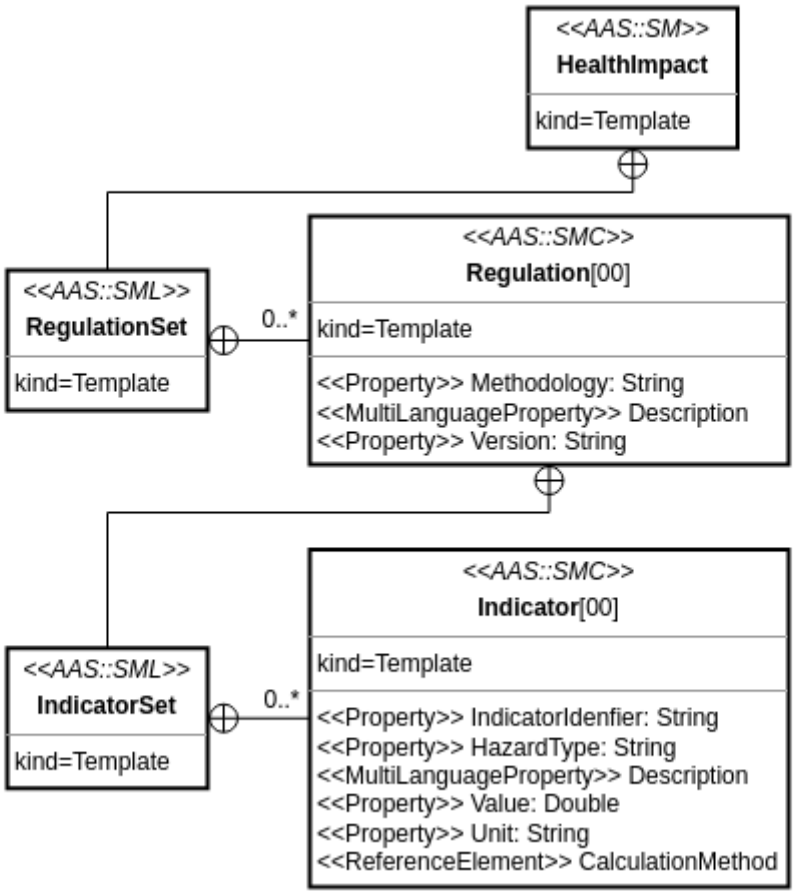


Figure 4-10: Health Impact submodel template

- **Indicator**: a collection of information about an indicator. Each indicator has a labeled number between 0 and 99.
- Indicator/**IndicatorIdentifier**: a string to present the identification of the indicator.
- Indicator/**HazardType**: a string to present the hazard type related to the indicator.
- Indicator/**Description**: a multi language string for the description of the indicator.
- Indicator/**Value** a double to present the score value following the indicator.
- Indicator/**Unit**: a string to present the unit of the value.
- Indicator/**CalculationMethod**: a reference to the method to calculate the score.

> Commercial Information

Reuse the *Commercial Properties* submodel template proposed by DFKI & RPTU & SF with some modifications. Figure 4-11 presents the structure of submodel Commercial Information using UML. Details of the submodel elements are as follows.

- **RequiredNorms**: a group of mandatory standards, certifications, or norms that the supplier or product must comply with.
- RequiredNorms**/ISO9001:** a string to present the ISO 9001 quality-management standard.

- RequiredNorms**/IATF16949:** a string to present the IATF 16949 automotive quality-management standard.
- **TenderCriteria**: a collection of commercial requirements defined for a tender, RFQ, or procurement process.
- TenderCriteria/**PreferredVenueOfProvision**: a string to present the preferred location where the goods or services should be delivered or provided.

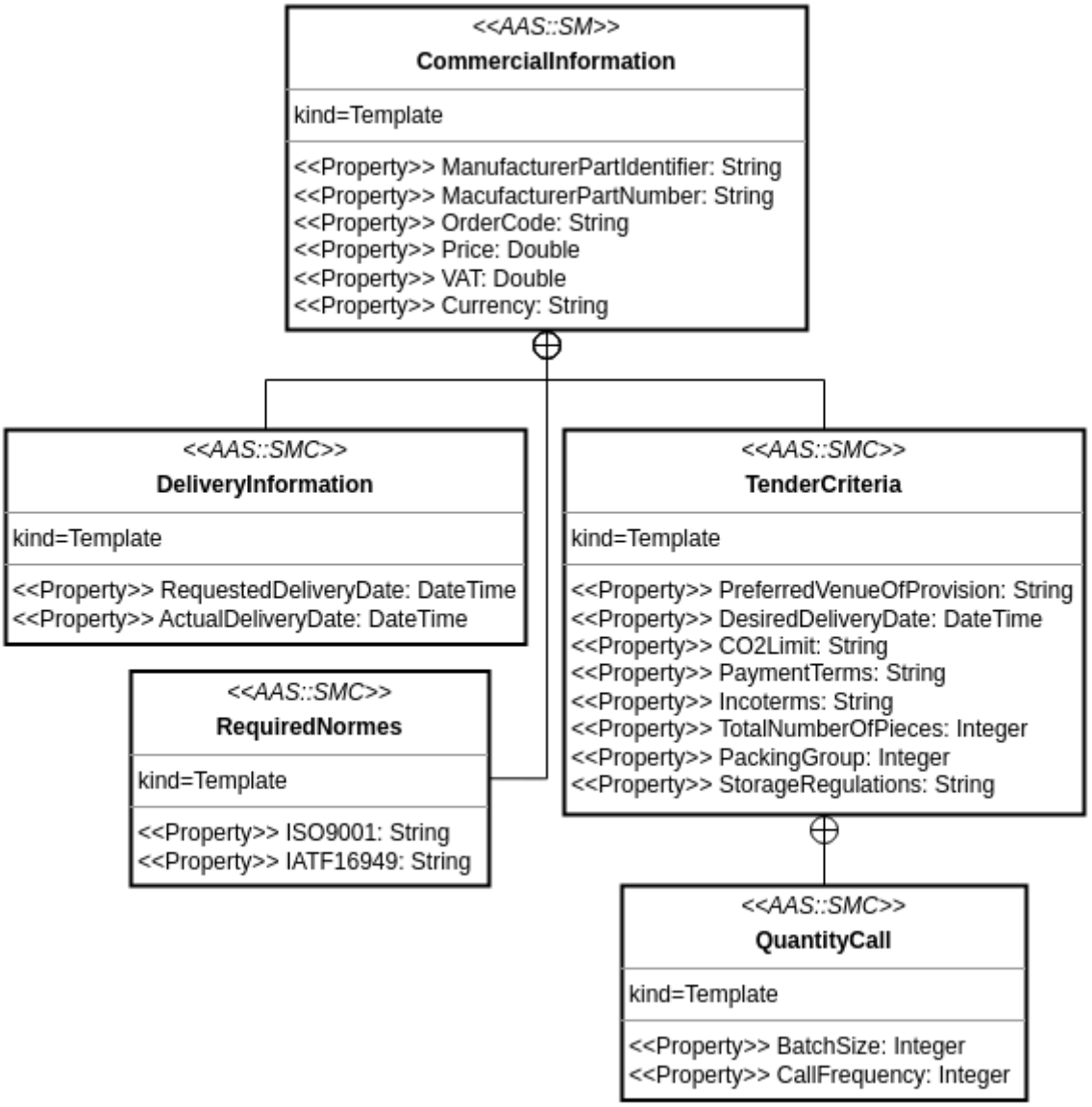


Figure 4-11: Commercial Information submodel template

- TenderCriteria/**DesiredDeliveryDate**: the target datetime by which the customer wishes to receive the goods or services.
- TenderCriteria/**CO2Limit**: a string to present the maximum allowable CO_2 emissions associated with production, transport, or delivery.

- TenderCriteria/**PaymentTerms**: a string to present the agreed payment conditions (e.g., Net 30, advance payment, milestone-based).
- TenderCriteria/**Incoterms**: a string to present the applicable Incoterms rule governing delivery responsibilities, risk transfer, and logistics obligations.
- TenderCriteria/**TotalNumberOfPieces**: an integer to present the total quantity of items requested or contracted.
- TenderCriteria/**PackingGroup**: an integer to present the packaging classification, often used for hazardous materials or regulated goods.
- TenderCriteria/**StorageRegulations**: a string to present the special storage requirements.
- TenderCriteria/**QuantityCall**: a collection describing how quantities are ordered or called off over time.
- QuantityCall/**BatchSize**: an integer to present the size of each production or delivery batch.
- QuantityCall/**CallFrequency**: an integer to present the frequency that batches are requested.
- **DeliveryInformation**: a collection of fields describing planned and actual delivery details.
- DeliveryInformation/**RequestedDeliveryDate**: a delivery datetime requested by the customer for a specific order or batch.
- DeliveryInformation/**ActualDeliveryDate**: a datetime on which the goods were actually delivered.

> Operational Instructions

Figure 4-12 presents the structure of submodel Operational Instructions using UML. Details of the submodel elements are as follows.

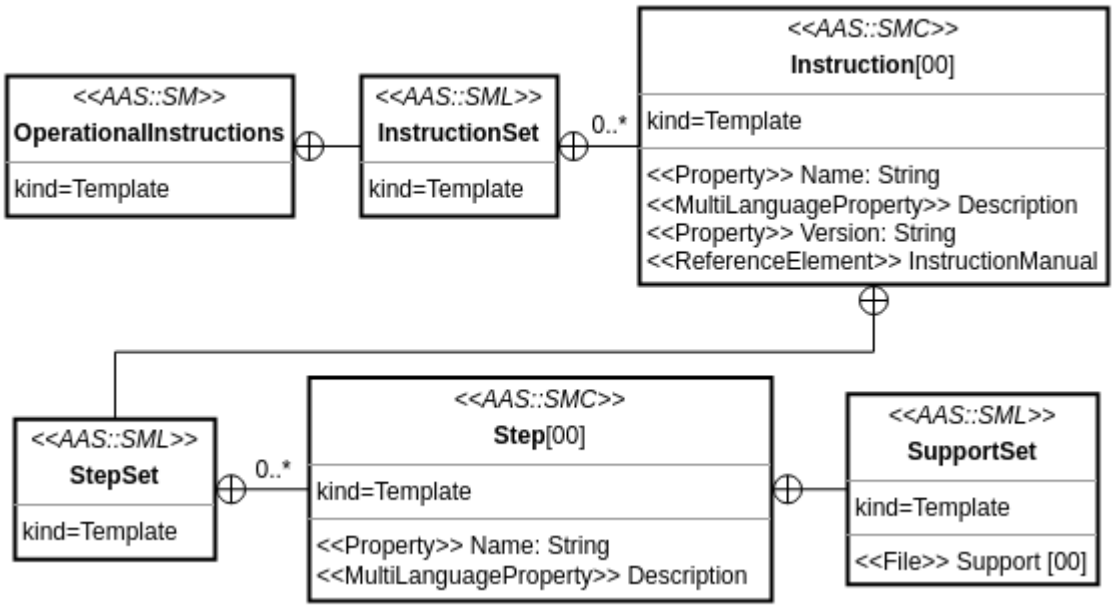


Figure 4-12: Operational Instructions submodel template

- **InstructionSet**: a list of instructions.

- **Instruction**: a collection of information about an instruction. Each instruction has a labeled number between 0 and 99.
- Instruction**/Name**: a string to present the name/title of the instruction.
- Instruction**/Description**: a multi language string for the description of the instruction.
- Instruction**/Version**: a string to present the version of the instruction.
- Instruction**/InstructionManual**: a reference to the instruction manual which can be a file or an online resource.
- **StepSet**: a list of steps constituting the instruction.
- **Step**: a collection of information about a step. Each step has a labeled number between 0 and 99.
- Step/**Name**: a string to present the name of the step.
- Step/**Description**: a multi language string for the description of the step.
- **SupportSet**: a list of items to support a step.
- SupportSet/**Support**: a link to a file that describes the step. Each support file has a labeled number between 0 and 99.

> Characteristics

Figure 4-13 presents the structure of submodel Characteristics using UML. Details of the submodel elements are as follows.

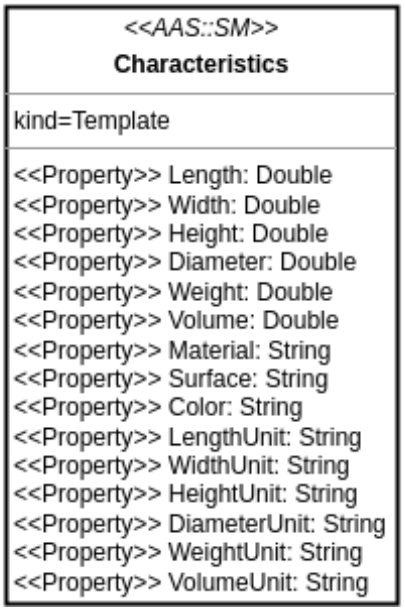


Figure 4-13: Characteristics submodel template

- **Length**: a double to present the product's length.
- **Width**: a double to present the product's width.
- **Height**: a double to present the product's height.
- **Diameter**: a double to present the product's diameter.
- **Weight**: a double to present the product's weight.
- **Volume**: a double to present the product's volume.

- **Material**: a string to present the product's material.
- **Surface**: a string to present the product's surface type.
- **Color**: a string to present the product's color.
- **LengthUnit**: a string to present the unit of the length.
- **WidthUnit**: a string to present the unit of the width.
- **HeightUnit**: a string to present the unit of the height.
- **DiameterUnit**: a string to present the unit of the diameter.
- **WeightUnit**: a string to present the unit of the weight.
- **VolumeUnit**: a string to present the unit of the volume.

> Automotive Characteristics

Figure 4-14 presents the structure of submodel Automotive Characteristics using UML. Some fields of Automotive Characteristics are already presented in Figure 4-12. Details of the submodel elements are as follows.

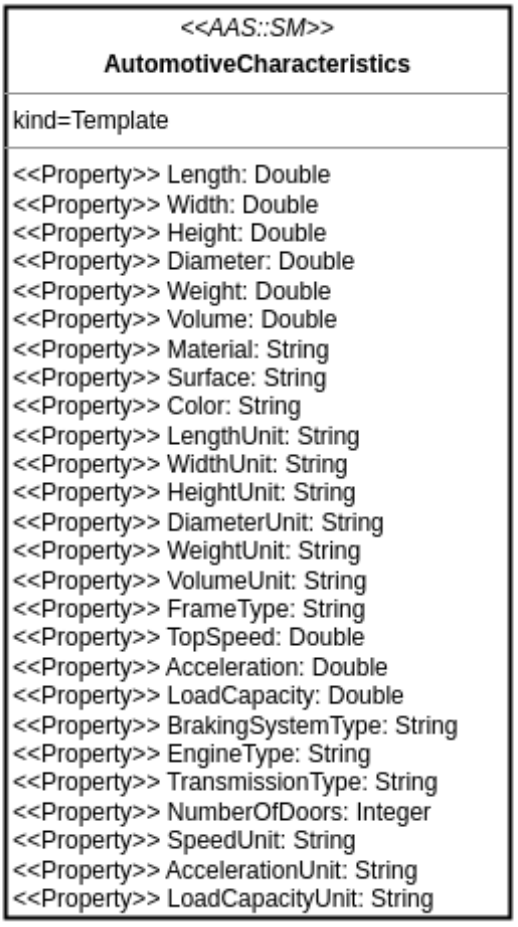


Figure 4-14: Automotive Characteristics submodel template

- **FrameType**: a string to present the type of body and frame of the automotive.
- **TopSpeed**: a double to present the maximal speed supported by the automotive.
- **Acceleration**: a double to present the maximal acceleration supported by the automotive.

- **LoadCapacity**: a double to present the load capacity supported by the automotive.
- **BrakingSystemType**: a string to present the braking system of the automotive.
- **EngineType**: a string to present the engine of the automotive.
- **TransmissionType**: a string to present the transmission system of the automotive.
- **NumberOfDoors**: an integer to present the number of doors of the automotive.
- **SpeedUnit**: a string to present the unit of speed.
- **AccelerationUnit**: a string to present the unit of acceleration.
- **LoadCapacityUnit**: a string to present the unit of load capacity.

> Electric Bike Characteristics

Figure 4-15 presents the structure of submodel Electric Bike Characteristics using UML. Some fields of Electric Bike Characteristics are already presented in Figure 4-12 and Figure 4-13. Details of the submodel elements are as follows.

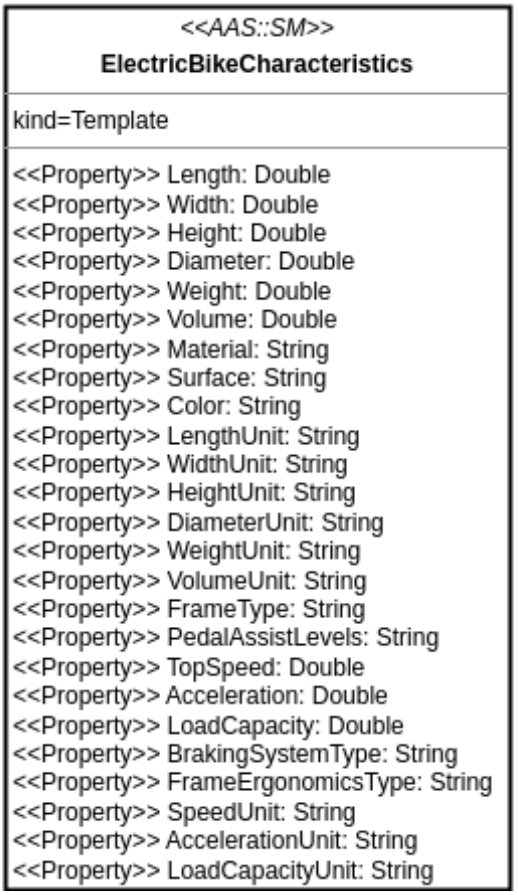


Figure 4-15: Electric Bike Characteristics submodel template

- **PedalAssistLevels**: a string to present the level supported by the pedal assistance of the electric bike.
- **FrameErgonomicsType**: a string to present the frame ergonomics of the electric bike.

> Battery Characteristics

Figure 4-16 presents the structure of submodel Battery Characteristics using UML. Some fields of Battery Characteristics are already presented in Figure 4-12. Details of the submodel elements are as follows.

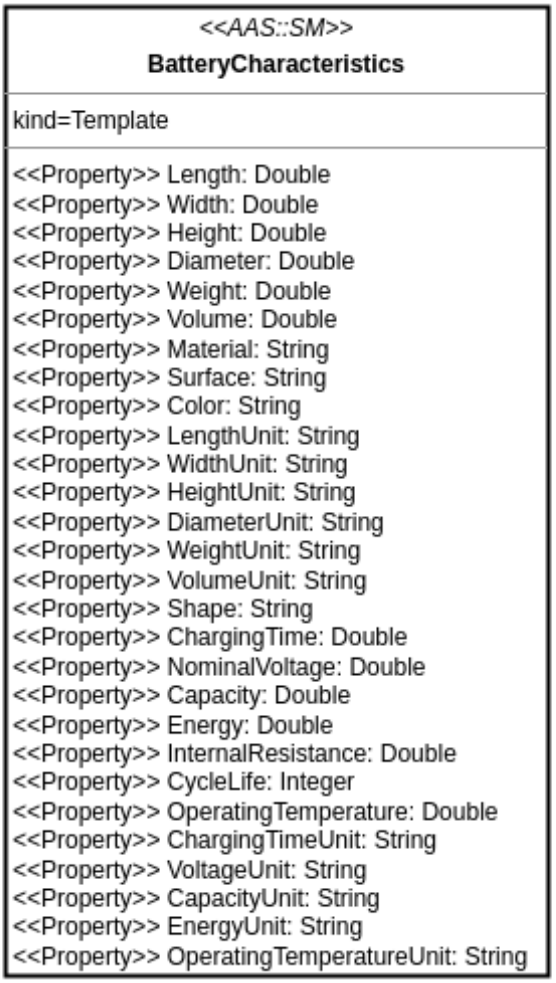


Figure 4-16: Battery Characteristics submodel template

- **Shape**: a string to present the form/shape of the battery.
- **ChargingTime**: a double to present the duration for full charge.
- **NominalVoltage**: a double to present the typical operating voltage of the battery.
- **Capacity**: a double to present the total charge storage of the battery.
- **Energy**: a double to present the energy stored in the battery.
- **InternalResistance**: a double to present the opposition to current flow inside the battery.
- **CycleLife**: an integer to present the number of charge-discharge cycles before capacity drops below threshold.
- **OperatingTemperature**: a double to present the typical operating temperature of the battery.
- **ChargingTime**: a string to present the unit of charging time.
- **VoltageUnit**: a string to present the unit of the voltage.

- **CapacityUnit**: a string to present the unit of the capacity.
- **EnergyUnit**: a string to present the unit of the energy.
- **OperatingTemperatureUnit**: a string to present the unit of the temperature.

Adopting the design method presented in Section 4.2 and using the PDT templates from this section, PDT instance designers can create a custom PDT information model for product instances. Figure 4-17 illustrates an example of a PDT instance of an electric bike.

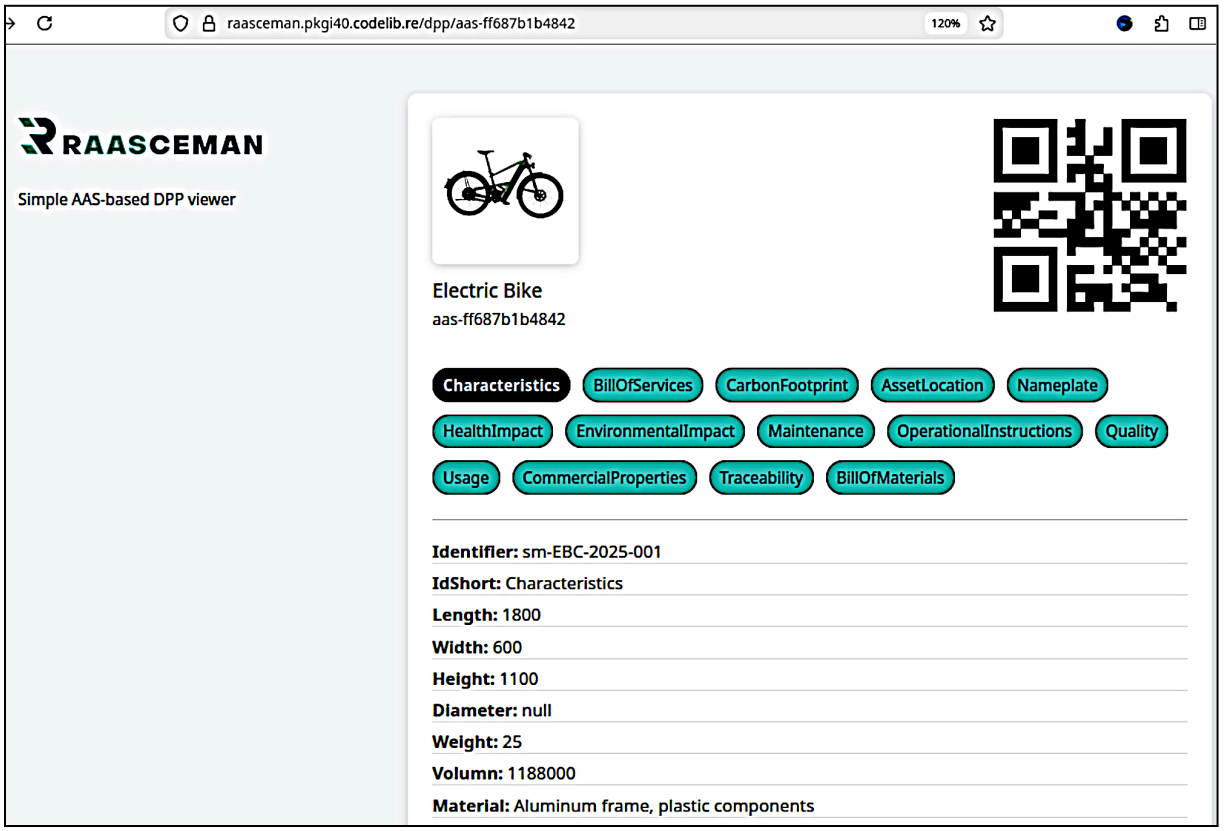


Figure 4-17: PDT information model of an electric bike instance

5 Proposed Tooling System

The tools presented in this section are particularly designed and implemented to support the development and deployment of PDTs. They have different concerns and objectives from the tools in T2.3, T2.4, as well as those in WP3 and WP4.

The first tool is Model Lake that manages PDT submodel templates and provides features to facilitate other stakeholders using the templates. The second is the Large Language Model (LLM)-powered module to support stakeholders to ask questions and get answers related to the PDT submodel designs. The third is a new version of Papyrus4Manufacturing that enables stakeholders to develop PDT submodel templates and instances compatible with AAS specification version 3 (AAS v3) using Model-Based System Engineering (MBSE).

5.1 Model Lake

Model Lake is basically a server that can store and manage models. In the RAASCEMAN context, the model lake is used to manage AAS submodel templates for PDT. A submodel template can have two different serializations: (1) the JSON format is officially supported and developed by IDTA, and (2) the UML/XMI format is familiar to the MBSE community and supported by Papyrus's ecosystem tools. In this sense, RAASCEMAN provides more than one solution for stakeholders and industrial partners to use PDTs and DPPs.

The model lake of RAASCEMAN has three major features:

- ***Cloud-based repository*** that allows users to access, download, upload, and delete PDTs through REST API. More information about the API is available in Appendix A. While the basic users (e.g. AAS instance designers) only have the right to view and download AAS templates from Model Lake, the advanced users (e.g. AAS template designers) can also upload and delete PDT templates.
- ***Converter*** is considered one core feature of the model lake. This aims to support not only the conversion between the JSON format of IDTA and UML/XMI format of Papyrus, but also the conversion between AAS v2 and AAS v3. In this sense, the model lake can support multiple types of systems, improving the interoperability of the RAASCEMAN partners' network. For example, a stakeholder may design AAS submodel templates in the UML/XMI format using Papyrus4Manufacturing. Other stakeholders' systems compatible only with AAS JSON format can download the submodel templates from Model Lake to work.
- ***Assistant*** is another advantage of the RAASCEMAN model lake especially for AAS instance designers. This feature can analyze questions in the form of Natural Language (NL) input and provide intelligent answers to PDT-related questions in the frame of RAASCEMAN context. An intelligent answer is an answer that helps the RAASCEMAN users to save time from complex analysis or studies to answer a question related to RAASCEMAN.

Figure 5.1 illustrates the basic principles of the three features.

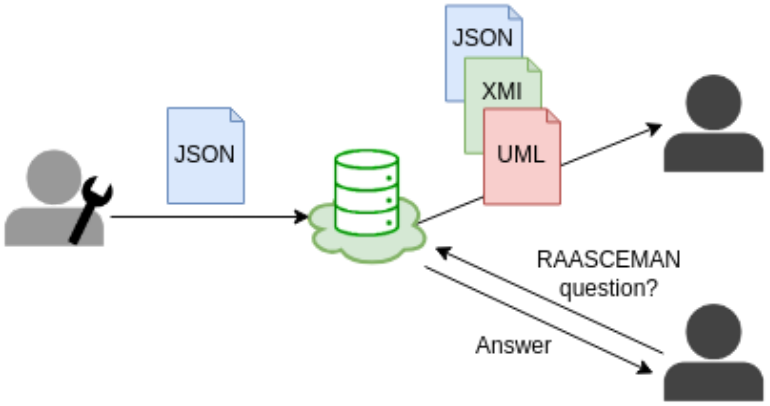


Figure 5-1: Model Lake features: (1) submodel templates management, (2) format converter, and (3) assistant

5.2 LLM-Powered Assistant for PDT Development

The LLM-Powered Assistant is a module introduced together with the Model Lake, but it can work independently. In essence, it is an expert system composed of (1) the fact base, including AAS-based PDT instances represented in YAML format, and (2) the rule base, including rules extracted from the requirements of stakeholders. The idea is to profit from the natural language analysis feature of LLM to replace the knowledge engineer role in the expert system. Note that in traditional expert systems, knowledge engineering is an important role as it must (1) collect information from domain experts, (2) translate facts and rules into machine-understandable language, and (3) implement and maintain the expert system. Using LLM can reduce the cost of having knowledge engineers.

The deployment of the assistant includes three steps:

- Transforming all AAS submodel templates presented in Section 4.3 into YAML format: the Model Lake converter supports also the YAML format; thus, it is possible to download all AAS submodel templates in the YAML format.
- Transforming all requirements into human-understandable rules: the requirements in Section 3 are manually translated IF-THEN rules. New rules can be added if necessary.
- Organizing all facts and rules in the prompt template together with instructions and user query: the assistant processes the zero-shot prompt engineering technique, in which, the LLM engine can be an external LLM service, such as Google Gemini, or an internal LLM.

In the scope of the WP2, the LLM-powered assistant is tested with the Google Gemini 2.5 model (free-tier). It can basically answer four groups of requests: (1) to provide a list of submodel templates that contain a specific property (e.g., *"give submodels that have speed property"*), (2) to provide a list of submodel templates associated with a specific product (e.g., *"give submodels for a car product"*), (3) to provide a list of submodel templates required by a RAASCEMAN application or service (e.g.,

"give submodels required by the decision support tool"), and (4) to suggest properties for products which are not predefined in the project (e.g., *"create properties for a pen product"*).

5.3 Papyrus4Manufacturing for PDT Template Design

Papyrus4Manufacturing Web (P4MW) is an extension of the **Papyrus Web** UML modeler (https://gitlab.eclipse.org/eclipse/papyrus/org.eclipse.papyrus-web) that enables modeling and designing of the AAS v3 (standard version 3) compliant DTs. Note that there is another version of Papyrus4Manufacturing Desktop (classic version) that supports the AAS v2 design. Behind both Papyrus4Manufacturing versions (Web and Desktop) are two UML profiles aligned with the meta-model defined in the AAS standard.

By moving the design into a browser environment, P4MW allows a user development team regardless of their geographical location to collaboratively design and manage DT models in real time. Multiple users can work on the same models. It enables faster iteration, improved coordination, and better knowledge sharing among engineering teams.

The web nature of P4MW eliminates the need for local desktop installations and complex software setups. This significantly reduces the barrier to entry for stakeholders such as system engineers, software developers, and domain experts. As a result, the platform facilitates seamless collaboration, efficient model sharing, and streamlined DT development workflows, making it particularly suitable for distributed industrial engineering environments. The development methodology of P4MW includes four simple steps:

- **Access the Platform**: Open P4MW in a standard web browser. No installation is required. Users can run the docker file and log in to P4MW to access team projects and existing models.
- **Create or import models**: User can either: (1) Start a new AAS-compliant DT model from scratch, (2) Import existing UML models from Papyrus Web or other compatible sources.
- **Validation and Compliance**: P4MW includes tools to validate models against the AAS version 3 standard, ensuring compliance and correctness before deployment.
- **Export and Integration**: Export models in standard UML/XMI formats integration.

The detailed installation procedure and platform access are described in Appendix B.

6 Conclusion

Deliverable 2.2 presents in detail the contributions of Task 2.2 within Work Package (WP) 2. These contributions not only fulfill the predefined objectives, including (1) the definitions of product digital twins (PDTs), and (2) the state of the art of deploying AAS-based PDTs, but also provide several additional results, which are as follows: (a) a new architecture to deploy PDT templates and PDT instances, (b) the concept and implementation of a Model Lake, (c) an LLM-powered assistant to support development of AAS-based PDT, and (c) the Papyrus4Manufacturing Web tool for PDT designer. These outcomes are based on the information and requirements studied from other work packages and tasks and are specifically dedicated to the RAASCEMAN project. Moreover, these outcomes have a strong potential to be applied in other projects beyond RAASCEMAN and in real industrial environments.

This work has some limitations. First, as Task 2.2 precedes WP5 (the phase involving deployment of real use cases), we as a consortium were not able to support the design of AAS information models for actual products (e.g. car models, e-bike models, etc.) during this phase. Therefore, the AAS submodel templates proposed in this deliverable should only be considered as references for the real implementations in WP5. Second, the Proof-of-Concept / tooling system has been implemented based on a set of basic requirements and then is tested with a small number of samples, thus, its Technology Readiness Level (TRL) remains relatively low. Third, although the ECLASS vocabulary has been identified as part of PDT design within the RAASCEMAN project, its integration will be explored in the real use cases and is outside the scope of the present work.

Funded by
the European Union

Scientific Works

2026 (Paper accepted as of April 2026)

- [Nguyen et al., 2026] Q. -D. Nguyen, K. Suri and G. D. S. Sidibe, "Towards an LLM-powered expert system for AAS-based product digital twin development", *2026 The 2nd International Workshop on Software Architecture and Generative AI (SAGAI) @ ICSA 2026*.

- [Suri et al., 2026] K. Suri and F. Arnez, "Digital Twins and World Models: A systematic taxonomic disambiguation," *2026 The 4th Workshop on the Modelling and Implementation of Digital Twins for Complex Systems (MIDas4CS) @ CAiSE 2026*

2025 (Paper Published)

- [Nguyen et al., 2025] Q. -D. Nguyen, K. Suri and G. D. S. Sidibe, "Towards Engineering Product Digital Twins for Industry 5.0: Definition and Modeling Approach," *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Porto, Portugal, 2025, pp. 1-4.
    - See Poster in Appendix C

Glossary

Asset: refers to a "*physical, digital, or intangible entity that has value to an individual, an organization, or a government*" [IDTA, 2023]

Asset Administration Shell (AAS): refers to **"***standardized digital representation of an asset*" [IDTA, 2023]

Bill of Materials (BOM): refers to a list of components or parts required to assemble a product.

Bill of Processes (BOP): refers to a list of manufacturing steps or operations required to manufacture a product.

Bill of Services (BOS): refers to a list of services required by one or several service providers to make a product available to a client.

Capabilities-Skills-Service Model (CSS): extends the Product-Process-Resource representation paradigm (PPR) with the concept of Capabilities, Skills, and Services to improve the flexibility in production and supply chain [Plattform Industrie 4.0]

Customer Order: is "*an order from a customer for a particular product or number of products*" and "*is often referred to as an actual demand to distinguish it from a forecasted demand*" [ASCM, 2025]

Digital Product Passport: is *"the collection of all product information relevant for circularity and authorities*" [DDCC, 2024]

ECLASS: is a standardized and ISO/IEC-compliant system and vocabulary for classifying products and services in industry and manufacturing.

Model Lake: is a repository responsible for storing different types of models and is integrated with services that support their management.

Product: refers to an entity which can be a prototype (blueprint or specification) or an instance (specific and realized unit)

Product Digital Twin (PDT): is "*the collection of all information of a product including the ones of the DPP*" [DDCC, 2024]

Product Digital Twin Instance: is an information model representing a PDT for a specific unit of product.

Product Digital Twin Prototype: is an information model representing the information shared by a group of PDTs.

Product Digital Twin Template: an information model representing the structure and semantics of a group of PDTs and is used to specify PDT instances.

Product instance: is a specific unit of product prototype and is associated with a unique identity.

Product prototype: refers to "*a final or intermediate product type*" with "*the difference between a product type and the realized work piece at runtime*" [Pfrommer et al., 2013]

Product-Process-Resource Paradigm: is a well-known conceptual system to categorize core elements in a manufacturing infrastructure [Pfrommer et al., 2013]

Production Order: is "*a document, group of documents, or schedule conveying authority of the manufacture of specified parts or products in specified quantities*" [ASCM, 2025]

Quotation: is "*a statement of price, terms of sale, and a description of goods or services offered by a supplier to a prospective purchaser. This can also be known as a bid. When given in response to an inquiry, it is usually considered as offer to sell*" [ASMC, 2025]

Submodel: defines "*a specific aspect of the asset represented by the AAS*" and "*is used to structure and modularize the information and is considered a logical container of semantically related submodel elements*" [IDTA, 2023].

Submodel template: is an AAS submodel representing the structure of an aspect of an asset without specific information dedicated to a specific product unit.

Submodel instance: is an AAS submodel containing information about a specific aspect associated with a specific product unit or prototype.

References

[AIF, 2023] F. Sanchez. (2023). “Digital Twin: Leveraging the Digital Transformation of the Industry”. Alliance Industrie du Future, France, Brochure.

[Admin-shell-io, 2025] Admin-shell-io by Industrial Digital Twin Association. (2025). “Aasx-package-explorer”. [Online]. Available: https://github.com/admin-shell-io/aasx-package-explorer

[ASCM, 2025] Association for Supply Chain Management (2025). “ASCM Supply Chain Dictionary, 18th Edition”. [Online]. Available: https://www.ascm.org/learning-development/certifications-credentials/dictionary/

[CEN, 2024] European Committee for Standardization. (2024). Trusted Data Transaction (CEN Workshop Agreement CWA18125; p. 11). CEN-CENELEC Management Centre.

[CIRPASS, 2024] C. Bernier and F. Danash. (2024). “DPP Prototypes”. CIRPASS Consortium, European Union, Report.

[DDCC, 2024] C. A. Ku. (2024). “Implementing the EU Digital Product Passport with Digital Data Chain and Asset Administration Shell Overview”. Digital Data Chain Consortium, VCI – Webinar on DPP Standardization.

[Dhouib, 2023] S. Dhouib, Y. Huang, A. Smaoui, T. Bhanja and V. Gezer. (2023). “Papyrus4Manufacturing: A Model-Based Systems Engineering approach to AAS Digital Twins”. 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), 1-8

[DTC, 2020] Digital Twin Consortium. (2020). “The definition of a digital twin”. [Online]. Available: https://www.digitaltwinconsortium.org/hot-topics/the-definition-of-a-digital-twin.htm

[Ehrlinger et al., 2016] L. Ehrlinger and W. Wöß. (2016). “Towards a definition of knowledge graphs”. In *Proceedings of the SEMANTiCS 2016 Conference* (pp. 1–4). CEUR-WS

[Grieves, 2023] M. W. Grieves. (2023). “Digital Twins: Past, Present, and Future,” in The Digital Twin. Cham: Springer International Publishing, pp. 97–121.

[Gozalez-Gerpe et al., 2024] S. González-Gerpe, M. Poveda-Villalón, and R. García-Castro. (2024). DTAG: A Methodology for Aggregating Digital Twins Using the WoTDT Ontology. Applied Sciences, 14(13), 5960.

[IDTA, 2023] Industrial Digital Twin Association. (2023). Specification of Asset Administration Shell—Part 1: Metamodel Schema (Specification IDTA Number: 01001-3-0; p. 215). Industrial Digital Twin Association.

[IDTA, 2024] Industrial Digital Twin Association. (2024). “Asset Administration Shell Quick Start Guide”. Industrial Digital Twin Association.

[ISO/ICE, 2023] International Electrotechnical Commission. (2023). “Digital twin - Concepts and terminology,” International Organization for Standardization, International Standard ISO/IEC 30173:2023-11.

[Kaebisch et al., 2023] S. Kaebisch, M. McCool, and E. Korkan. (2023). “Web of Things (WoT) Thing Description 1.1”. [Online]. Available: https://www.w3.org/TR/wot-thing-description11/

[Muller et al., 2004] Muller, P.-A., & Gaertner, N. (2004). Modélisation objet avec UML. Eyrolles.

[Musen et al., 2015] M. A. Musen, B. Middleton, and R. A. Greenes. (2015). “The Protégé project: A look back and a look forward”. AI Magazine, 36(4), 4–12.

[Nguyen et al., 2024] Q.-D. Nguyen, Y. Huang, F. Keith, C. Leroy, M.-T. Thi, and S. Dhouib. (2024). "Manufacturing 4.0: Checking the Feasibility of a Work Cell Using Asset Administration Shell and Physics-Based Three-Dimensional Digital Twins". Machines, 12(2), 95.

[Pfrommer et al., 2013] J. Pfrommer, M. Schleipen and J. Beyerer. (2013). "PPRS: Production skills and their relation to product, process, and resource". 2013 IEEE 18th Conference on Emerging Technologies and Factory Automation (ETFA), 1–4.

[Platfform Industrie 4.0, 2022] Plattform Industrie 4.0. (2022). *Information Model for Capabilities, Skills & Services* (p. 46) [Discussion Paper]. Plattform Industrie 4.0.

[Psarommatis et al., 2024] F. Psarommatis and G. May. (2024). "Digital Product Passport: A Pathway to Circularity and Sustainability in Modern Manufacturing," Sustainability, vol. 16, no. 1, p. 396.

[RAASCEMAN, 2024] RAASCEMAN. (2024). "Project proposal – Technical description (Part B)," RAASCEMAN project.

[RAASCEMAN, 2025] RAASCEMAN. (2025). "Deliverable 1.1 - Requirements and Specifications," RAASCEMAN project.

[RAASCEMAN, 2025a] RAASCEMAN. (2025). "Deliverable 1.2 - Demonstrator Descriptions and Evaluation Method," RAASCEMAN project.

[RAASCEMAN, 2025b] RAASCEMAN. (2025). "Deliverable 1.3 - Software and information architecture," RAASCEMAN project.

[Stellato et al., 2020] A. Stellato, M. Fiorelli, A. Turbati, T. Lorenzetti, W. van Gemert, D. Dechandon, C. Laaboudi-Spoiden, A. Gerencsér, A. Waniart, E. Costetchi, and J. Keizer (2020). "VocBench 3: A collaborative Semantic Web editor for ontologies, thesauri and lexicons". Semantic Web – Interoperability, Usability, Applicability, 11(5), 855–881.

[Studer et al., 1998] R. Studer, V. R. Benjamins and D. Fensel. (1998). Knowledge engineering: principles and methods. Data & Knowledge Engineering, 25(1–2), 161–197.

[UNTP, 2025] United Nations Transparency Protocol. (2025). "Digital Product Passport Version 0.6.0". [Online]. Available: https://uncefact.github.io/spec-untp/docs/specification/DigitalProductPassport

[WebThingsIO, 2025] Web Things IO. (2025). "WoT Capability Schemas". [Online]. Available: https://webthings.io/schemas/

Funded by the European Union

Appendix A: Model Lake REST API

Model Lake Application Protocol Interface (API) is a HTTP REST API collection designed for the model lake solution in RAASCEMAN. The APIs are deployed and tested in the server proposed by CEA.

1- Get List of Submodel templates:

- Query type: GET
- Path: /api/v1/submodels

Example:

```
https://your.endpoint/api/v1/submodels
```

- Request BODY: <EMPTY>
- Response BODY:

Field	Data Type	Value
<EMPTY>	JSON Array	[<SUBMODEL_1>, ... <SUBMODEL_n>]

<AAS_SUBMODEL>

Field	Data Type	Value
filename	String	<STRING_FILE_NAME>
content	JSON Object	<JSON_AAS_SUBMODEL>

Example:

```
[
  {
     "idShort": "MRP_ERP",
     "id": "https://example.com/ids/sm/4003_7022_7052_5508",
     "kind": "Template",
     "submodelElements": [
          {
               "idShort": "MonitoringData",
               "modelType": "SubmodelElementCollection"
          }
     ],
     "modelType": "Submodel"
   }
]
```

2- Download a Submodel template:

- Query type: GET

- Path: /api/v1/file/<TYPE>/<FILE_NAME>.<TYPE>
  - <TYPE>: file type which can be json, xmi, and uml
  - <FILE_NAME>: name of file

Example:

https://your.endpoint/api/v1/file/json/RAASCEMAN.MRP_ERP.json

- Request BODY: <EMPTY>
- Response BODY:
  - Success: a file with name <FILE_NAME>
  - Failure: {"detail": "File not found"}

3- Upload a Submodel template:

- Query type: POST
- Path: /api/v1/upload

Example:

https://your.endpoint/api/v1/upload

- Request HEADER:
  - accept: application
  - Modelake-Key: raasceman
- Request BODY: a file with format <ORGANIZATION>.<FILE_NAME>.<TYPE>
  - <ORGANIZATION>: organization that proposes the file (e.g. IDTA, DFKI, and CEA)
  - <FILE_NAME>: name of file
  - <TYPE>: file type which can be json, xmi, and uml
- Response BODY:
  - Success: {"message": "<ORGANIZATION>.<FILE_NAME>.<TYPE> is uploaded"}

4- Delete a Submodel template:

- Query type: DELETE
- Path: /api/v1/delete/<ORGANIZATION>.<FILE_NAME>.<TYPE>

Example:

https://your.endpoint/api/v1/delete/RAASCEMAN.MRP_ERP.json

- Request HEADER:
  - Modelake-Key: raasceman
- Request BODY: <EMPTY>
- Response BODY:
  - Success: {"message": "<ORGANIZATION>.<FILE_NAME>.<TYPE> is deleted"}

5- Ask a question:

- Query type: POST
- Path: /api/v1/ask

Example:

https://your.endpoint/api/v1/ask

- Request BODY: content of the question

Example:

Give me a list that decision support tool needs

- Response BODY: content of the answer

Example:

nameplate bill of materials bill of services

Figure A.1 illustrates the simple frontend proposed by CEA as an example to interact with Model Lake. The frontend GUI includes (1) a sidebar to list all of the submodel templates that have been validated by the RAASCEMAN expert team, (2) a main view to present basic submodel elements and buttons for downloading template in the different formats, and (3) a chat box enabling users to ask questions related to the templates and to the applications or use cases related to RAASCEMAN project.

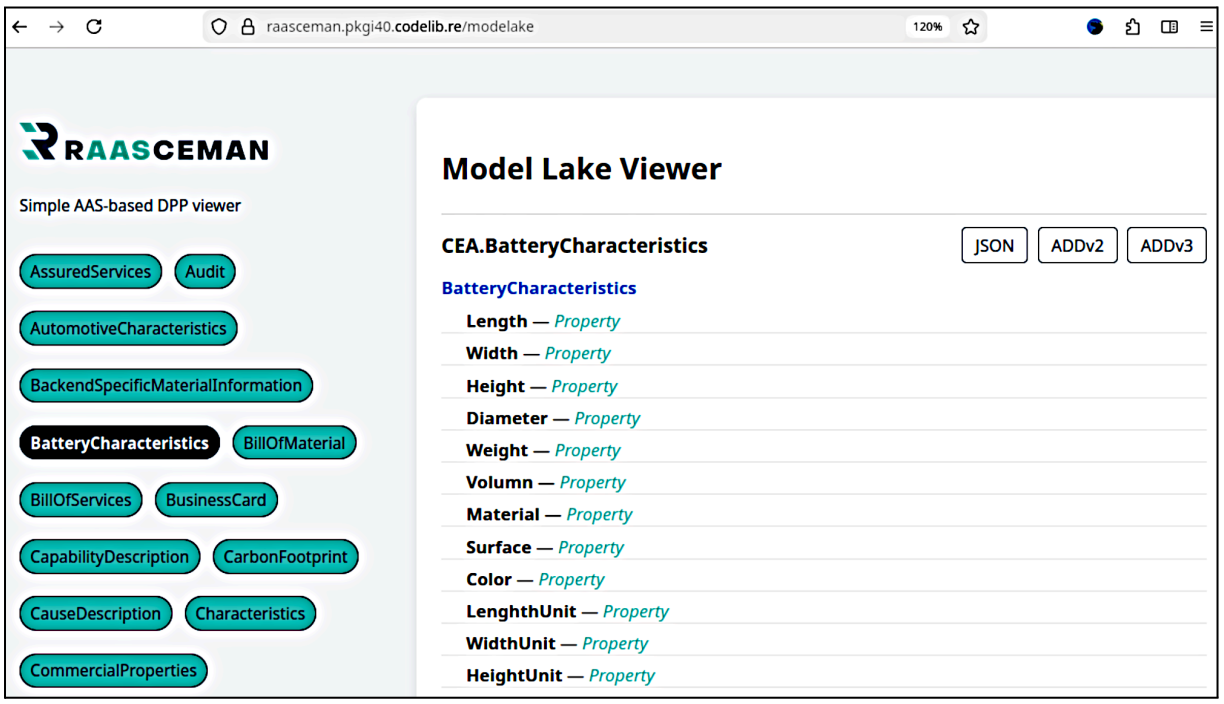


Figure A-1: Simple frontend proposed by CEA to test the Model Lake

Appendix B: Papyrus4Manufacturing Guideline

Papyrus4Manufacturing (**P4M**) has two versions: Desktop and Web (P4MW). Note that the Desktop version supports only the AAS v2 (version 2), and the Web version supports AAS v3 (version 3). Both are available on the P4M website as viewed in Figure B-1.

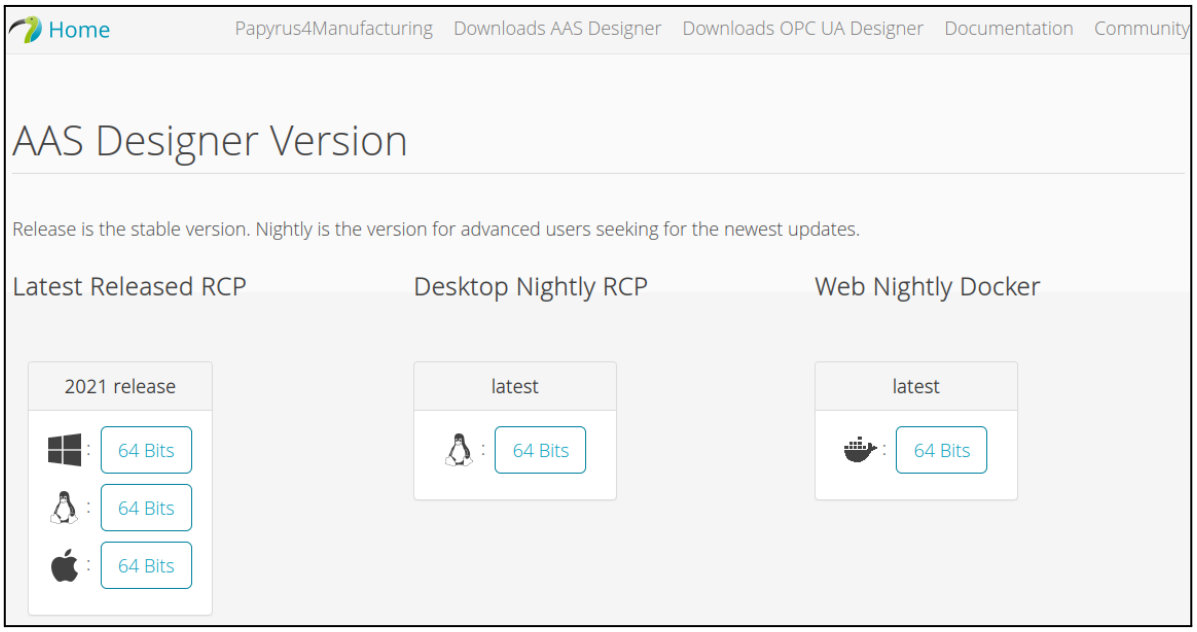


Figure B-1: P4M supports AAS v2 (Desktop Nightly RCP) and P4MW support AAS v3 (Web Nightly Docker)

1- Papyrus4Manufacturing Desktop:

P4M Desktop can be downloaded from the official website of Papyrus4Manufacturing. Users open a web browser to access: https://eclipse.dev/papyrus/components/manufacturing/downloadaas.html, then navigate *Downloads AAS Designer > AAS Designer Version* and choose one of the versions Windows, Linux, or MacOS appropriate to the machine. Note that the versions in the *Latest Released RCP* are outdated (latest updated in 2021) and can cause some issues on new operating systems. The Linux version in *Desktop Nightly RCP* is up to date. The following guideline is for a Linux machine version *Desktop Nightly RCP*.

Extract the downloaded package, then run the Papyrus4Manufacturing application:

```
$ tar –xzvf org.eclipse.papyrus.aas.rcp.product-linux.gtk.x86_64.tar.gz
$ cd Papyrus4AAS
$ ./Papyrus4Manufacturing
```

P4M tool is built upon the Eclipse framework, thus, the GUI of P4M is like one of the Eclipse IDE. The detailed steps of creating a Papyrus Project and designing an AAS submodel template can be found on the official website, in the Documentation tab. It includes basic guides with documents and videos and advanced guides. Figure B-2 illustrates a capture of P4M which represents the submodel Sample.

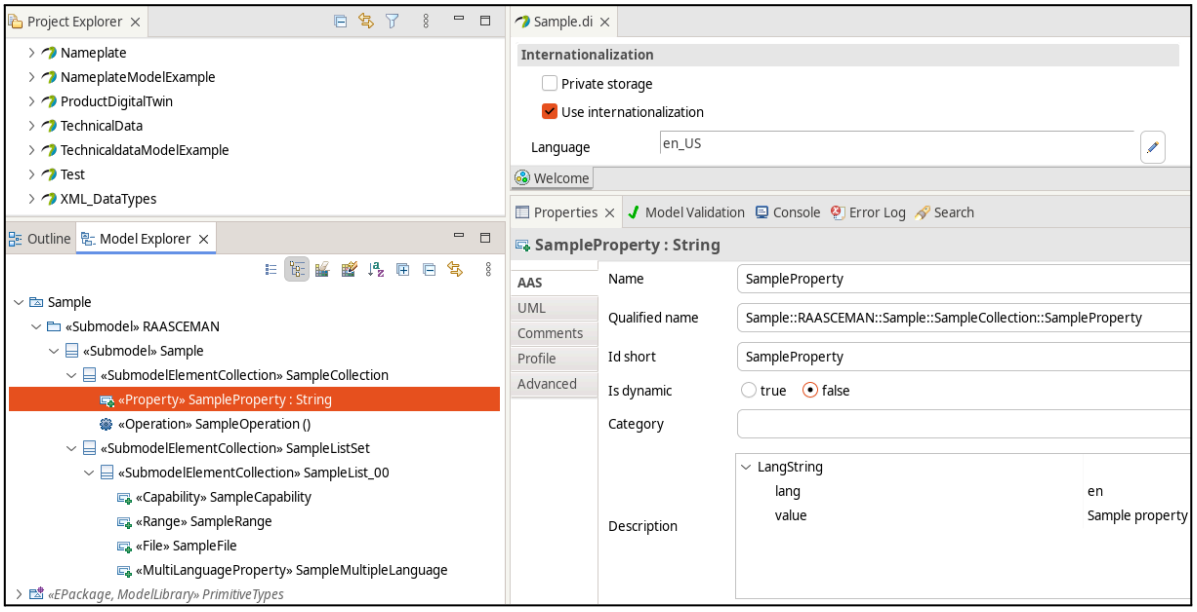


Figure B-2: Submodel Sample template viewed by Papyrus4Manufacturing Desktop version

A Papyrus project includes three basic files: di, notation, and UML. The UML file contains the semantics of the design encoded in the UML/XMI format. Users can convert this format to another, such as JSON for AASX Package Explorer or UML for P4M Web version using the Model Lake's convert feature. The following three basic steps are required:

1. Modify the name of the UML file: Users need to enter the folder containing the Papyrus project, then edit the name of the UML by adding the organization and change the extension from .uml into .xmi

```
$ cd ~/workspace/template/sample/
$ cp Sample.uml RAASCEMAN.Sample.xmi
```

2. Upload the file to Model Lake: Users can use any HTTP tool, such as cURL or Postman to upload the file to Model Lake as presented in Appendix A.

```
$ curl –X POST "https:// your.endpoint/api/v1/upload" -H "accept: application/xmi" -H
"Content-Type: multipart/form-data" -H "Modelake-Key: raasceman" -F
"file=@RAASCEMAN.Sample.xmi"
```

3. Download the formatted file: Users can use any HTTP tool or web browser to donwload the file from Mode Lake as presented in Appendix A.

```
$ curl "https://your.endpoint/api/v1/file/json/RAASCEMAN.Sample.json"
```

2- Papyrus4Manufacturing Web:

P4MW can be downloaded from the official website of Papyrus4Manufacturing. Users open a web browser to access: https://eclipse.dev/papyrus/components/manufacturing/downloadaas.html, then

navigate Downloads AAS Designer > Latest Released RCP and choose the version Web Nightly Docker. Figure B-3 is an example of a submodel template created by P4MW, which illustrates the default layout and structure of an AAS-compliant DT.

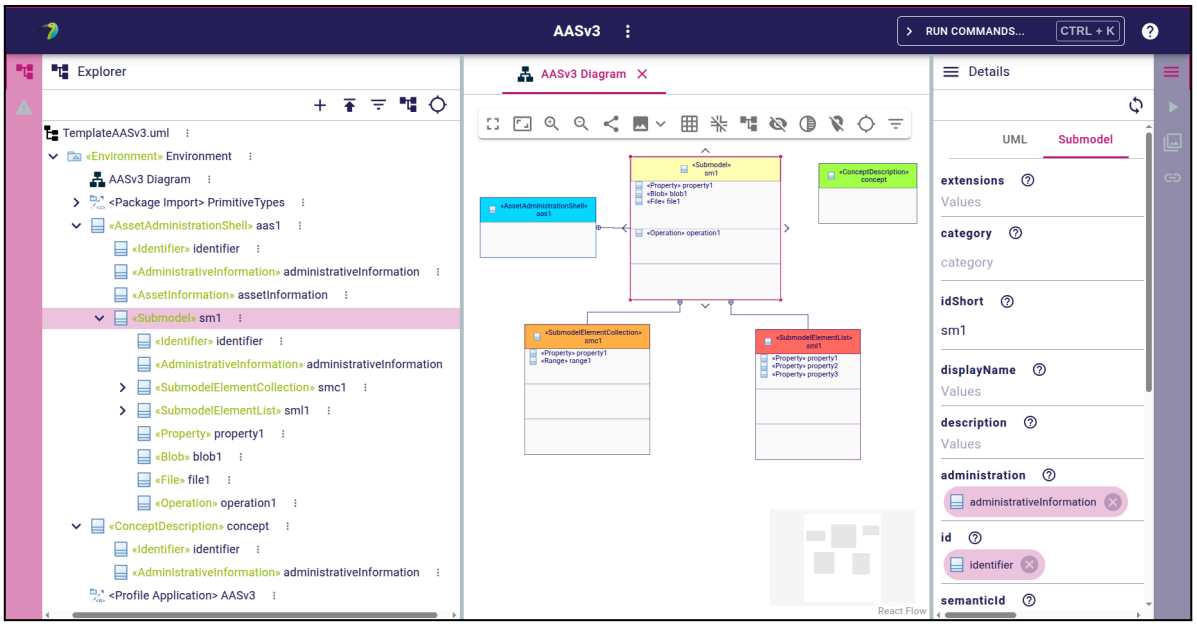


Figure B-3: Submodel Sample template viewed by P4MW

The download folder contains:

- **papyrus-web-app.tar.gz**: Docker image of P4MW.
- **docker-compose.yml** : configuration file used to run the platform with docker-compose.

The following instruction is for a Linux machine on which docker and docker-compose must be pre-installed beforehand.

1. Load the docker image: Navigate to the folder containing the downloaded files and load the Docker image using:

```
$ docker load –i papyrus-web-app.tar.gz
```

2. Start P4MW and the database: Run the following command to start the platform and its database service.

```
$ docker compose up
```

3. Access the platform: Once the containers are running, P4MW can be accessed at http://localhost:9595/ using a web browser. Figure B-4 is the screenshot of the home page of P4MW. From the P4MW home page, right-click on the AASv3 icon. This action initializes a workspace containing the default AAS template where users can begin modeling an AAS-based DT and PDT.

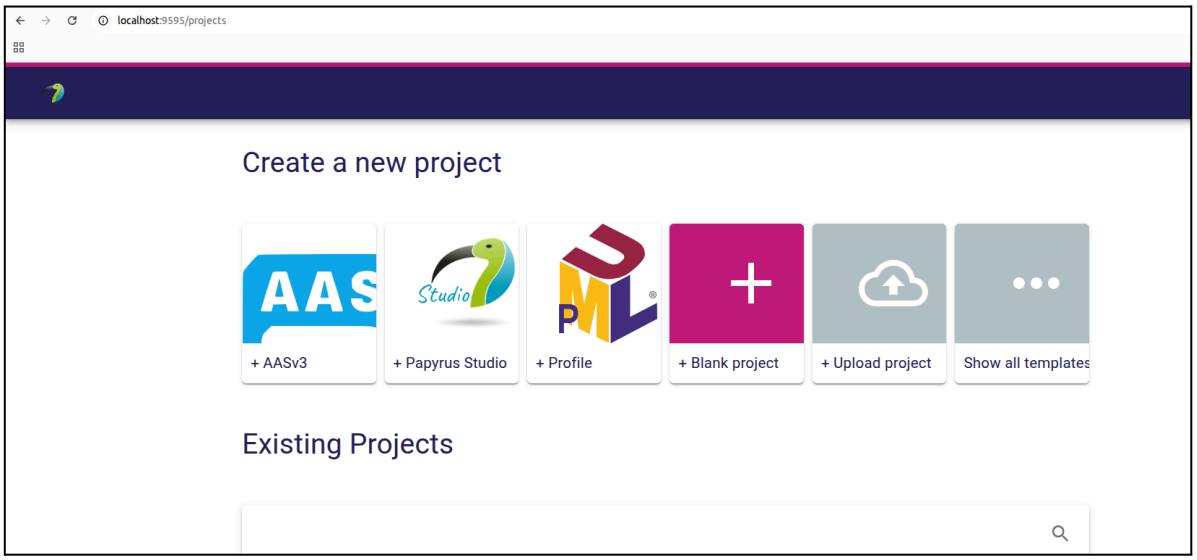
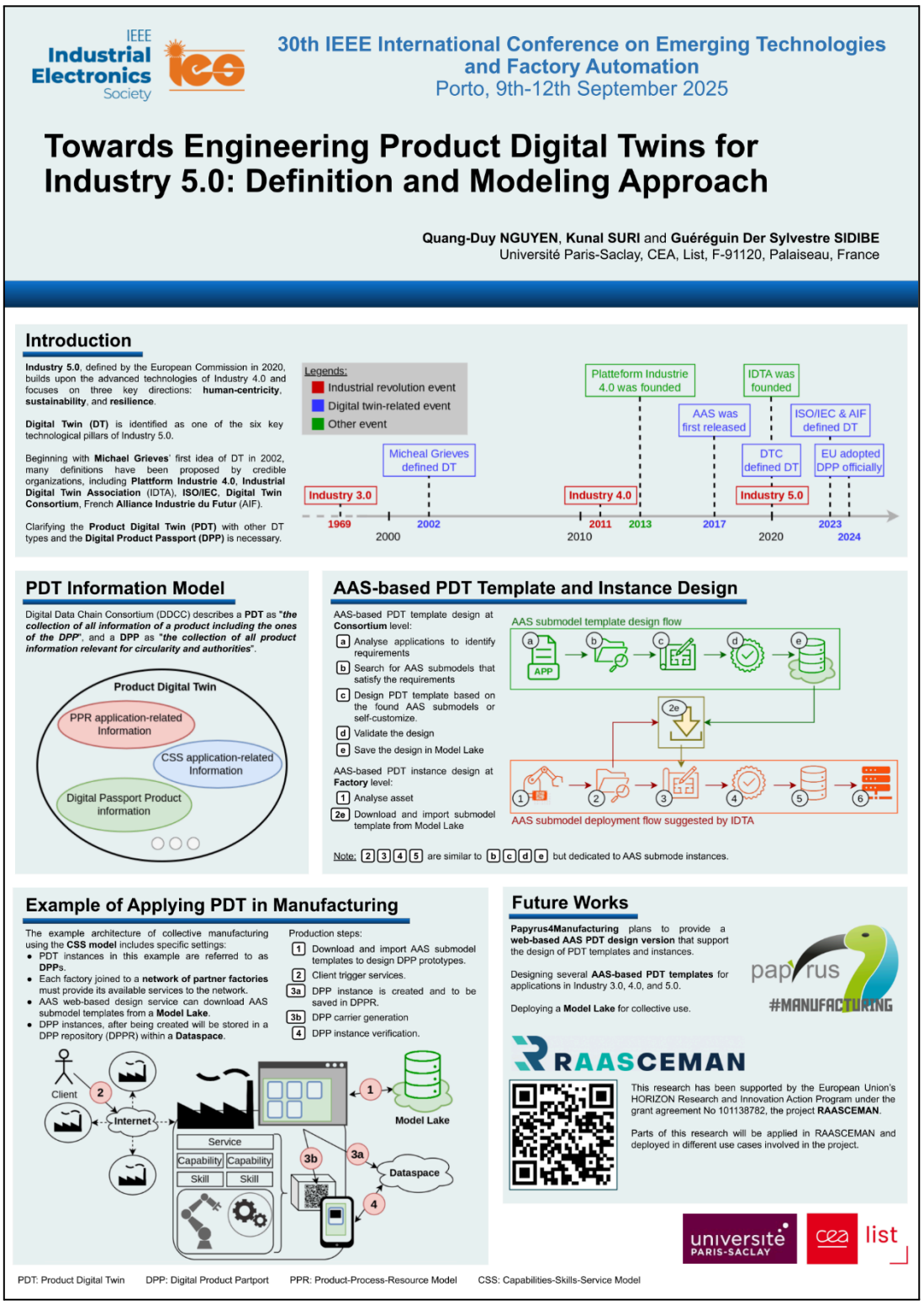


Figure B-4: P4MW's home page

Appendix C: Poster presented in IEEE ETFA 2025



Appendix D: GANTT Chart Dedicated to T2.2 and the Related Tasks

The timeline developing T2.2 is partly and wholly parallel with some other tasks in RAASCEMAN.

	2025											2026			
	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4
T1.3															
T2.1															
T2.2															
T2.3															
T2.4															
T3.1															
T3.2															
T3.3															
T3.4															
T4.1															
T4.2															
T4.3															
T5.1															
T5.2															
T5.3															

Funded by
the European Union