



Deliverable 4.1

SINFONICA Knowledge Map Creation and System Architecture Specification



Funded by
the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101064988.

Version number:	1.0 (Final)
Main authors:	Konstantinos Fokeas, Alexandros Liazos, Anna Antonakopoulou
Dissemination level:	Public
Internal Reviewers:	IRTSX, RELAB
Lead contractor:	ICCS
Due date:	30 November 2024
Delivery date:	4 December 2024

Version history			
Version	Date	Main authors	Summary of changes
0.1	2 nd September 2024	Konstantinos Fokeas, Alexandros Liazos, Anna Antonakopoulou	Table of Contents, Chapters 2 and 3
0.2	1 st October 2024	Konstantinos Fokeas, Alexandros Liazos, Anna Antonakopoulou	Chapters 1 and 5
0.21	1 st November 2024	Konstantinos Fokeas, Alexandros Liazos, Anna Antonakopoulou	Executive Summary and Conclusions
0.22	18 th November 2024	Konstantinos Fokeas, Alexandros Liazos, Anna Antonakopoulou, Evangelos Tsougiannis	Final quality check
0.23	29 th November 2024	Dario Irrera (RE:LAB), Tarek Chouaki (IRTSX)	Peer review
0.3	2 nd December 2024	Konstantinos Fokeas, Alexandros Liazos, Anna Antonakopoulou	Third version after peer reviewers' comments
1.0	4 th December 2024	Andrew Winder, Giulia Renzi	Final review by Quality Manager and submission by Coordinator



Legal disclaimer

Funded by the European Union. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

This document and its content are the property of the SINFONICA Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. SINFONICA partners may use this document in conformity with the SINFONICA Consortium Grant Agreement provisions.

Executive Summary

The "SINFONICA Knowledge Map Creation and System Architecture Specification" deliverable focuses on the methodologies, architectural design, decisions made, technical work and development of the Knowledge Map Explorer, a tool designed to provide targeted insights and guidance on Cooperative, Connected, and Automated Mobility (CCAM) solutions. Positioned between significant milestones within the SINFONICA project, this deliverable advances the architecture and prototype development of the Knowledge Map Explorer, aligning with milestones for the initial, alpha, and beta versions of the tool.

Through structured data collection from diverse groups—including under-researched demographics such as women, disabled individuals, and rural residents—the Knowledge Map Explorer integrates domain-specific insights into CCAM solutions, fostering accessibility and inclusivity. By utilizing recommendation engines approaches, this tool aims to go one step beyond traditional knowledge mapping, which often captures only explicit, quantitative data. This tool also incorporates qualitative knowledge, thus enriching CCAM solution frameworks with the multifaceted needs and expectations of diverse user profiles.

Furthermore, this deliverable specifies the technical architecture of the Knowledge Map Explorer, composed of a Semantic-Based System and a Rule-Based System, each with user-centric modules for profile-based recommendations. This architecture will support end users, including policy-makers, developers, and researchers, in deploying CCAM solutions that are inclusive, user-oriented, and aligned with CCAM users' expectations. By mapping and synthesizing knowledge assets, the Knowledge Map Explorer sets a new paradigm for creating, sharing, and implementing domain-specific guidance in CCAM.

Contents

1. Introduction.....	7
2. Theoretical Framework	8
2.1 Overview of Ontologies	8
2.2 Ontology Languages.....	9
2.3 Ontology Development Tools	15
2.4 Recommendation Engines	16
3. Ontology Development	19
3.1 Design the Ontology Structure	19
3.2 Populate the Knowledge Map	21
4. Knowledge Map Explorer Development	23
4.1 Requirement Analysis and Specifications	23
4.1.1 Target Groups	23
4.1.2 User Stories.....	25
4.1.3 Requirement Analysis and Specifications.....	34
4.2 Implementation of the Rule-Based System	35
4.3 Integration of the Components	36
4.3.1 Components of the Architecture	38
4.3.2 Workflow Summary	39
5. Conclusions.....	41
6. References.....	42

List of tables

Table 1: Spreadsheets Transformation	21
Table 2: Example of a Mapping Definition Function.....	22
Table 3: Stakeholders' Needs.....	23
SINFONICA – D4.1: Knowledge Map Creation and System Architecture Specifications	

Table 4: User Story 1 - Industry Stakeholders.....	25
Table 5: User Story 2 – Service Providers	26
Table 6: User Story 3 - Transport Operators.....	27
Table 7: User Story 4 - Public Administration	28
Table 8: User Story 5 - Research Sector	29
Table 9: User Story 6 - Legislators.....	31
Table 10: User Story 7 - Representative Bodies.....	31
Table 11: User Story 8 - Citizens	32
Table 12: User Story 9 - Large Scale Demonstration projects	33
Table 13: Functional Requirements	34
Table 14: Non-Functional Requirements	34

List of figures

Figure 1: The general RDF statement form (Cyganiak et al., 2014).....	9
Figure 2: An RDF graph example (Saha, 2007).....	11
Figure 3: An overview of the RDFS vocabulary, in terms of classes and properties (Tomaszuk & Haudebourg, 2024)	12
Figure 4: An overview of the RDFS vocabulary, in terms of classes and properties (Tomaszuk & Haudebourg, 2024)	13
Figure 5: Main classes of the KME	20
Figure 6: Part of the ontological schema	20
Figure 7: Example of a Rule.....	35
Figure 8: Architecture of the KME	37

1. Introduction

SINFONICA is creating practical, efficient, and innovative strategies, methods, and tools to engage users, providers, and other stakeholders of Cooperative, Connected, and Automated Mobility (CCAM) ecosystem. This includes citizens (especially vulnerable users), transport operators, public administrations, service providers, researchers, and vehicle and technology suppliers. The aim is to collect, understand, and systematically organize their needs, desires, and concerns regarding CCAM in a manageable and actionable way. SINFONICA will collaboratively develop decision support tools, one of them is the Knowledge Map Explorer (KME), for designers and policy makers to promote the seamless and sustainable deployment of CCAM, ensuring inclusivity and equity for all citizens. However, SINFONICA will not directly deploy, test, or operate any CCAM systems, nor will it process personal data. Instead, it will offer methodologies, guidance, and recommendations. To facilitate this, public datasets will be utilized and gathered—for example, through focus groups and questionnaires.

This deliverable outlines the design, development, and architectural specifications of the SINFONICA Knowledge Map Explorer, a tool dedicated to synthesizing and presenting critical knowledge for CCAM solutions. In collaboration with other work packages and informed by data on user needs, preferences, concerns and challenges, this tool is designed to play a pivotal role in connecting CCAM deployers, stakeholders and users with tailored insights. Positioned between three core milestones—Milestone 10 (Architecture and Ontology Specification), Milestone 13 (Alpha Prototype), and Milestone 16 (Beta Prototype)—this deliverable highlights the iterative progress and alignment with SINFONICA’s overall goals for inclusive CCAM development. The system architecture specification translates this conceptual framework into a practical implementation plan. It details the technological components, integration strategies, and configuration settings required to realize the KME’s objectives.

The Knowledge Map Explorer serves as an intelligent navigation system that offers stakeholders specialized guidance on implementing CCAM solutions, based on the user type, context, and scenario. The system leverages ontologies and expert-driven mapping, employing tools like the Web Ontology Language (OWL) for structured representation. By employing a dual-system architecture, the Semantic-Based System and the Rule-Based System, this deliverable specifies how the Knowledge Map Explorer will recommend best practices in CCAM, based on explicit and inferred knowledge. The result is a tool that empowers stakeholders with accessible, domain-specific insights, supporting the development of truly inclusive CCAM technologies.

This deliverable is structured into four main chapters starting from a theoretical background of ontologies and recommendations engines along with their advantages and disadvantages (chapter 2), moving to a description of the ontology development (chapter 3), while the last chapter 4 focuses on the implementation steps for building such a system like the Knowledge Map Explorer.

2. Theoretical Framework

2.1 Overview of Ontologies

Definition

Objects, concepts and other entities belonging in an area (domain) of interest, along with the relationships amongst them, define an abstract and simplified view of formally represented knowledge, known as conceptualization (Genesereth & Nilsson, 1987). An ontology is an explicit representation of such a conceptualization, or in other words, a structured framework for system modelling, i.e., its entities and inter-and outer-relationships, that enables computers to understand and process the semantics of information (Guarino, Oberle, & Staab, 2009).

Components

An ontology can be, mathematics-wise, defined as a tuple, and its components are concepts (classes), relationships, functions, instances (objects) and axioms (Gruber, 1993).

Concepts (classes) are the main modelled elements, entities, “things” of the domain; each concept belongs to a taxonomy, while it has to comply to their descriptions given by properties (Reyes-Pena & Tovar-Vidal, 2019).

Properties can be of, mainly, two types; object and datatype ones. Object properties connect elements with elements, while datatypes properties connect elements to datatype values, such as integers, strings, date etc. A property is characterized by its specified domain and range, while object properties in particular may have an inverse property (e.g. “hasFriend”, “hasEnemy”) (Saha, 2007).

Relationships link the concepts among each other and those interconnections from the scheme or structure of the ontology, in a taxonomic or non-taxonomic way. Functions are special relationships-elements of an ontology representing the formulas for calculating information using data from other elements. Instances are created to depict a concept, a relationship or another element of a taxonomy, at a particular time moment. Axioms model sentences that always hold true and set rules and restrictions using logic arguments that relationships between ontology objects should respect. Annotations are an ontology’s meta data; they provide additional information about the ontology elements, such as labels or comments (Reyes-Pena & Tovar-Vidal, 2019).

Ontology Evaluation Criteria

Each ontology should be verified (be built ensuring that it fulfils the ontology requirements, and that it can function in real world conditions), validated (have its models checked in terms of correctly depicting the known, real world) and assessed (be judged in terms of the usefulness, understandability and portability of its components and definitions from a user’s point of view) (Gomez-Perez, 2004).

An ontology can be evaluated in terms of (Gomez-Perez, 2004):

1. **Consistency:** a definition of an ontology is considered consistent if and only if both the formal and the informal definition do not contradict with the real world, and both mean the same – individual consistency, and contradictory sentences can't be concluded by other definitions or axioms – inferential consistency.
2. **Completeness:** An ontology is considered complete if and only if, firstly, everything supposed to belong to it, is clear stated within it or can be easily concluded by it, and secondly, each of its definitions is complete.
3. **Conciseness:** if the definitions within an ontology do not have any redundancies by themselves, and also no redundancies can be inferred from other definitions interconnected with them, the ontology is concise.
4. **Sensitiveness:** describes how sensitive are an ontology's already well-defined properties in minor alterations in a definition.
5. **Expandability:** how easy is it to add new information to an ontology and more knowledge to its definitions without changing the already well-defined properties.

2.2 Ontology Languages

Ontology languages are tools for representing and managing knowledge. They provide a structured framework to define concepts, relationships, and constraints, enabling machines to interpret and process information. Several ontology languages are available, each with its own set of advantages and disadvantages that make them suitable for different applications.

RDF (Resource Description Framework) - RDFS (RDF Schema)

RDF is a general framework, initially adopted in 1999, utilized to represent information on the web, characterized by simple semantics that can be expressed via XML (Saha, 2007). It is designed similarly to other conceptual models, such as the entity-relationship models and class diagrams. The basis of this framework is an abstract syntax, composed of triples. Each triple is an RDF statement, defined as a relationship, indicated by the predicate, that holds between the subject (the resource) and the object (Cyganiak, Hyland-Wood, & Lanthaler, 2014). The above is depicted as a directed graph, representing subject and object as nodes, and the predicate as an arc:

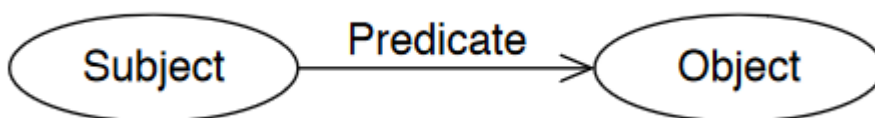


Figure 1: The general RDF statement form (Cyganiak et al., 2014)

Each of the three parts can hold a Uniform Resource Identifier (URI), while the object can also hold a literal value. E.g., to represent the notion “The vehicle has speed equal to 60 km/h” in RDF, the subject denotes the “Vehicle”, the object denotes the literal value “60 km/h”, and their relationship is denoted by the predicate “has speed equal to”.

Two or more RDF triples/statements form a directed, labelled RDF multigraph. At any given time, an RDF data model is a set of static snapshots of information, although if combined with appropriate vocabulary terms, RDF graphs can carry information about events and temporal aspects. RDF triples are a coded depiction of a simple logical expression or a claim, while the RDF graph is the logical join (“AND”) of the triples it corresponds to (Saha, 2007).

Some relationships between RDF graphs are the following (Cyganiak, Hyland-Wood, & Lanthaler, 2014):

- Entailment: An RDF graph A entails another RDF graph B if every possible arrangement of the world that makes A true also makes B true; thus, in such a case, if graph A holds demonstrated truth, then the truth of graph B is automatically established.
- Equivalence: If graphs A and B entail the same claims, they are considered equivalent; in this case, graph A entails B while B also entails A.
- Inconsistency: If some of the statements of a graph conflict with each other, in every possible world arrangement, the RDF graph is inconsistent.

Within an RDF schema, nodes are divided into three types (Cyganiak, Hyland-Wood, & Lanthaler, 2014):

- International Resource Identifiers (IRIs),
- Literals, and
- Blank nodes.

IRIs and literals denote real entities – anything from physical things, abstract concepts, strings or numbers. IRI gives an entity’s referent, while literal gives its literal value (non-alterable) and is connected with a datatype (e.g., string, number, date). The predicate of an RDF statement is always an IRI and denotes a property (a binary entity). A blank node does not denote any (Cyganiak, Hyland-Wood, & Lanthaler, 2014).

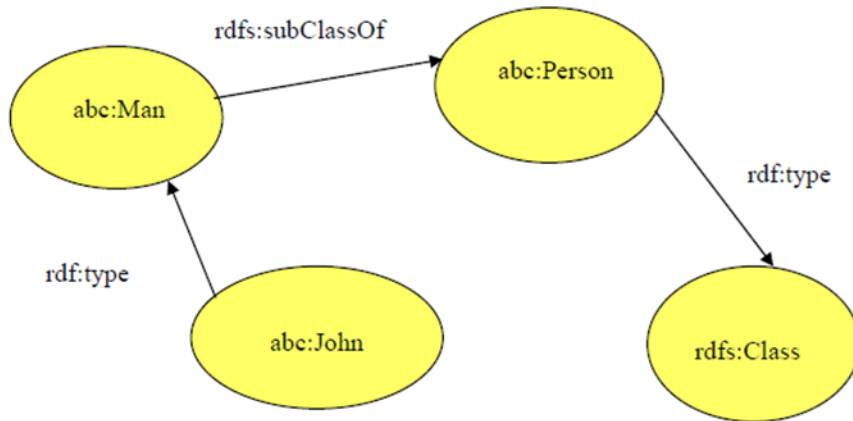


Figure 2: An RDF graph example (Saha, 2007)

In terms of serialization, i.e., translating a data structure into a storable or transmittable form, the serialization formats, as defined by W3C, are the following:

- Turtle (Terse RDF Triple Language)
- N-Triples (Notation of Triples)
- N3 (Notation 3)
- RDF/XML

The RDF schema (RDFS or RDF-S) is constructed by the vocabulary of RDF, acting as a semantic extension of it. RDFS consists of a set of classes that can describe ontologies. The first version of RDFS was published in 1999 while its most recent version is the 1.1, published in 2014. All resources are divided into sets, defined as Classes, while each member of a class is an Instance of this class. The `rdf:type` property denotes that a particular resource is an instance of a particular class. In RDFS, two classes can differ while having exactly the same set of instances.

An overview of the RDFS vocabulary, in terms of classes and properties, follows:

Class name	comment
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:langString	The class of language-tagged string literal values.
rdf:HTML	The class of HTML literal values.
rdf:XMLLiteral	The class of XML literal values.
rdf:JSON	The class of JSON literal values.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements.
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, <code>rdf:_1</code> , <code>rdf:_2</code> , ..., all of which are sub-properties of 'member'.
rdf:List	The class of RDF Lists.

Figure 3: An overview of the RDFS vocabulary, in terms of classes and properties (Tomaszuk & Haudebourg, 2024)

Property name	comment	domain	range
rdf:type	The subject is an instance of a class.	<code>rdfs:Resource</code>	<code>rdfs:Class</code>
rdfs:subClassOf	The subject is a subclass of a class.	<code>rdfs:Class</code>	<code>rdfs:Class</code>
rdfs:subPropertyOf	The subject is a subproperty of a property.	<code>rdf:Property</code>	<code>rdf:Property</code>
rdfs:domain	A domain of the subject property.	<code>rdf:Property</code>	<code>rdfs:Class</code>
rdfs:range	A range of the subject property.	<code>rdf:Property</code>	<code>rdfs:Class</code>
rdfs:label	A human-readable name for the subject.	<code>rdfs:Resource</code>	<code>rdfs:Literal</code>
rdfs:comment	A description of the subject resource.	<code>rdfs:Resource</code>	<code>rdfs:Literal</code>
rdfs:member	A member of the subject resource.	<code>rdfs:Resource</code>	<code>rdfs:Resource</code>
rdf:first	The first item in the subject RDF list.	<code>rdf:List</code>	<code>rdfs:Resource</code>
rdf:rest	The rest of the subject RDF list after the first item.	<code>rdf:List</code>	<code>rdf:List</code>
rdfs:seeAlso	Further information about the subject resource.	<code>rdfs:Resource</code>	<code>rdfs:Resource</code>
rdfs:isDefinedBy	The definition of the subject resource.	<code>rdfs:Resource</code>	<code>rdfs:Resource</code>
rdf:value	Idiomatic property used for structured values.	<code>rdfs:Resource</code>	<code>rdfs:Resource</code>
rdf:subject	The subject of the subject RDF statement.	<code>rdf:Statement</code>	<code>rdfs:Resource</code>
rdf:predicate	The predicate of the subject RDF statement.	<code>rdf:Statement</code>	<code>rdfs:Resource</code>
rdf:object	The object of the subject RDF statement.	<code>rdf:Statement</code>	<code>rdfs:Resource</code>

Figure 4: An overview of the RDFS vocabulary, in terms of classes and properties (Tomaszuk & Haudebourg, 2024)

RDF's simplicity makes it accessible for developers and allows for the incremental development of ontologies. It supports linking data from diverse sources, which is crucial for the semantic web. While RDF on its own lacks the expressiveness of languages like OWL, it can be extended with RDFS to provide basic schema definitions. This combination is often sufficient for many applications that require a balance between expressiveness and performance.

The disadvantage of RDF is that it does not inherently support advanced reasoning or constraints. For applications that require inferencing capabilities, relying solely on RDF may not be enough. However, it can be combined with more expressive languages to achieve the projects' goals. (Antonioni & van Harmelen, 2004).

OWL (Web Ontology Language)

OWL is one of the most widely used ontology languages and includes a family of languages that define and describe web ontologies in formal semantics. In 2004 and 2009 respectively, OWL and OWL2 were introduced. OWL is built and based on the RDF framework, while being more expressive than RDFS, and thus, it can describe more complex ontologies and support richer semantics (Saha, 2007). Its semantics include classes, properties, individuals, and data types, enabling reasoning and inference capabilities. This makes OWL ideal for applications requiring knowledge representation, such as semantic web services (Motik & Horrocks, 2008). However, the high expressiveness of OWL comes with increased computational complexity. Reasoning over OWL ontologies can be resource-intensive, which may impact performance in large-scale or real-time systems. Additionally, the complexity of OWL can impose difficulties for developers new to ontology modelling.

OWL ontology structure is specified in high level syntax, that defines the ontology as a series of axioms, facts and annotations; axioms and facts provide information about the classes, properties and instances of the ontology, while annotations are the meta-data of the ontology, understandable both by humans and computers. Each ontology in an OWL syntax should have a unique Uniform Resource Identifier (URI), whilst each class, property and instance can either also hold a URI or be anonymous (Motik, Patel-Schneider, & Parsia, OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), 2012).

For example, an ontology based on a class called Vehicle, described in OWL2 XML syntax:

```
<Ontology ontologyIRI="http://example.org/vehicle.owl" ...>
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#" />
  <Declaration>
    <Class IRI="Vehicle" />
  </Declaration>
</Ontology>
```

The most well-known sub-languages of OWL are:

- OWL Lite, that can express simple hierarchy and constraints,
- OWL DL, which is characterized by computational completeness and decidability, and
- OWL Full, which gives the most freedom of expression and can enrich the meaning of the pre-defined vocabulary

Simple Knowledge Organization System (SKOS)

SKOS is designed for representing simple knowledge structures like taxonomies and vocabularies. It provides a lightweight framework that is easy to use and understand, making it suitable for applications in library science and knowledge management (Baker, et al., 2013). However, SKOS's limited expressiveness restricts its ability to model complex relationships and constraints. It lacks support for advanced reasoning features, which may be necessary for advanced applications.



Given the advantages and disadvantages of each ontology language and the requirements of the project, it was decided to utilize the RDF and OWL capabilities.

2.3 Ontology Development Tools

Since the development of an ontology is a complex and domain-oriented process, tools for modelling, editing, visualizing, validating and integrating ontologies are essential, and thus, many of them have been introduced since the development of ontology languages. A user can define concepts, relationships and instances (Noy & Musen, 2004). The most well-known ones are Protégé, SWOOP, NeOn Toolkit TopBraid composer, Ontolingua, OntoTrack and Knoodl.

Protégé

Protégé is an open-source ontology editor and knowledge-based framework, based on Java; it is quite flexible for a user especially for application development and prototyping. Ontologies within Protégé can be modelled via two types of editors: Protégé-Frames and Protégé-OWL. It can export ontologies into RDF, OWL, XML and other formats. OWLViz, a visual editor for OWL, storage back-ends to Jena (a Java framework for semantic Web applications) and Sesame (an open-source RDF database), as well as a variety of plugins, make Protégé a flexible and extensible ontology development tool (Stanford, n.d.).

NeOn Toolkit

NeOn Toolkit is a strong ontology editor, open-source and extensible with many available plugins. It is known for its ability to support “heavy” projects, such as multi-lingual or multi-modular ontologies (NeonToolkit, n.d.).

SWOOP

Swoop is a small and simple web ontology browser and editor, tailored to OWL ontologies. Its User Interface (UI) is based on the familiar, standard web browser, using URIs as its key elements for constructing an ontology (Mindswap, 2024).

TopBraid Composer

TopBraid Composer is a commercial, multi-purpose semantic web editor for managing ontologies and linked data. It can support development, management and testing of any knowledge model configuration, while its framework is extensible through published APIs for client-server or browser-based solutions. Through Composer, RDFS, OWL, SPARQL Query language and Semantic Web Rule Language (SWRL) can be developed, and scalable database back-ends such as Jena are provided (TopBrain, n.d.).

Ontolingua

Ontolingua is one of the oldest servers related to ontologies, running since 1995. It supports community and distributed editing, browsing and development of ontologies and comes with a

form-based web application. Ontolingua Frame Ontology language and KIF are the representation languages compatible with the server (Farqhar, Fikes, & Rice, 1997). Sophisticated tools responsible for the development and maintenance of frame-based ontologies have also been built throughout the years (Ahmad & Colomb, 2007).

OntoTrack

OntoTrack is a non-commercial editor, developed as an ontology authoring tool that provides a graph-based ontology layout, allowing many editing features. It also comes with an external reasoner that comes with every modelling and editing step. OntoTrack has been developed in Java and is mainly connected with OWL Lite (OntoTrack, n.d.).

Knoodl

Knoodl is an editor that supports community-oriented OWL-based ontologies and RDF-based knowledge bases; it also offers interfaces based either on Java or SPARQL, thus provides flexibility to communities to develop their semantic applications and knowledge bases. It is hosted in the Amazon EC2 cloud (Knoodl, n.d.).

These tools offer a range of functionalities to support various aspects of ontology development, from basic editing and visualization to advanced reasoning and collaborative features. For ontology development, Protégé was chosen, because it provides support for reasoning engines like HermiT and Pellet, which are essential for performing reasoning tasks. Its full compliance with OWL (Web Ontology Language) allows for the creation of expressive and semantically rich ontologies compatibilities while also supports RDF integration with other data sources and tools that utilize standard data formats.

2.4 Recommendation Engines

Recommendation engines (RS) are algorithms designed to suggest relevant items or content to users based on their preferences, behaviours, or other data (Ma, 2024). RS can enhance user experience, increasing engagement, and increase value by presenting users with content they are more likely to find useful. These systems are widely used in industries like e-commerce, streaming services, social media, and online advertising to provide personalized experiences (Fayyaz, 2020).

Recommendation engines rely on several approaches to predict user preferences and provide tailored recommendations. The most common techniques include:

- **Collaborative Filtering:** This approach makes recommendations based on the behaviour and preferences of similar users. It can be further separated into two distinctive kinds of filtering:
 - **User-based collaborative filtering:** Recommends items that similar users have liked. If user A and user B have shown similar behaviours in the past, the system will suggest items to user A that user B has interacted with, and vice versa.
 - **Item-based collaborative filtering:** Recommends items similar to those the user has liked in the past. For example, if a user liked a particular movie, the system suggests movies that other users who liked that movie also enjoyed.

Traditionally collaborative filtering-based systems suffer from the cold-start problem and privacy concerns as there is a need to share user data. However, collaborative filtering approaches do not require any knowledge of item features for generating a recommendation.

- **Content-based Filtering:** This method focuses on recommending items that are similar to those a user has liked in the past. It examines the attributes/features of items and matches them with user preferences. A major disadvantage is the need for a priori extraction of features that optimally describes the items (M. Marcuzzo).
- **Machine Learning-Based Approaches:** Deep learning models are common in recommendation engines like e-commerce and streaming platforms. These models can capture complex patterns in user behaviour and content. They process vast amounts of data to provide highly personalized recommendations by learning from user interactions, metadata, and even contextual signals like time of day or location.
- **Knowledge-Based Recommender Systems:** These systems use explicit user preferences or expert knowledge to suggest items. This method is typically used when recommendations need to be based on detailed and explicit requirements rather than past behaviour (Fayyaz, 2020).
- **Hybrid Models:** Combination of collaborative and content-based filtering to leverage the strengths of both methods. This approach uses a hybrid model by combining user behaviour, viewing history, and content similarity to provide accurate recommendations.

Given the nature of SINFONICA project and the available data it was decided that the best approach would be a Knowledge-Based Recommendation System. Furthermore, this type of recommendation engine which is based on a **rule-based systems** and **ontologies**, is ideal in domains where you have well-defined relationships (taxonomy), attributes, and specific user needs or domain knowledge. In summary this type of recommendation system relies especially on explicit rules and structured knowledge rather than data-driven machine learning models. Some of the key concepts of that kind of recommendation engines are described as follows:

1. **Rule-Based System:** A rule-based system uses if-then logic to generate recommendations. Rules are manually defined and encoded based on domain knowledge.
2. **Ontology:** In recommendation systems, ontologies can help define relationships between users, items, attributes, and categories. It provides a semantic layer, offering more meaningful recommendations by leveraging domain-specific knowledge.

A recommendation system like this is based on defining an ontology that captures application's domain, the relationships between entities such as users (preferences and demographics), items features and attributes. Tools like OWL (Web Ontology Language) and Protégé assist in creating and managing this ontology, while RDF (Resource Description Framework) structures the information about these resources. Next, you define the rules that govern your recommendation engine by mapping user preferences to items based on the relationships in your ontology. For example, if a user likes a specific genre, the engine recommends items from the same genre; if a user enjoys works by a particular movie director, it suggests other movies by that director. Tools and programming languages such as SWRL (Semantic Web Rule Language), Drools, and Jena help in expressing and managing these rules. After establishing the rules, an inference engine is established, that applies them to the ontology to generate recommendations by analysing user profiles and item attributes. Ontology reasoners like Pellet perform reasoning tasks and inference, while Drools



manages complex rule-based systems by matching rules against user preferences. Finally, you deliver the recommendations to users by exposing the engine's results through a REST API. The frontend displays these recommendations in real time. Technologies involved in this process include Protégé for ontology development and as rule engine, OWL/SWRL for rule definition, Apache Jena as a SPARQL server, Blazegraph or Fuseki for RDF storage, and suitable backend and frontend technologies for serving and presenting the recommendations.

3. Ontology Development

3.1 Design the Ontology Structure

Within the concept of Task 4.1: Technical creation of knowledge map and T4.2: SINFONICA Knowledge map explorer architecture specification, the classes and their hierarchies were structured into a framework. The ontology structure had to capture the interactions and dependencies within SINFONICA domain. After having this structure in place, the instances or else put the individuals were added within the ontology following the project's requirements. These instances were created based mainly from data derived from interviews, focus groups and workshops.

For the requirements of SINFONICA project Protégé tool was used, which is an open-source ontology editor and knowledge management system. Even though listing all the classes and subclasses created within the project is out of scope of this deliverable, the main classes along with their description is listed below:

1. The **Data Concept** represents elements consolidated from already available open access ontologies, research activities within SINFONICA, while also includes data collected from interviews, focus groups and workshops.
2. The **Domain Concept** consist of a wide range of entities and information relevant to the CCAM (Connected, Cooperative, and Automated Mobility) ecosystem. It includes the stakeholders and citizens preference and priorities such as education, work, and leisure/sports, as well as participants' mobility behaviours. Additionally, it accounts for properties related to transportation and data concerning user acceptance.
3. The **General Concept** includes individual's personal situation, and specific characteristics of users such as age, income level, ethnicity, etc. These elements help provide context about users and address their unique traits.
4. The **Recommendations** class is used to organize different recommendation types, depending on the subject. These recommendations include guidelines and best practice towards safety, accessibility, affordability, or other topics depending on the context and requirements.

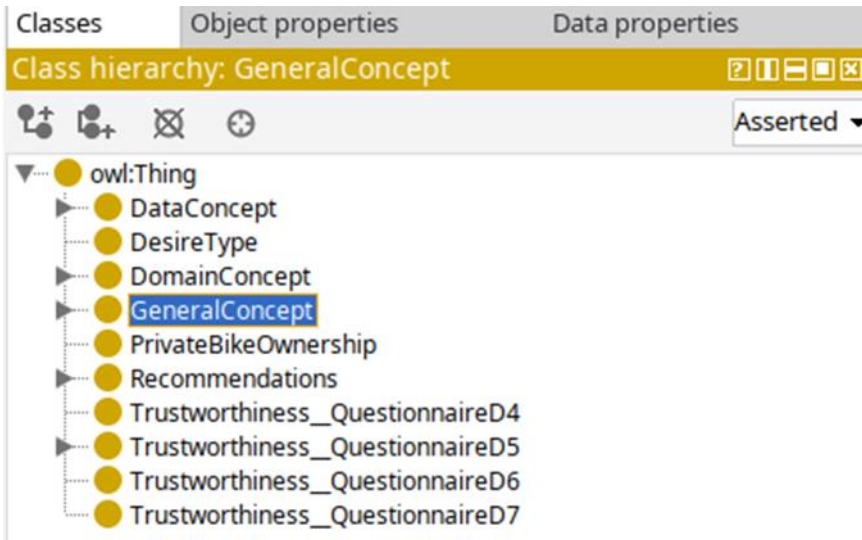


Figure 5: Main classes of the KME

The following image depicts an ontological schema created in Protégé, illustrating various classes, properties, and relationships used to model concepts and interactions.

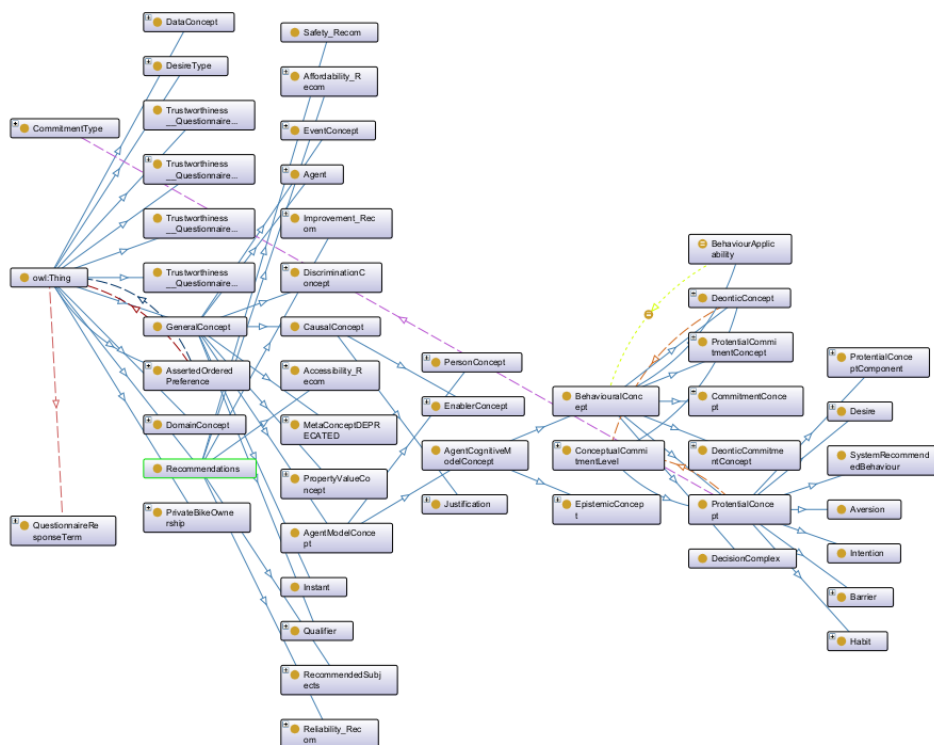


Figure 6: Part of the ontological schema

3.2 Populate the Knowledge Map

Populating a knowledge map from structured data like spreadsheets into an ontology can be streamlined using **Cellfie**, a tool within the **Protégé** ontology editor. Cellfie is a common way to transform spreadsheet data into structured ontology models, while this process enables you to visualize and manage complex information more effectively. Below is a brief description on how Cellfie was used for populating a knowledge map from spreadsheet data (interviews and focus groups).

The spreadsheets had to be transformed into a well-structured machine-readable format before importing them into the ontological schema. The structure of the spreadsheet had to be transformed into columns, rows and headers as:

- **Columns:** Each column should represent a specific attribute or property.
- **Rows:** Each row should correspond to an individual instance or entity.
- **Headers:** Use clear and descriptive headers for each column to avoid confusion.

An indicative structure of spreadsheet structure looks like the table below:

Table 1: Spreadsheets Transformation

Class	Instance ID	Property	Value
Agent	Participant	believes	CCAM is trustworthy
Agent	Participant	hasHabit	Use Private Car
Agent	Participant	hasGender	Female

After having the spreadsheets into the right format then the data records had been cleaned and follow consistencies in names, dates etc.

- **Remove duplicates:** Ensure no repeated instances.
- **Fill missing values:** Address any gaps in the data.
- **Standardize formats:** Use consistent formats for dates, names, etc.

Given that Cellfie uses a specific syntax to map spreadsheet data to ontology classes and properties, a mapping definition had to be established. The mapping definition specifies how each cell in the spreadsheet corresponds to an element in the ontology. The mapping file that defines how your spreadsheet data will populate the ontology is a plain text. Here is an example of a simple mapping definition:

Table 2: Example of a Mapping Definition Function

Mapping Definition Function	Comments
<pre> Class: @A* Annotations: rdfs:label @C* SubClassOf: :hasProperty some (rdfs:label "name"^^xsd:string and :value @D*) </pre>	<ul style="list-style-type: none"> • @A* refers to the content in column A. • @C* refers to the content in column C. • @D* refers to the content in column D.

The same procedure as described above had to be followed for every file received from the Groups of Interest, containing data from the interviews and the focus groups.

4. Knowledge Map Explorer Development

In this chapter, the development of back end system is illustrated, starting with a Requirement Analysis and Specifications to identify the core functionalities based on end users' needs and user stories. Follows the development of the semantic-Based System, we detail the Implementation of the Rule-Based System, showcasing how predefined rules drive system behaviour. Finally, it is discussed about the integration of the Components, illustrating how the semantic and rule-based systems are cohesively combined to deliver a unified solution that meets the specified requirements.

4.1 Requirement Analysis and Specifications

4.1.1 Target Groups

The Knowledge Map Explorer's (KME) success is dependent on the adoption of its results by the end users. Therefore, the pathways to impact can only be credible upon how well its results respond to target groups specific needs. A well characterization of the end users has been established on the preparatory stage of SINFONICA, this will allow the KME to address their specific needs and expectations. The information, summarized in Table 2, along with the user stories that follows, are the first step to derive the functional and non-functional requirements of the tool.

Table 3: Stakeholders' Needs

Stakeholders	Specific needs / competencies	SINFONICA's impact
CCAM Industry actors (technology developers & suppliers, vehicles' manufacturers, service providers.)	Need to know the point of view of future users to try to adapt new technologies to their real needs. For start-ups and SMEs, need to access market opportunities and needs of the market. Need to adopt a user-centred design to deliver inclusive, affordable, sustainable, and resilient solutions, both for service providers and manufacturers. Need to foster standardization processes to avoid fragmentation in CCAM development and deployment.	Guidance on designing CCAM which meets the needs of different types of users and a checklist of considerations to be considered. Recommendation based on co-creation with relevant stakeholders and users involved in the SINFONICA activities.
Transport operators (Road authorities, Transport authorities)	Need to know how citizens, future users and stakeholders approach the CCAM world, considering also - and above all - the point of view of vulnerable road users. Need channels to raise attention on relevant issues related to public transport.	As above, plus guidance for informing road users (both users and nonusers of CCAM), considering that deployment will not be uniform everywhere.
Public administration (Municipalities, provinces,	Need to have access to knowledge on CCAM Innovation. Need to adopt reliable methodologies for	Guidance for local and regional authorities on citizens and stakeholder groups

Stakeholders	Specific needs / competencies	SINFONICA's impact
regions, Local Transport Authorities)	stakeholders' engagement and be guided in starting discussions with citizens and define participatory processes. Need to adjust legislation and local regulations in the light of the digital transition in the mobility domain.	interaction, as well as policy recommendations depending on the context. SNFONICA will provide user-friendly tools that will be particularly relevant to those local and regional authorities with a low level of CCAM experience and deployment.
Research sector (Universities Research centres, R&I departments)	Need to increase the knowledge base to start new studies and develop innovative products; include social considerations in innovation studies and projects.	improved knowledge on assessment and evaluation of CCAM solutions from a socio-economic perspective.
Legislators (insurers, policy makers and regulators)	Need to adapt policies and regulations of CCAM operations and deployments, to ensure safety, equity, fairness and inclusiveness.	Provide recommendations and guidance on inclusiveness, accessibility of CCAM future deployments, thus setting the basis for new ethic and regulatory frameworks of CCAM.
Representative bodies (Road users and stakeholders Associations)	Need to ensure that the needs and perspective of citizens are addressed, to create user acceptance and awareness.	Share knowledge on how to communicate with users to increase the CCAM acceptance.
Citizens (vulnerable users, commuters, citizens from rural or peripheral areas)	Need to fully understand the potential of implementing CCAM in public transport. Need to contribute to discussions about the development of CCAM innovative solutions.	The social dimension is very important in the development of the SINFONICA project. The activities and the data collection are carried out with an inclusive approach, to understand needs and expectations regarding CCAM.
Large Scale Demonstration projects	Needs to improve the comparability of results from different projects; improved benchmarking to gain a better understanding when making investment or deployment decisions based on project results.	Guidance and recommendations, providing tools to help projects deliver demonstrations and evaluation methodologies more efficiently, enabling robust and comparable results to be obtained.

4.1.2 User Stories

User stories are concise descriptions of a functionality from the perspective of the end-user. Since SINFONICA is a research project, we didn't have the end users onboard in order to engage them into this activity, so we run a round of brainstorming with the partners of the consortium, instead.

Table 4: User Story 1 - Industry Stakeholders

Story ID	01
Story Title	Enhanced CCAM Technology Integration through Personalized Recommendations and Interactive Knowledge Map.
SINFONICA Partner	ICCS
Story Actor	Industry (e.g., Technology developers & suppliers, Automobile manufacturers, Automotive suppliers, Telecom industry).
Overview	An automotive supplier needs an efficient way to explore and integrate Connected, Cooperative, and Automated Mobility (CCAM) technologies into their product development process.
Triggers	The desire to streamline the research and selection process for CCAM technologies.
Goal	Provide a comprehensive tool that offers tailored recommendations and an interactive knowledge map for exploring the landscape of CCAM technologies, enabling efficient and informed decision-making in technology integration.
User Benefits	<p>Efficient Discovery: Quickly identify relevant CCAM technologies and suppliers.</p> <p>Informed Decisions: Access case studies, and technical documentation to make informed choices.</p> <p>Streamlined Research: Interactive knowledge map for exploring technology interconnections and industry standards.</p>
Acceptance Criteria	<p>The system should provide tailored recommendations for CCAM technologies based on the specific needs and interests of the user.</p> <p>Recommendations should include a variety of solutions such as hardware components, software platforms, and integration services.</p> <p>Provide access to a library of case studies and best practices in CCAM technology deployment.</p>
Other issues	

Table 5: User Story 2 – Service Providers

Story ID	02
Story Title	Assisting Service Providers with Targeted Recommendations for Seamless Integration in the CCAM Ecosystem.
SINFONICA Partner	ICCS
Story Actor	Service providers (e.g., Technology companies, Insurers, Cloud providers, Mobile network operators, Digital enabled platforms providers, Payment providers).
Overview	As a Service Provider involved in the CCAM ecosystem, I want reliable and actionable recommendations tailored to my industry needs, so that I can enhance safety, improve service quality, and support seamless integration within the autonomous vehicle landscape.
Triggers	<p>The ongoing shift toward connected, cooperative, and automated mobility (CCAM) means service providers need to adapt to maintain relevance, ensure compliance, and seize new business opportunities in the evolving autonomous vehicle landscape.</p> <p>New developments in autonomous vehicle technology and increased demand for reliable, secure, and user-friendly services prompt service providers to seek targeted guidance.</p>
Goal	<p>Service providers aim to leverage CCAM best practices to improve service offerings, ensure seamless integration, maintain compliance, and contribute to safer, more reliable autonomous vehicle networks.</p> <p>Proactively identify areas to streamline operations and improve data security, scalability, and network reliability, all tailored to the distinct needs of autonomous mobility.</p>
User Benefits	<p>Actionable recommendations allow service providers to implement CCAM-aligned practices efficiently, saving time and resources by following a clear roadmap.</p> <p>Contributing to the success of the CCAM ecosystem ultimately strengthens service providers' market positions and enhances the overall societal benefits of autonomous mobility solutions.</p>
Acceptance Criteria	<p>Recommendations should consider the specific role of each type of service provider.</p> <p>Each recommendation provides clear, actionable examples, making it easy to implement.</p> <p>Recommendations come with suggested metrics or KPIs to help service providers assess the impact of their actions in the CCAM ecosystem.</p>
Other issues	

Table 6: User Story 3 - Transport Operators

Story ID	03
Story Title	Enhancing Road Safety and Operational Efficiency with CCAM.
SINFONICA Partner	RE:LAB
Story Actor	Transport/Mobility Operators (e.g., Road Authorities, Transport authorities, Road operators)
Overview	This story focuses on how transport and mobility operators can use the Knowledge Map Explorer to enhance road safety and improve operational efficiency within the CCAM system.
Triggers	<ul style="list-style-type: none"> Increasing road accidents and the need to improve transport safety. Pressure to optimize operations and reduce costs. Need for resilient infrastructure to handle unforeseen events such as natural disasters and major accidents.
Goal	<ul style="list-style-type: none"> Implement technological solutions to reduce the number of road accidents. Improve traffic management and operational efficiency. Strengthen the resilience of transport infrastructure against unforeseen events. Collect and analyse data related to road safety and operational efficiency.
User Benefits	<ul style="list-style-type: none"> Reduce the number of road accidents and improve overall transport safety. Optimize transport operations, reducing downtime and operational costs. Increase infrastructure resilience, minimizing the impact of unforeseen events. Access tools and resources to monitor and improve safety and operational efficiency.
Acceptance Criteria	<p>Data Collection and Analysis:</p> <ul style="list-style-type: none"> The tool must enable the collection of data related to road accidents and operational efficiency. It should offer analytical tools to interpret and visualize this data effectively. <p>User Interface and Experience:</p> <ul style="list-style-type: none"> The tool must have an intuitive interface accessible to users with varying technical skills. It should comply with accessibility standards to ensure usability by operators with disabilities.

	<p>Knowledge Dissemination:</p> <ul style="list-style-type: none"> The tool must provide features for generating reports and presentations on road safety and operational efficiency. It should support sharing guidelines and best practices with other operators and stakeholders. <p>Interdisciplinary Collaboration:</p> <ul style="list-style-type: none"> The tool must facilitate collaboration with other organizations and interest groups to enhance safety and efficiency. It should include features for networking and sharing experiences and solutions.
Other issues	<ul style="list-style-type: none"> Ensure the tool complies with data privacy regulations and secures sensitive information. Provide training and support to help users maximize the tool's potential. Design the tool to handle growing amounts of data and increasing numbers of users over time. Ensure the tool can integrate with existing transport management systems.

Table 7: User Story 4 - Public Administration

Story ID	04
Story Title	Public Administration Seeks Best Practices for CCAM Deployment.
SINFONICA Partner	ICCS
Story Actor	Public Administration (e.g., Municipalities, Provinces, Regions, Local Transport Authorities)
Overview	A recommendation system that provides best practices and tailored recommendations on how to effectively deploy Cooperative, Connected, and Automated Mobility (CCAM) solutions.
Triggers	I want to know what mobility needs are there that are not met. And, if possible, to be met by CCAM solutions. Vulnerable users.
Goal	To be informed about the current status of the region of interest. Extract information which could easily be converted into policies. Optimize our transport infrastructure, improve safety, reduce traffic congestion, and enhance the overall efficiency of our public transportation system.

User Benefits	<p>To increase the chances that mobility needs are met emphasizing vulnerable citizens.</p> <p>Streamlines the process of identifying and implementing CCAM solutions.</p> <p>Provides tailored recommendations that consider the unique needs and constraints of different regions.</p> <p>Offers evidence-based best practices and case studies to support decision-making.</p>
Acceptance Criteria	The system should provide detailed, actionable, and context-specific recommendations.
Other issues	

Table 8: User Story 5 - Research Sector

Story ID	05
Story Title	Bridging the Gap: Enhancing Inclusivity and Accessibility in CCAM through Research.
SINFONICA Partner	UNIMORE
Story Actor	Research Sector (e.g., Universities, Research Centres, R&I Departments).
Overview	This user story focuses on creating a knowledge tool that bridges the gap between technical research on CCAM and its social innovation potential, aiming at developing a comprehensive understanding of the inclusivity and accessibility aspects of CCAM.
Triggers	<ul style="list-style-type: none"> • A growing demand for inclusive and accessible transportation solutions. • The need to integrate social innovation perspectives with technical advancements in CCAM. • Funding opportunities or policy directives emphasizing the importance of inclusivity and accessibility in CCAM research.
Goal	<ul style="list-style-type: none"> • Gather and analyse data on the inclusivity and accessibility of CCAM. • Understand the social impact of CCAM technologies. • Disseminate findings to stakeholders, including policymakers, industry players, and the public. • Collaborate with other researchers and institutions to foster inclusive and accessible CCAM solutions.
User Benefits	<ul style="list-style-type: none"> • Gain in-depth insights into the intersection of technology and social innovation in CCAM.

	<ul style="list-style-type: none"> • Make well-informed decisions on research directions and policy recommendations. • Facilitate collaboration with other researchers, industry stakeholders, and policymakers. • Contribute to the development of inclusive and accessible transportation systems, benefiting a wide range of users, including those with mobility challenges.
<p>Acceptance Criteria</p>	<p>Data collection and analysis:</p> <ul style="list-style-type: none"> • The tool must enable the collection of diverse data sources related to CCAM's inclusivity and accessibility. • It should offer analytical tools to interpret and visualize this data effectively. <p>User Interface and experience:</p> <ul style="list-style-type: none"> • The tool must have an intuitive interface accessible to users with varying technical backgrounds. • It should comply with accessibility standards to ensure usability by researchers with disabilities. <p>Knowledge dissemination:</p> <ul style="list-style-type: none"> • The tool must provide features for generating reports and presentations. • It should support collaboration and sharing of research findings within the academic community and beyond. <p>Interdisciplinary collaboration:</p> <ul style="list-style-type: none"> • The tool must facilitate connections between technical research and social innovation aspects of CCAM. • It should include features for networking and collaborating with other researchers and stakeholders.
<p>Other issues</p>	<ul style="list-style-type: none"> • Ensure the tool complies with data privacy regulations and secures sensitive information. • Consider the availability of funding and resources for the development and maintenance of the tool. • Provide training materials and support to help users maximize the tool's potential. • Design the tool to accommodate growing amounts of data and an increasing number of users over time. • Ensure the tool can integrate with existing research databases and collaboration platforms.

Table 9: User Story 6 - Legislators

Story ID	06
Story Title	Legislators Looking for Legal Recommendations.
SINFONICA Partner	ERTICO
Story Actor	Legislators (e.g., Local and National Regulators), Standards Bodies, Policy makers.
Overview	A recommendation system that provides examples of suitable legislation and legal requirements regarding public and shared transport accessibility, including effects of CCAM.
Triggers	Developing legislation, policy or standards related to Automated Public Transport Services.
Goal	Understand the topics / areas that should be included in future legislation, standards and best practice. What have other regions / organisations included in their documents.
User Benefits	Allow better and faster legislation for automated mobility, based on good practice and experiences from other countries/ jurisdictions, benefitting the industry needing legislation to take part. The needs of passengers of automated mobility can be considered in legislation.
Acceptance Criteria	Information relevant to legal / policy / regulation and standards can be found quickly. Ability to find references to other legal instruments, standards and regulations in other countries.
Other issues	

Table 10: User Story 7 - Representative Bodies

Story ID	07
Story Title	Supporting Representative Bodies with Targeted CCAM Insights for Effective Advocacy and Member Empowerment.
SINFONICA Partner	ICCS
Story Actor	Representative bodies (e.g., Automobile Associations, Trade Associations, Technology cluster etc.)

Overview	Insightful recommendations and best practices to inform members of relevant developments and support collaborative growth in the autonomous vehicle landscape.
Triggers	The rapid growth in connected and automated mobility introduces new challenges.
Goal	Seek to advocating for members’ interests while promoting industry growth, collaboration, and compliance with CCAM standards.
User Benefits	Equipped with relevant recommendations, representative bodies can advocate for their members' needs. Staying updated on CCAM advancements and reduce knowledge gaps across member organizations.
Acceptance Criteria	Recommendations include updates on relevant policies, evolving industry standards, and compliance guidelines for representatives to share with members. Recommendations outline collaboration methods with other stakeholders (e.g., tech companies, service providers, public entities) to encourage unified efforts across the ecosystem.
Other issues	

Table 11: User Story 8 - Citizens

Story ID	08
Story Title	(Potential) Users Searching for Orientation about CCAM.
SINFONICA Partner	TUD
Story Actor	Citizens (e.g., Vulnerable Users, Commuters, People with mobility challenges, Citizens from Rural or Peripheral Areas, etc.)
Overview	This story addresses the user perspective. Future users, especially people with mobility challenges, need to know how CCAM can contribute to fulfilling their mobility needs. What benefits are to be expected in comparison to current public transport solutions?
Triggers	Mobility needs (arising from different mobility challenges, i.e., physical or cognitive abilities, skill level, socioeconomic characteristics, and limited transport options) are not yet fully addressed by current public transport services.
Goal	<ul style="list-style-type: none"> Understanding what CCAM is (How does it work? What is new? What is similar to current public transport? Are there already examples?)

	<ul style="list-style-type: none"> • What are the benefits for different users, especially for different groups of people with mobility challenges? • How does CCAM facilitate Availability, Accessibility, Affordability, and Acceptability?
User Benefits	Allowing users to develop an understanding of CCAM as a prerequisite to accept and utilize CCAM services and subsequently benefit from it directly in the future.
Acceptance Criteria	Information (see goals) on CCAM is presented shortly and concisely and can be easily found.
Other issues	Accessibility of the information might be important as some groups of people with mobility challenges might also experience challenges in using a digital knowledge map.

Table 12: User Story 9 - Large Scale Demonstration projects

Story ID	09
Story Title	Large-Scale Demonstrators Seek Guidance.
SINFONICA Partner	ERTICO
Story Actor	Large-Scale Demonstration projects.
Overview	Large-Scale Demonstrators searching for knowledge.
Triggers	Organisations involved in large-scale demonstration projects (Coordinator, operators, researchers) looking for advice on how to successfully deploy their project.
Goal	Learn from SINFONICA findings, apply recommendations to their own project. Find relevant examples and best practice in toolkit. Consider accessibility and inclusion issues that might apply to the project.
User Benefits	Improve the operation of large-scale demonstration projects. Allow people with inclusivity / accessibility needs to access vehicles and services provided in large-scale demonstrators.
Acceptance Criteria	Information relevant to large scale demonstration projects can be found easily.
Other issues	

4.1.3 Requirement Analysis and Specifications

The requirement analysis is comprised of functional and non-functional requirements, architectural specifications, and technical requirements (IEEE, 1998). These specifications will be further improved during task 4.4 activities, while architectural specifications will be explained on the next chapters.

1. Functional Requirements

Functional requirements describe the core capabilities and features that the system must deliver. For a rule-based recommendation engine with ontologies, these include:

Table 13: Functional Requirements

Type of Functional Requirements	Comments
User Profiles Management	<p>Preferences: Users can explicitly set preferences such as categories, or attributes.</p> <p>User Classification: The system should classify users based on their preferences and map them to relevant segments in the ontology.</p>
Item Management	<p>Item Relationships: Manage relationships between items (e.g., “belongs to ,” “Equivalence to ”).</p>
Ontology-Based Recommendation	<p>Ontology Creation: Define a knowledge base (ontology) with well-structured relationships between concepts, such as users, items, and attributes.</p> <p>Reasoning & Inference: Use reasoning mechanisms to infer recommendations based on rules and ontology relationships.</p>
Rule-Based Engine	<p>Rule Definition: Define, manage, and update rules based on user profiles and item attributes.</p> <p>Rule Execution: Apply rules to infer which items to recommend to the user.</p>
Real-Time Recommendations	<p>Provide recommendations in real-time, based on user behaviour, preferences, and rule execution.</p>

2. Non-Functional Requirements

Non-functional requirements define the performance, security, and reliability characteristics of the system.

Table 14: Non-Functional Requirements

Type of Non-Functional Requirements	Comments
Scalability	<p>The system should handle growing user bases and a large volume of item metadata.</p>

Performance	<p>Latency: Recommendations must be generated within a reasonable time frame.</p> <p>The system must be capable of handling a large number of recommendation requests simultaneously.</p>
Maintainability	<p>The rule-based engine should allow easy rule modifications and updates.</p> <p>Updating the ontology structure should not break the system’s functionality.</p>
Security	<p>User Data Protection: Since the KME is not expected to ask the user to register or track any information about the activity of the user, this functionality was not considered.</p>
Extensibility	<p>The system should be easily extendable to add new recommendation rules, modify ontologies, or integrate new data sources as needed.</p>
Availability and Fault Tolerance	<p>Availability: Ensure the system has high availability with minimal downtime.</p>

4.2 Implementation of the Rule-Based System

The environment used to create rules is the Protégé and the Semantic Web Rule Language (SWRL) as the languages to express the Rules. For each recommendation, one or more rules, assigned into one or more use cases. The use cases are certain outputs of the recommendation engine based on certain criteria insert from the end user. The Pellet reasoner was used to run the actual rules, which as a result created the inferred RDF files, which are later uploaded to the Fuseki server. Finally, the SPARQL is used for creating queries that are then used in the API endpoints, depending on the parameters added from the end user. An image indicating the creation of a rule-based recommendation is illustrated below.

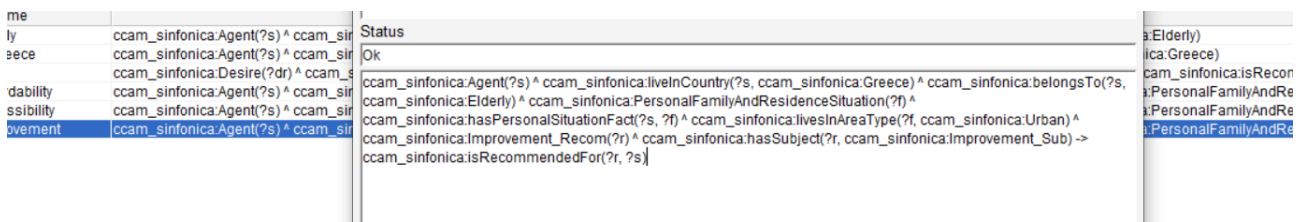


Figure 7: Example of a Rule

This rule says that as many guidelines have improvement as a theme, should be recommended for citizens who live in Greece, are elderly and live in an urban area.

Using Semantic Web Rule Language (SWRL), we established a rule-based system that leverages ontologies and semantic reasoning to enhance decision-making and recommendation accuracy. The SWRL enables the definition of complex rules in the form of “if-then” statements that operate with



the ontology schema. SWRL is designed to cooperate with OWL thus allowing both class-based logic as attribute specific conditions into the rule set.

In order to apply the SWRL rules the Pellet reasoner was used to infer logical outcomes based on the ontology. Pellet performs reasoning by applying SWRL rules to the ontology, uncovering also new knowledge that isn't directly stated but can be inferred. The outcome of running the Pellet reasoner is an inferred RDF file, containing enriched information.

Apache Fuseki was chosen as storage solution and querying RDF data. In order to query and retrieve recommendation SPARQL was used. An example of a SPARQL query that brings the recommendations for those who are adults live in urban areas of Greece, is the following:

```
PREFIX      test:<http://www.iccs.gr/sinfonica/ccam_sinfonica#>      PREFIX      rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
```

```
select distinct ?recom ?label
```

```
where
```

```
{
```

```
  ?p test:hasPersonalSituationFact ?c .
```

```
    ?p test:livesInCountry test:Greece .
```

```
    ?p test:belongsTo test:Elderly .
```

```
  ?c test:livesInAreaType test:Urban .
```

```
  ?recom test:isRecommendedFor ?p .
```

```
  ?recom rdfs:comment ?label .
```

```
}
```

4.3 Integration of the Components

To integrate the various components of a rule-based recommendation engine using ontologies, it is essential to have a clear architecture that allows smooth communication between different layers and systems. The architecture needs to ensure that data flows effectively between the frontend, backend services, rule engine, ontology, and databases. Here's an illustration of how the integration would look, followed by a detailed description of each component and its interaction.

The system can be conceived as a web application with two main components, which are the frontend and the back-end parts. designed as a modular architecture with the following key components:

Frontend

- A web application where users view recommendations, set preferences, and provide feedback.

Backend

- Recommendation Engine: The core of the system that handles rules and ontology-based inference.
- User Profile: Manages user data and preferences.
- Item Service: Manages items (recommendations in this case) and integrates with the ontology.
- Rule Engine: Executes rules and handles inference logic.

APIs

- REST APIs for interaction between frontend and backend (e.g., to fetch recommendations, or user preferences).

The following figure demonstrates an architecture diagram of the back-end that utilizes a combination of technologies for semantic data management and ontology-based querying.

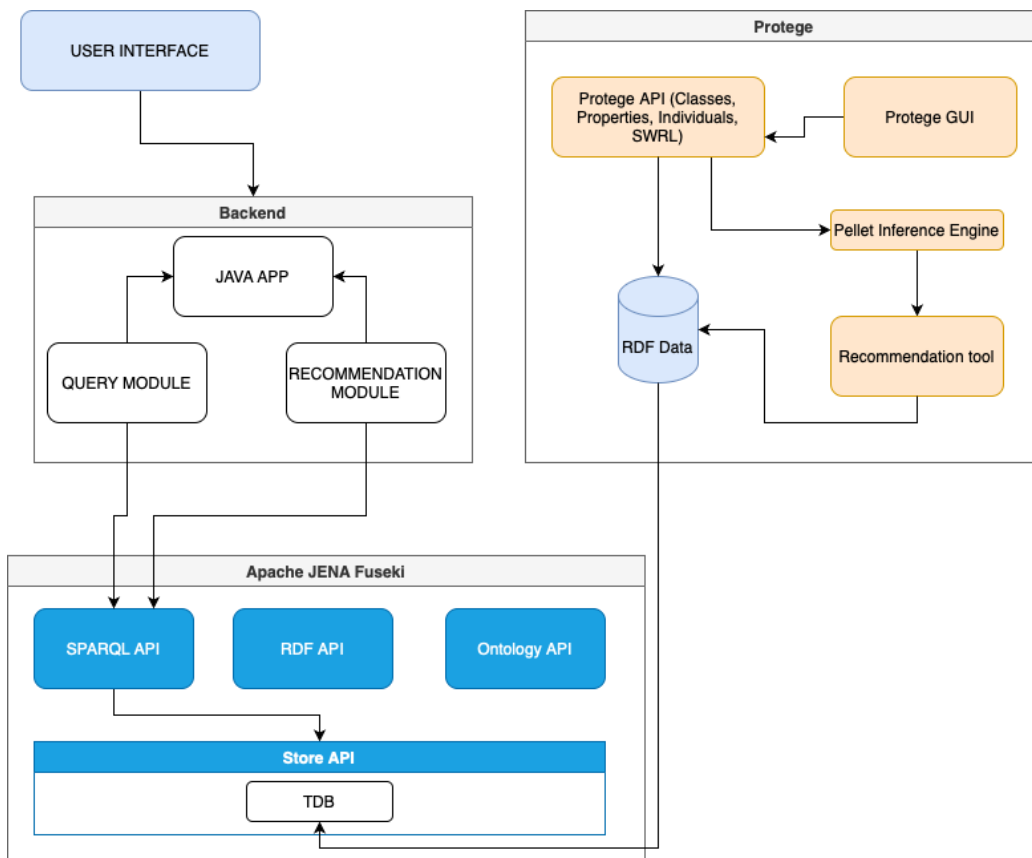


Figure 8: Architecture of the KME

4.3.1 Components of the Architecture

1. User Interface Application:

This is the front-end application where users interact with the system. It allows users to input queries and view results generated by the system.

2. Query Module:

This module is responsible for handling user queries. It processes inputs from the User Interface Application and sends them to the Fuseki Server for execution.

3. Results Module:

This module is responsible for handling and displaying the results from the queries processed by the system.

4. Fuseki Server:

Apache Jena Fuseki is a server that provides an HTTP interface for working with RDF data. It facilitates SPARQL querying over RDF datasets, connecting the Query Module with the underlying data storage and ontology framework.

5. Protégé:

Protégé is the ontology development and management tool. Within this architecture, it includes:

- Protégé API: Manages classes, properties, and individuals within the ontology.
- DB Storage: Stores the ontology data.
- Protégé GUI: A graphical user interface that allows users to visualize and edit the ontology.

6. Apache Jena:

Apache Jena is the framework for building semantic web applications. It consists of APIs and modules, such as:

- SPARQL API: Used to execute SPARQL queries over RDF data.
- RDF API: Handles RDF data manipulation and operations.
- Ontology API: Manages ontology data structures, allowing the application to interact with ontologies.
- Inference API: Provides reasoning capabilities:
 - Built-in Rule Reasoner: Allows inference using predefined or custom rules.
 - External Reasoner: Supports integration with external reasoning engines.
- Store API: Manages data storage, with two options:
 - In-memory: Stores data temporarily in memory.
 - TDB: A persistent storage solution within Jena for large RDF datasets.

The system architecture comprises several interconnected components that facilitate interaction between users and the semantic data. At the forefront is the User Interface Application, which allows users to input queries and view the generated results. When a user submits a query, the Query Module processes this input and forwards it to the Fuseki Server for execution. Leveraging Apache Jena Fuseki, the Fuseki Server acts as an HTTP interface that enables SPARQL querying over RDF datasets, effectively bridging the Query Module with the underlying data storage and ontology framework. Once the query is executed, the Results Module handles the retrieval and presentation of results back to the user.

The Recommendation Engine serves as the system's core, generating recommendations by interacting with the Rule Engine and the Ontology to produce suggestions based on predefined rules and relationships captured in the ontology. Its components include the Rule Engine (Drools/SWRL), which executes rule-based logic to infer which items to recommend by evaluating conditions based on user preferences and item attributes, and the Ontology Service (Pellet, Apache Jena), which manages the ontology and performs reasoning tasks by assessing relationships between concepts such as user preferences and item categories, thereby feeding this information into the recommendation process. When a user requests a recommendation, the Rule Engine queries the database to obtain relevant data like user preferences, demographics then applies predefined rules to infer recommendations; for example, a rule might state: "If a user identifies themselves as a CCAM deployer interested in the preferences and priorities of elderly people located in Greece who have low income and live in an urban area, recommend the following guidelines to make the CCAM system more inclusive," with the Rule Engine evaluating this condition using the ontology. Database is utilized for storing item metadata and the ontology, with components including Item Data that contains metadata and Ontology Data (RDF Store) that stores the ontology in RDF format, enabling reasoning and querying using SPARQL; the Recommendation Engine queries these databases to retrieve the necessary data for producing recommendations by accessing user data to understand preferences, item data to match these preferences with item attributes, and ontology data where the Ontology Service queries the RDF store for relationships and definitions to guide the inference process. The Rule Engine evaluates predefined if-then rules to infer recommendations, pulling information from user profiles, item metadata, and ontology relationships to make decisions, interacting with the User Profile and by obtaining user and item data and evaluating rules based on these inputs, and with the Ontology Service by utilizing the ontology to comprehend relationships.

The Protégé supports this architecture as an ontology development and management tool. Protégé includes a graphical user interface for visualizing and editing ontologies, a Protégé API for managing classes, properties, and individuals within the ontology, and a database for storing ontology data. Complementing Protégé, Apache Jena serves as the core framework for building semantic web applications within the system. It provides various APIs and modules, including the SPARQL API for executing queries, the RDF API for data manipulation, and the Ontology API for interacting with ontology structures. Apache Jena's Inference API offers reasoning capabilities through a built-in rule reasoner or integration with external reasoning engines. For data storage, it offers both in-memory storage and TDB, a persistent storage solution suitable for large RDF datasets.

4.3.2 Workflow Summary

1. Users interact with the User Interface to submit queries.



2. The front-end sends a request to the API Gateway for recommendations, including user preferences.
3. The Query Module processes the query and sends it to the Fuseki Server, which uses Apache Jena's SPARQL API to execute SPARQL queries on the data.
4. Data and ontology structure are managed through Protégé and Apache Jena components. Protégé provides ontology management tools, while Jena offers querying, inference, and storage capabilities.
5. The Results Module receives query results from the system and presents them to the user.

This architecture combines ontology management (Protégé) and semantic querying capabilities (Apache Jena) with a user-facing application, creating a robust system for ontology-based data querying and reasoning. This integration ensures that the front-end, back-end services, rule engine, and ontology service work cohesively to deliver accurate and personalized recommendations based on predefined rules and ontological reasoning.

5. Conclusions

In conclusion, this deliverable outlines the system's general architecture and associated technologies, provides theoretical background on ontologies and recommendation engines, and establishes the technical requirements and specifications for building a rule-based recommendation system using ontologies. Dependent on Deliverable 3.3 and the gathered data, the next steps involve prototyping the system, validating it against functional requirements, and creating use cases based on data from interviews, focus groups, and workshops to implement within the system.

6. References

- Ahmad, M., & Colomb, R. (2007). Overview of ontology servers research. *Webology*.
- Antoniou, G., & van Harmelen, F. (2004). *A Semantic Web Primer*. The MIT Press.
- Baker, T., Bechhofer, S., Isaac, A., Miles, A., Schreiber, G., & Summers, E. (2013). Key choices in the design of Simple Knowledge Organization System (SKOS). *Journal of Web Semantics*, 35-49.
- Cyганиак, R., Hyland-Wood, D., & Lanthaler, M. (2014). RDF1.1 Concepts and Abstract Syntax. *W3C Proposed Recommendation*.
- Farqhar, A., Fikes, R., & Rice, J. (1997). The Ontolingua server: A tool for collaborative ontology construction. *International Journal of human-computer studies*, 707-727.
- Fayyaz, Z. M. (2020). Recommendation Systems: Algorithms, Challenges, Metrics and Business Opportunities. *Applied Sciences*. doi:<https://doi.org/10.3390/app10217748>
- Genesereth, M. R., & Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers Inc.
- Gomez-Perez, A. (2004). Ontology Evaluation. In S. Staab, & R. Studer, *Handbook on Ontologies* (pp. 251-273). Springer.
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 199-220.
- Guarino, N., Oberle, D., & Staab, S. (2009). What is an ontology? In S. Staab, & R. Studer, *Handbook on Ontologies* (pp. 1-17). Springer.
- Gupta, S. M. (2023). Recommendation Engine: Challenges and Scope. *Emerging Technologies in Data Mining and Information Security. Advances in Intelligent Systems and Computing.*, 1348. Retrieved from http://doi.org/10.1007/978-981-19-4676-9_59
- IEEE. (1998, October 20). IEEE Recommended Practices for Software Requirements Specifications. *IEEE std 830*. doi:10.1109/IEEESTD.1998.88286
- Knoodl. (n.d.). Retrieved from <https://knoodl.com/ui/home.html>
- M. Marcuzzo, A. Z. (n.d.). Recommendation Systems: An insight into Current Development and Future Research Challenges. *10*, 86578-86623. doi:10.1109/ACCESS.2022.3194536
- Ma, X. M. (2024). Advancements in Recommender Systems: A comprehensive Analysis Based on Data, Algorithms, and Evaluation.

Mindswap. (2024, 10 1). *Mindswap*. Retrieved from Mindswap: <https://www.mindswap.org/2004/SWOOP>

Motik, B., & Horrocks, I. (2008). OWL Datatypes: Design and implementation. *International Semantic Web Conference* (pp. 307-322). Springer Berlin Heidelberg.

Motik, B., Patel-Schneider, P., & Parsia, B. (2012). OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). *W3C Recommendation*.

NeonToolkit. (n.d.). Retrieved 9 29, 2024, from <https://neon-toolkit.org>

Noy, N. F., & Musen, M. A. (2004). Specifying Ontology Views by Traversal. *International Semantic Web Conference* (pp. 713-725). Berlin: Springer Berlin Heidelberg.

OntoTrack. (n.d.). Retrieved from <https://www.informatik.uni-ulm.de/ki/ontotrack>

Reyes-Pena, C., & Tovar-Vidal, M. (2019). Ontology: Components and evaluation, a review. *Research in Computing Science*, 257-265.

Roy, D. D. (2022). A systematic review and research perspective on recommender systems. *Big Data* 9. Retrieved from <https://doi.org/10.1186/s40537-022-00592-5>

Saha, G. (2007). Web Ontology Language (OWL) and Semantic Web. *Ubiquity*.

Stanford. (n.d.). *Protege*. Retrieved 10 1, 2024, from Protege: <https://protege.stanford.edu/>

Tomaszuk, D., & Haudebourg, T. (2024). RDF Schema 1.2. *W3C Working Draft*.

TopBrain. (n.d.). *TopBrain Composer*. Retrieved from <https://archive.topquadrant.com/products/topbraid-composer-install/>



For more information

SINFONICA Project Coordinator

UNIMORE – University of Modena and Reggio Emilia

Via Giovanni Amendola, 2

42122 Reggio Emilia, IT

sinfonica@sinfonica.eu

www.sinfonica.eu



Funded by
the European Union

SINFONICA Project has received funding under the Horizon Europe Research and Innovation Program (Grant Agreement n° 101064988).