Advances in Human Factors of Transportation, Vol. 186, 2025, 437–447
https://doi.org/10.54941/ahfe1006533

Decision Support System to Guide User-Centric Cooperative, Connected, and Automated Mobility (CCAM) Deployments: The SINFONICA Knowledge Map Explorer

Anna Antonakopoulou, Konstantinos Fokeas, Evangelos Tsougiannis, Maria Krikochoriti, and Angelos Amditis

Institute of Communication and Computer Systems (ICCS), 9 Ir. Polytechneiou Zografou, Athens, Greece

ABSTRACT

This paper presents the SINFONICA Knowledge Map Explorer (KME), a decision-support tool designed to support the inclusive deployment of Cooperative, Connected, and Automated Mobility (CCAM) solutions. The KME consolidates data from multiple sources—interviews, focus groups, and workshops across four European contexts—to capture the needs, preferences, and concerns of diverse stakeholder groups, including vulnerable road users, transport operators, service providers, and public authorities. The resulting ontology-driven framework, implemented in Protégé using OWL (Web Ontology Language) and facilitated by the Cellfie plugin, enables the dynamic organization and visualization of CCAM-related knowledge. The KME functions as an intelligent navigation system, combining a Semantic-Based System with a Rule-Based System to provide context guidance. By harmonizing explicit and inferred knowledge, it generates tailored recommendations for CCAM solutions based on user type, scenario, and other contextual factors. This paper details the conceptual foundation, architectural specifications, and implementation roadmap for integrating the KME's semantic and rule-based components. The resulting unified solution supports stakeholders by offering evidence-based insights and best practices, fostering more equitable and sustainable CCAM deployments.

Keywords: Knowledge management system, Web ontology language (OWL), Sinfonica project, Semantic-based system, Rule-based system, Decision support system, CCAM

INTRODUCTION

The work presented is part of the SINFONICA[1] project, funded by the EU, which aims to develop effective and innovative strategies, methods, and tools to engage users, providers, and stakeholders within the Cooperative, Connected, and Automated Mobility (CCAM) ecosystem. This includes citizens, particularly vulnerable users, transport operators, public

[1] https://sinfonica.eu/

administrations, service providers, researchers, and vehicle and technology suppliers. The objective is to systematically gather, understand, and organize their needs, desires, and concerns regarding CCAM in a way that is both manageable and actionable. SINFONICA will collaboratively create decision support tools, including the Knowledge Map Explorer (KME), specifically designed for CCAM designers and policymakers to facilitate the seamless and sustainable deployment of CCAM, ensuring inclusivity and equity for all citizens. However, SINFONICA isn't meant to directly deploy, test, or operate any CCAM systems, nor process personal data. Instead, it offers methodologies, guidance, and recommendations. To facilitate this, public datasets will be utilized and gathered—for example, through focus groups and questionnaires.

This paper details the design, development, and architectural specifications of the SINFONICA KME, a tool that synthesizes and presents essential knowledge for CCAM solutions. The proposed tool utilizes an ontological structure that captures interactions and dependencies within the SINFONICA domain. The ontological framework is populated with data from interviews, focus groups, and workshops conducted in four different European contexts. For SINFONICA project needs, the Protégé tool (Gennari et al., 2003), an open-source ontology editor and knowledge management system, is employed. The transformation of structured data, like spreadsheets, into an ontology is streamlined using Cellfie, a tool within Protégé, facilitating the visualization and management of complex information. The KME is informed by data on user needs, preferences, concerns, and challenges, playing a crucial role in linking CCAM deployers, stakeholders, and users with tailored insights. The KME acts as an intelligent navigation system, offering stakeholders specialized guidance on implementing CCAM solutions based on user type, context, and scenario. It leverages ontologies and expert-driven mapping, using the formal language Web Ontology Language (OWL) (Antoniou et al., 2009) for structured representation. By employing a dual-system architecture—the Semantic-Based System and the Rule-Based System—this work specifies how the KME will recommend best practices in CCAM based on explicit and inferred knowledge. The tool provides stakeholders with accessible, domain-specific insights, supporting the development of inclusive CCAM technologies. The system architecture specification translates the SINFONICA conceptual framework into a practical implementation plan. It specifies the technological components, integration strategies, and configuration settings required to achieve the objectives of the KME. Detailed in the document, the development of the backend system begins with a requirement analysis and specifications to identify core functionalities based on end-users' needs and user stories. Following this, the development of the Semantic-Based System is explained, and the implementation of the Rule-Based System is showcased, illustrating how predefined rules govern system behavior. Finally, the integration of the components is discussed, demonstrating how the semantic and rule-based systems are cohesively combined to deliver a unified solution that meets the specified requirements. The document guides the reader through the development and integration of the KME. It begins with Requirement

Analysis and Specifications, defining the objectives and constraints of the project. The Ontology Structure chapter establishes the framework of concepts and relationships, followed by the Populate the Knowledge Map, which focuses on enriching the ontology with relevant data. The next chapter, Implementation of the Rule-Based System, explains how reasoning and decision-making mechanisms are developed using the knowledge base. Finally, Integration of the Components and the Workflow Summary describe how all elements come together to form a functional system.

REQUIREMENTS ANALYSIS AND SPECIFICATIONS

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

Table 1: Functional requirements.

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

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

Table 2: Non-functional requirements.

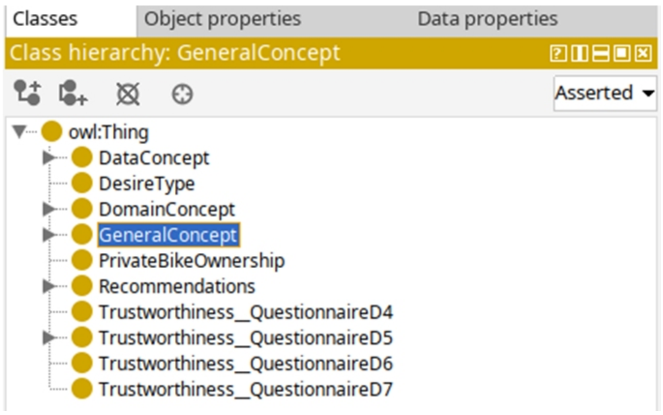
Type of Non-Functional Requirements	**Comments**
Scalability	The system should handle growing user bases and a large volume of item metadata.
Performance	Latency: Recommendations must be generated within a reasonable time frame. The system must be capable of handling a large number of recommendation requests simultaneously.
Maintainability	The rule-based engine should allow easy rule modifications and updates. Updating the ontology structure should not break the system's functionality.
Security	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.
Extensibility	The system should be easily extendable to add new recommendation rules, modify ontologies, or integrate new data sources as needed.
Availability and Fault Tolerance	Availability: Ensure the system has high availability with minimal downtime.

ONTOLOGY STRUCTURE

The ontology structure had to capture the interactions and dependencies within SINFONICA domain. After having this structure in place, the instances 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 paper, 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.



owl:Thing
- DataConcept
- DesireType
- DomainConcept
- GeneralConcept
- PrivateBikeOwnership
- Recommendations
- Trustworthiness_QuestionnaireD4
- Trustworthiness_QuestionnaireD5
- Trustworthiness_QuestionnaireD6
- Trustworthiness_QuestionnaireD7

Figure 1: Main classes of the KME as depicted on Protégé.

POPULATE THE KNOWLEDGE MAP

Populating the KME from structured data like spreadsheets into an ontology was performed using Cellfie[2], a module 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

[2]https://github.com/protegeproject/cellfie-plugin

ontological schema. The structure of the spreadsheet had to be transformed into columns, rows and headers as follows:

- **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.

Table 3: Spreadsheets transformation (indicative structure).

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 were cleaned in order to 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, mapping definitions 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. An example of a simple mapping definition is given on Table 4.

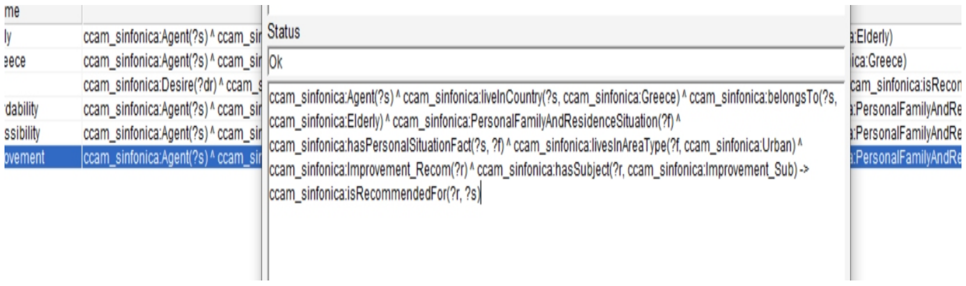
Table 4: Example of a mapping definition function.

Mapping Definition Function	Comments
Class: @A* Annotations: rdfs:label @C* SubClassOf: :hasProperty some (rdfs:label "name"^^xsd:string and :value @D*)	• @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 collected file.

IMPLEMENTATION OF THE RULE-BASED SYSTEM

The environment used to create rules is the Protégé and the Semantic Web Rule Language (SWRL) (O'Connor et al., 2005) as the language 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 (Sirin et al., 2007) was used to run the actual rules, which as a result created the inferred RDF files, which are later uploaded to the Fuseki server (Chokshi et al., 2022). Finally, the SPARQL (DuCharme et al., 2013) 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.


```
me
ly          ccam_sinfonica:Agent(?s) ^ ccam_sir      a:Elderly)
eece        ccam_sinfonica:Agent(?s) ^ ccam_sir      ica:Greece)
            ccam_sinfonica:Desire(?dr) ^ ccam_s      cam_sinfonica:isRecon
dability    ccam_sinfonica:Agent(?s) ^ ccam_sir      a:PersonalFamilyAndRe
ssibility   ccam_sinfonica:Agent(?s) ^ ccam_sir      a:PersonalFamilyAndRe
ovement     ccam_sinfonica:Agent(?s) ^ ccam_sir      a:PersonalFamilyAndRe
```

Status

Ok

```
ccam_sinfonica:Agent(?s) ^ ccam_sinfonica:liveInCountry(?s, ccam_sinfonica:Greece) ^ ccam_sinfonica:belongsTo(?s,
ccam_sinfonica:Elderly) ^ ccam_sinfonica:PersonalFamilyAndResidenceSituation(?f) ^
ccam_sinfonica:hasPersonalSituationFact(?s, ?f) ^ ccam_sinfonica:livesInAreaType(?f, ccam_sinfonica:Urban) ^
ccam_sinfonica:Improvement_Recom(?r) ^ ccam_sinfonica:hasSubject(?r, ccam_sinfonica:Improvement_Sub) ->
ccam_sinfonica:isRecommendedFor(?r, ?s)
```

Figure 2: 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.

By 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 the 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:liveInCountry test:Greece .
      ?p test:belongsTo test:Elderly .
?c test:livesInAreaType test:Urban .
?recom test:isRecommendedFor ?p .
?recom rdfs:comment ?label .
}
```

INTEGRATION OF THE COMPONENTS

To integrate the various components of a rule-based recommendation engine using ontologies, it is essential to have an architecture that allows seamless communication between the different layers. This architecture needs to ensure that data flows between the frontend, the backend services, the rule engine, the ontology, and the database. The system can be conceived as a web application (client – server) with two main components, which are the front-end and the back-end parts.

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.

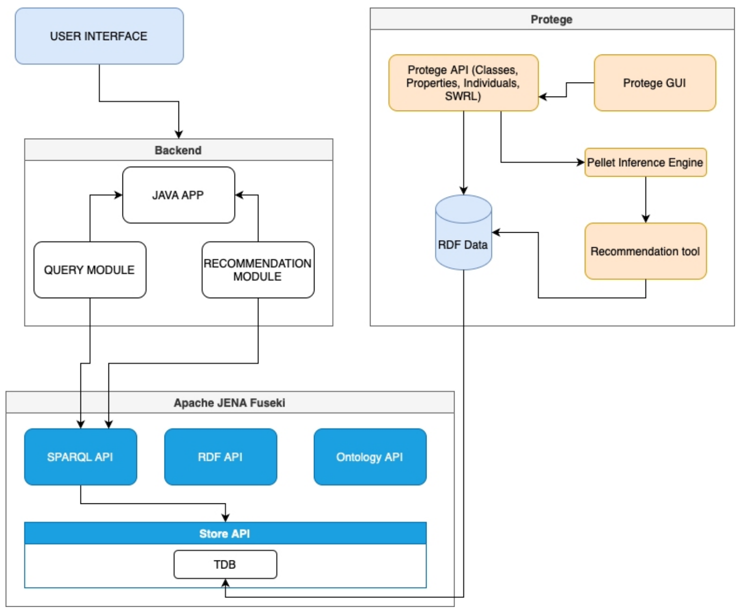


Figure 3: Architecture of the KME.

The system architecture comprises of several interconnected components that facilitate interaction between users and the semantic data. At the forefront is the User Interface Application, which allows users to insert 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. Furthermore, a Database is utilized for storing item metadata and the ontology, with subcomponents 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.

Workflow Summary

1. User Interaction: The user interacts with a user interface (UI) to select specific categories, like the type of demographic group and thus forming the relevant query. The selections are sent to the backend server as a structured request in HTTP query parameters.
2. Backend Server Receives Request: The server (Spring Boot application) parses the user’s request to extract the selected categories.
3. Query Construction: Based on the user’s selections, the backend dynamically generates a SPARQL query tailored to the user’s request.

4. SPARQL Query Execution: The backend uses the Apache Jena API to execute the generated SPARQL query against the Fuseki server.
5. Pellet Reasoning (Precomputed Inference): The dataset in Fuseki already includes data inferred using Pellet and SWRL rules, so the reasoning process has been applied before querying.
6. Response Parsing: The query results, returned in SPARQL JSON format, are parsed by the backend using Jena APIs.
7. Response to UI: The transformed results are sent back to the UI in a JSON response via HTTP.

This architecture combines ontology management (Protégé) and semantic querying capabilities (Apache Jena), 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.

CONCLUSION

This paper outlined the system's general architecture and the associated technologies used, establishing the technical requirements and specifications for building a rule-based recommendation system using ontologies. The development process emphasized the integration of ontological reasoning with a rule-based approach to ensure context-aware recommendations. A usability evaluation test was conducted to assess the system's effectiveness and user experience. The feedback gathered from this test acted as input for refining the system, ensuring it aligns with end-user needs. Future work will focus on expanding the system's capabilities to include dynamic ontology updates, where automated mechanisms will be developed to adapt and expand the ontology based on real-world data and insights generated by machine learning models. This approach will enable the system to stay current and relevant as new knowledge becomes available. Another critical direction involves semantically enriched machine learning, combining symbolic reasoning with data-driven models to enhance the accuracy and reliability of inference processes.

ACKNOWLEDGMENT

This work was supported by European Commission under the Social INnovation to FOster iNclusIve Cooperative, connected and Automated mobility (SINFONICA) Project under Grant 101064988.

REFERENCES

Antoniou, G. & van Harmelen, F. (2009). Web ontology language: OWL. *Handbook on Ontologies*, pp. 91–110.

Chokshi, H. J. & Panchal, R. (2022). Using Apache Jena Fuseki Server for Execution of SPARQL Queries in Job Search Ontology Using Semantic Technology. *International Journal of Innovative Research in Computer Science & Technology*, 10(2), pp. 497–504.

DuCharme, B. (2013). *Learning SPARQL: Querying and Updating with SPARQL 1.1*. O'Reilly Media, Inc.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., Noy, N. F., & Tu, S. W. (2003). The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1), pp. 89–123.

O'Connor, M., Knublauch, H., Tu, S. W., Grosso, W. E., & Musen, M. A. (2005). Supporting rule system interoperability on the semantic web with SWRL. *The Semantic Web–ISWC 2005: 4th International Semantic Web Conference, Galway, Ireland, November 6-10, 2005. Proceedings*, vol. 4, Springer Berlin Heidelberg, pp. 97–112.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2), pp. 51–53.