U.P.B. Sci. Bull., Series C, Vol. 87, Iss. 4, 2025 ISSN 2286 – 3540

INTEGRATED ONTOLOGY IN A WEB APPLICATION FOR KNOWLEDGE MANAGEMENT IN A UNIVERSITY

Florina Denisa CORCODEL [1], Liliana DOBRICA [2,*]

This paper presents an ontology integrated into a web application for knowledge management in university organizations. Universities face challenges in organizing large volumes of educational data, leading to fragmented information and inefficient knowledge utilization. The proposed application integrates semantic web technologies to create a structured knowledge framework that supports three user roles with personalized interfaces. Students can easily access courses, resources, deadlines, and feedback through an intuitive dashboard with calendar integration and informative cards. Professors have dedicated functionalities to add or edit courses, upload educational materials, and manage assignments or tests, with access to student activity statistics and feedback monitoring. Administrators utilize specialized dashboards for managing courses, resources, and feedback, with data export capabilities for analysis and reporting. The ontology integration enables clear information structuring and semantic relationships between data elements. The application development combines modern technologies including Django for backend development, React for frontend user interface, SQLite for database management and Owlready2 for ontology interaction, ensuring scalability and maintainability. Implementation results demonstrate efficient ontology integration within the web application, improved knowledge organization, and enhanced user experience across all roles. The application provides an efficient tool for university knowledge management that can be extended with additional functionalities, increasing the practical value of semantic technologies in educational environments.

Keywords: ontologies, knowledge management, semantic web, web application, educational technology, symbolic AI

1. Introduction

Today, turning data into useful knowledge is very important for solving various problems. People and computers work together to do this. Good knowledge helps make better decisions. Knowledge combines facts, experience, and understanding. Universities handle lots of different information. This includes course materials, research papers, and online learning data. Without good organization, universities face problems. Information gets repeated, duplicated or lost. Work becomes inefficient.

[1] MSc. Eng., Faculty of Automation and Computers Science, NUST POLITEHNICA Bucharest, Romania, e-mail: florinadenisacorcodel@gmail.com
[2] Prof., Dept. of Automation and Industrial Informatics, NUST POLITEHNICA Bucharest, Romania, *corresponding author, e-mail: liliana.dobrica@upb.ro

Knowledge Management Systems (KMSs) help solving these problems supported by knowledge engineering processes [1]. These systems collect, organize, and share knowledge. They improve teamwork, encourage innovation and support continuous learning. KMSs handle two types of knowledge. Explicit knowledge is written down and formal. Tacit knowledge comes from experience and is harder to document.

Ontologies provide a structured way to organize knowledge [2][3]. These technologies capture important concepts and their relationships. In e-learning environments specific to universities, ontologies help with personalized learning and smart resource management [18].

In this paper we present our approach to improve educational efficiency through structured knowledge representation. We describe the development of an ontology and its integration in a web application for university knowledge management. The ontology acts as the system's knowledge base. It enables smart searches and visual displays of concept relationships. The web application is designed with specific functionalities for three users' types: students, professors, and administrators. Students can access personalized resources and give feedback. Professors can manage course content and track student progress. Administrators can oversee the entire knowledge system. This approach demonstrates how ontologies can modernize university knowledge management to ensure the quality of the educational process.

2. Related work

2.1. Knowledge Management

Knowledge management focuses on identifying and using information effectively. It helps achieve goals and make informed decisions. It also improves performance across different domains [2]. Many organizations have adopted web environments based on KMSs due to their features in terms of communication, learning, sharing information, information retrieval and learning functions integration [23][24]]. In universities, where education and research are the main activities, knowledge plays a crucial role. Peter Drucker emphasized this concept in his 1993 book "Post-Capitalist Society". He stated that organizations managing knowledge will gain competitive advantages. In a knowledge-based economy, success depends on how knowledge is used. A relevant quote from his book is: "Knowledge has become the key economic resource and the dominant—and perhaps even the only—source of competitive advantage." [4]. In universities, this means utilizing effectively professors' experience. Course materials and research results become valuable assets. Students' feedback also contributes to institutional knowledge. Knowledge management transforms experience into useful practices. It helps to avoid duplicate information and encourages collaboration between departments. Knowledge can be shared between professors and students. It can be stored in documents, online platforms, or databases. An efficient KMS preserves important information [20]. It helps new members learn quickly. Good organization reduces the risk of losing valuable data. It enables rapid problem-solving. Therefore, KMS supports continuous university development. It creates a foundation for institutional growth, by enhancing both education and research activities with benefits for all stakeholders.

2.2. Knowledge Classification

A key distinction in knowledge management exists between explicit and tacit knowledge [1]. Explicit knowledge can be easily written, stored, and shared. Examples include course notes, textbooks, regulations etc. This knowledge can be stored in databases. Anyone can access it quickly when needed. Tacit knowledge is harder to express in words. It involves personal experience, practical skills, and intuition. For example, a professor can adapt a lecture for specific students. However, explaining how this is done is difficult. This knowledge transfers through discussions, observation, or direct collaboration.

Nonaka and Takeuchi proposed the SECI model to explain knowledge flow [7]. This model has four stages: socialization, externalization, combination, and internalization. Socialization involves sharing tacit knowledge through direct interaction between people. Externalization means transforming tacit knowledge into explicit knowledge. This happens by writing guides or documentation. Combination refers to organizing and assembling explicit knowledge. Internalization is the process where people learn and integrate knowledge into their experience. A practical example in the university organization consists of an experienced professor working with a new colleague. They show directly in class how to discuss with students. They demonstrate how to adapt explanations. The new colleague learns by observing and interacting. No written documents are used initially. Later, the new professor writes down what was observed. They note the steps and techniques used. Thus, tacit knowledge becomes explicit knowledge. This takes the form of guides or notes. These notes are added to other materials like manuals or regulations. All information is organized into a complete teaching guide. Other professors read the guide and apply methods to their courses. Through practice, they transform written information into personal experience. This means converting explicit knowledge back into tacit knowledge. This model of knowledge production helps universities to use both explicit and tacit knowledge better. The process supports continuous learning and collaboration between professors and students. Knowledge becomes more accessible and easier to use for all university members.

2.3. Ontologies in the university domain

Ontologies are models that help represent and organize knowledge concepts from specific domains [5][22]. They establish main concepts, concrete instances, and connections between these concepts. This makes information easier to understand and use for both humans and computers. In software engineering, ontologies provide clear data models. They enable correct processing and interpretation of information based on a logical and structured representation of knowledge. For example, in a university, domain concepts can be "Student," "Course," or "Professor." Each concept can have specific instances. Properties show characteristics or relationships, such as who teaches a course. Rules exist that establish how these concepts can be used. For example, a specific course can be taught by only one professor. Ontologies can be simple, with just organized terms. They can also be complex, with advanced rules and relationships [18].

Ontologies are essential for the semantic web. They help computers understand the meaning of data, not just store it. To build ontologies, languages like OWL, RDF, and SPARQL, that are widely used in semantic web applications are frequently considered [15,16,17]. OWL is an ontology web language to formalize concepts and relationships. RDF describes resources and connections. SPARQL allows querying and extracting data [21]. Using ontologies brings many advantages. Information can be easily shared between applications and terms are clearly defined. Systems can draw new conclusions based on relationships between concepts. Information searching becomes more efficient and relevant.

In the domain of university, ontologies can describe specific faculty activities [19] or the institution structure. They show relationships between professors, students, and administrators. They represent educational activities and resources used. This helps better manage academic knowledge. A well-known example is Gene Ontology (GO), used in biology to describe gene functions. It defines concepts like "biological process" or "molecular function." It establishes relationships between them. This helps researchers organize and compare genetic data. Many websites provide already defined ontologies, that can be imported into applications or can be used as models to build new ontologies.

2.4. CommonKADS methodology and educational platforms

Universities are seeking better methods to organize and use information. Researchers developed methods like CommonKADS [8] to build ontologies. These methods are still used today in educational settings. CommonKADS helps gather and organize knowledge of specific domains given a context defined by an organization model, a task model and an agent model. The method guides knowledge engineers to collect relevant information and develop the model set on a context level, a concept level and artefact level. The knowledge model gives a description of knowledge based on the model sets specific to a defined context. It groups concepts into categories and establishes relations between them. Ontologies have become an essential representation and artefact of a knowledge model by structuring clearly concepts and connecting data from various sources.

Educational platforms using ontologies organize teaching materials more efficiently. For example, the ALOE system (Adaptive Learning with Ontologies and Examples) personalizes learning paths [10]. It uses semantic structures to adapt educational resources to individual student needs. OpenLearn platform organizes educational resources using semantic connections [9]. The LinkedUniversities project connects educational data between different institutions. This encourages knowledge sharing at academic levels. Learning Object Metadata (LOM) is used by many universities worldwide [12]. Open University and MIT OpenCourseWare use LOM to structure digital courses [11]. SKOS organizes concepts and terms in educational systems [6]. IEEE RCD Ontology describes competencies and skills [13].

The results of the comprehensive exploration of the literature revealed that using ontologies helps universities organize better information. They connect educational resources and personalize the learning process. For example, the study described in [10] shows how an ontology-based e-learning platform can automatically recommend suitable materials. These recommendations match each student's learning style. Another example is the OpenLearn project from Open University. Here, ontologies structure courses, resources, and competencies.

The main conclusion from these studies is clear. Integrating ontologies into educational systems provides better data organization. It helps adapt content to each user's needs. Using such systems universities can offer better support for students and professors.

3. Methodology

As presented above, our research started with an exploratory study of the current approaches about knowledge management and ontologies, and their use in university organizations.

The next stage in research included an analysis of the university conceptual domain. It has been identified the main actors involved in the educational process: students, professors, and administrators. It has been analyzed essential processes like course enrollment, student evaluation, feedback provision, and resource management. To organize these concepts, the CommonKADS model set for knowledge engineering has been used. This model is recognized for its clarity in structuring knowledge. It was helpful in the clear definition of each actor's role. Also, the identification of the critical processes and how information reaches each actor have been performed. Based on this conceptual analysis of the domain information using CommonKADS, the knowledge model has been implemented by building the ontology using Protégé version 5.6.4. This is a specialized tool for creating and editing ontologies. The ontology defines main concepts in classes like Course, Student, Professor, and Process. It establishes relationships between them:

"student is enrolled in course," "professor teaches course," "course develops competencies." Also, it defines relevant properties like course name, description, difficulty level, and received feedback.

To ensure the ontology meets real user needs, in this research approach competency questions have been formulated (see Table 1). These questions are essential for testing whether the ontology structure allows desired information extraction. For example, it's important to verify if it could be quickly find which students are enrolled in a specific course. Also, it has been considered to check what resources are associated with a course, what feedback a resource received, and what competencies are developed through a study program. This testing process was iterative. It should be adjusted the ontology whenever it is discovered that certain questions cannot receive clear answers.

Table 1

Competency Questions and Semantic Relevance

No.	Competency Question	Ontological Element(s) Involved	Practical Use in Application
1	What resources are associated with a specific course?	Course, Resource, hasResource	Display resources in course page
2	Which students are enrolled in a course?	Student, Enrollment, Course	View students enrolled by professors
3	What feedback has been given for a resource?	Feedback, Resource, givenFor	Evaluate resource quality
4	What relationships exist between courses and competencies?	Course, Competence, supportsCompetence	Semantic mapping of curriculum
5	What update processes are associated with a resource?	Resource, KnowledgeUpdating, updatedBy	History of modifications and versions

The web application was developed on a three-layer architecture, including frontend, backend and database. Fig. 1 presents the architecture of the web application. It details the design decisions regarding interconnections of the main architectural elements and the implementation technologies. The backend is built with Django version 3.2.25, a robust Python framework. It manages user authentication, data saving, and semantic query processing. The frontend is built with React version 18.2.0. This provides a modern, intuitive, and easy-to-use interface. For data storage, SQLite relational database is used. This is suitable for small and medium-sized educational applications. Ontology integration into the backend is done using the Owlready2 library. This allows loading the ontology, accessing defined concepts and relationships, and connecting them with application data. For example, when a user searches for all courses, they are enrolled in, the application uses the ontology to quickly find the correct answer. This works even if data is complex or comes from different sources.

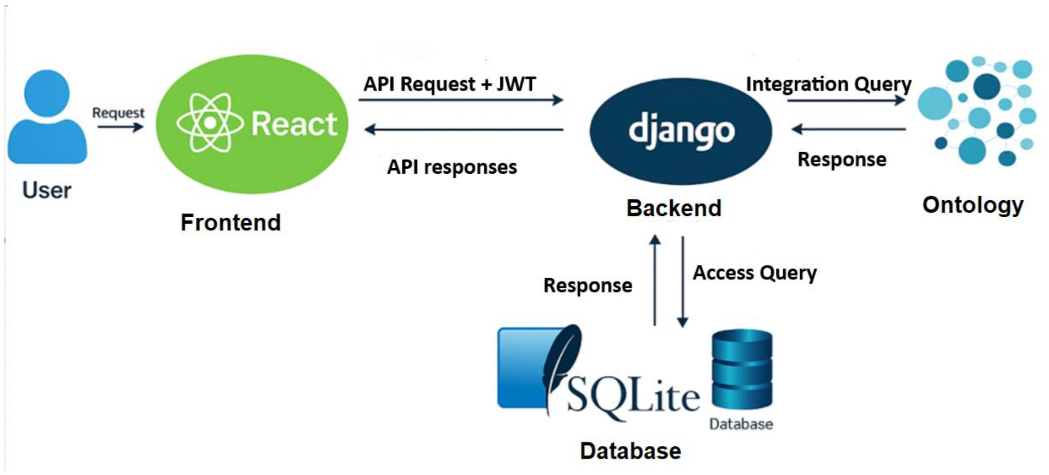


Fig. 1. Web application architecture

In the web application, user authentication is performed using JWT with Django on server and React on client. When a user logs in the React interface, authentication data is sent to the Django backend. The server verifies the data and, if correct, generates a JWT token containing user information. This token is sent back to the client and saved locally. For each subsequent request to the backend for accessing protected data, the React application attaches the JWT token in the request header. Django receives the request, extracts and verifies the JWT token. If valid, it allows access to the requested resources. Thus, users don't need to authenticate each time, and data security is ensured throughout the session.

System testing is performed on multiple levels. The validation of consistency of the ontology model in Protégé is based on using the HermiT 1.4.3.456 reasoner. This validation checks for logical errors or incorrect relationships. There are tested web application functionalities, ensuring that adding, modifying, deleting, and searching data operations work correctly. There are considered SPARQL queries to answer competency questions and simulated real university scenarios to see how users with different roles interact with the web application.

Following each stage of this research approach, it has been successfully design and implemented an ontology-based educational web application. It organizes better knowledge, provides quick access to relevant knowledge, and can be easily extended in the future.

4. Web application development

Backend development. The backend uses Django, a Python web framework. It follows the Model-View-Template (MVT) pattern. It includes:

- models.py - Defines data structure. Models for users, courses, resources, feedback, enrollments, and grades are considered. Each model has specific fields and relationships.
- views.py - Contains API logic with implemented functions for authentication, user management, courses, resources, and feedback. Also ontology queries and statistics functions are added.
- serializers.py - Converts data to JSON format. Serializers transform model data for frontend communication. To include extra information like student count per course a customization is required and feedback. Also ontology queries and statistics functions are added.
- admin.py - Configures Django admin panel by registering the main models: User, Course, Resource, Feedback, Enrollment, Grade. Display and filtering for each model are customized.
- urls.py - Defines API routes. Each route connects to a function in views.py. This allows frontend-backend communication.
- settings.py - Contains project settings. SQLite database and JWT authentication are configured. REST_framework and JWT packages are added. JWT settings include 20-minute access token lifetime, one day refresh token lifetime, and HS256 encryption.

When a user logs in, the server generates a JWT token. The client receives this token. For each request, the client sends the token in the header. The server verifies the token and allows access to resources. Users send login forms. Data is verified on the server. If correct, a session is created and users are redirected to their main page.

Frontend development. The frontend uses React for user interface and data display. It receives data from the backend. The development process includes an implementation of pages for authentication, registration, courses, resources, assignments, grades, and ontology structure. Users can send requests to the server, view charts and reports, and interact easily with all features.

- authContext.js - Located in frontend/src/context/. Manages authentication in the React application. Logic for login, logout, and registration are implemented. JWT tokens and user data are saved in localStorage. Users are redirected based on their role after authentication.
- useAxios.js - Creates a custom axios instance. It automatically handles JWT tokens for each backend request. If the access token expires, the function tries to refresh it using the refresh token. If refresh fails, the user is automatically logged out. This keeps users authenticated without interruptions and always sends the correct token to the server.
- views - Located in /frontend/src/views/. Contains JavaScript code for user dashboard pages and related pages.

- app.js - Manages all React application routes. It uses authentication context to check if users are logged in and what role they have: admin, student, or professor. Based on role, users are redirected to the appropriate dashboard and access specific pages.

Protected routes called "PrivateRoute" exist for pages requiring authentication. Public pages like login, register, home, and contact are accessible to everyone. Navigation and feature access are centrally controlled based on authentication status and user role.

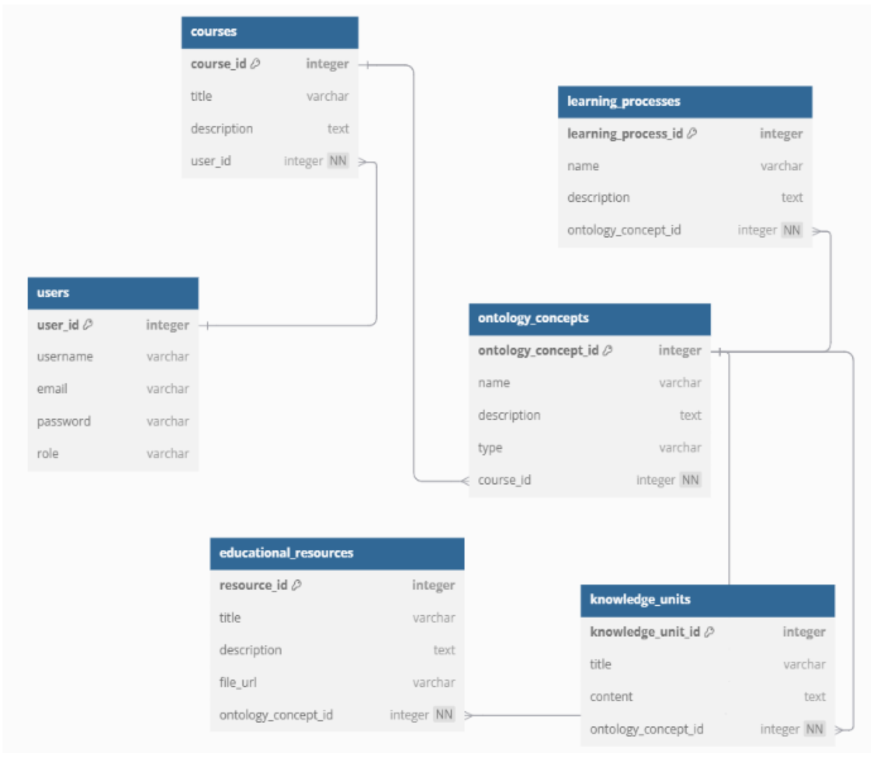


Fig. 2. Entity-Relationship Diagram

Database management. The application uses SQLite version 3.22.0, which is Django's default database. The database stores all important information: users, profiles, courses, resources, feedback, knowledge, enrollments, contact messages, ontology concepts, processes, deadlines, assignments, assignment uploads, and grades. Table structure is defined in "models.py" file. Each model corresponds to a database table. Django manages the database automatically. Migrations in the "migrations" folder allow structure updates without data loss. Data is organized logically and can be accessed quickly by the application.

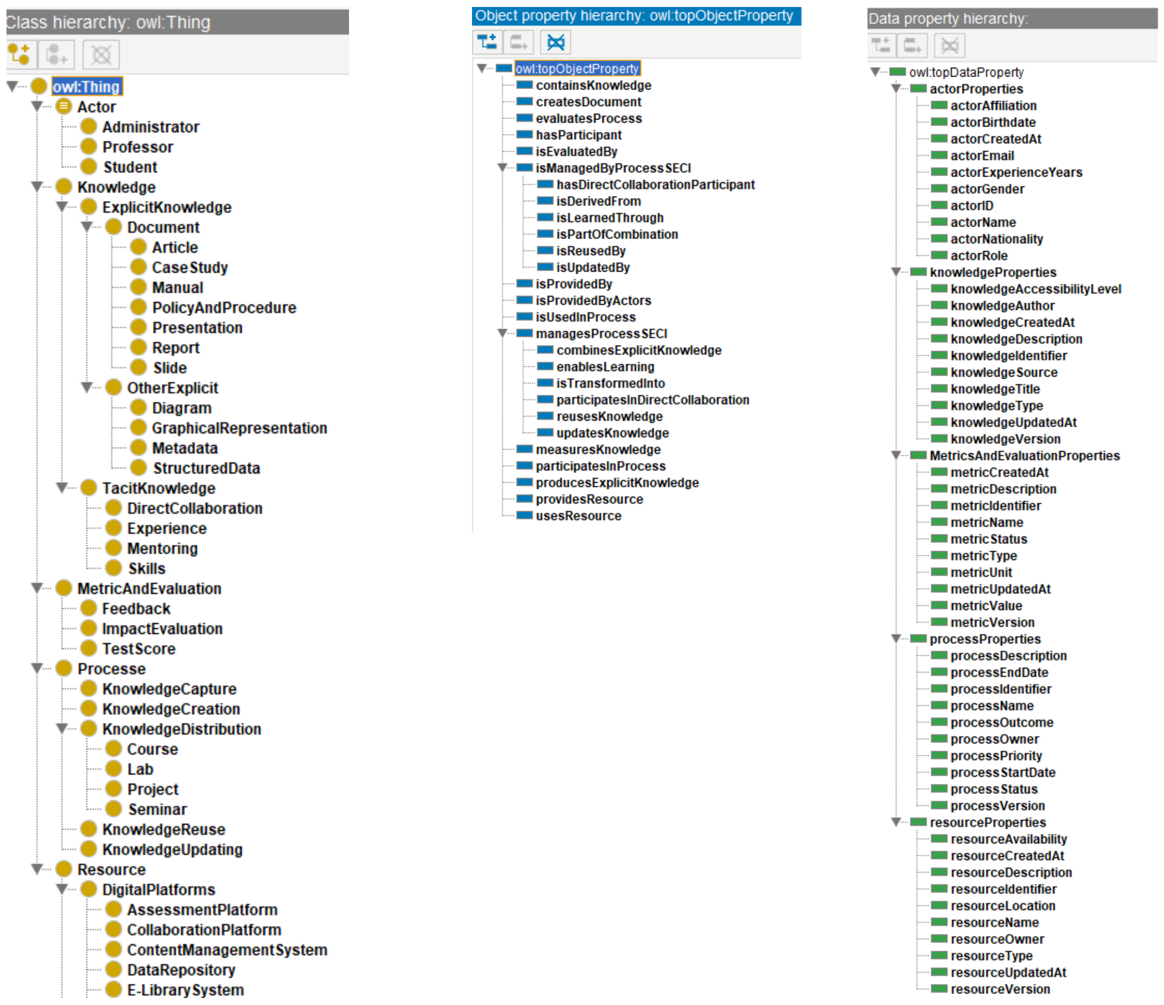
Tables and relationships in Fig. 2 were generated automatically using the Dbdiagram.io tool. Users are stored in the users table. Each course in the courses table is associated with a user. Each ontology concept of the ontology_concepts table belongs to a specific course. This helps with semantic organization of information. Educational resources, learning processes, and knowledge are directly linked to ontological concepts. This shows what topic or subject each item refers to. Table relationships allow clear data structuring and facilitate quick access to materials, processes, or knowledge relevant to each concept and course. This organization method helps efficient information management.

5. Ontology Development and integration in application

To organize university knowledge base, the CommonKADS methodology enhanced with symbolic artificial intelligence (AI) techniques are applied. Ontologies are part of symbolic AI, where knowledge is represented explicitly through formal logical structures. This approach enables logical inference, semantic reasoning, and automatic recommendations based on conceptual relationships. The system implements ontology-based recommendation algorithms, which are methods from the symbolic AI domain. These algorithms use formal knowledge structures with semantic relationships between concepts to provide intelligent suggestions to users.

The Knowledge Model structure demonstrates how symbolic AI organizes university information through explicit knowledge representation. Each node represents a formally defined concept with logical properties and relationships (see Fig. 3 -class hierarchy). The AI system performs automated reasoning to maintain consistency and infers new knowledge from existing relationships. The Actor class implements symbolic AI classification where each role, professor, student, administrator, is formally defined with logical rules. The system uses inference engines to determine appropriate roles and permissions based on semantic properties and relationships. The system implements symbolic AI algorithms for intelligent knowledge discovery, concept similarity calculation using semantic distance metrics, automated relationship detection through property analysis, intelligent query expansion using subclass reasoning, context-aware recommendations based on ontological proximity.

Object properties. Object properties establish links between ontology class instances. For example, they can connect an Actor with a learning process or a process with resulting knowledge. The *hasParticipant* property shows who participates in a process, while *producesExplicitKnowledge* indicates that a process can generate explicit knowledge, such as documents. The *participatesInDirectCollaboration* property links actors who collaborate directly, and *usesResource* shows what resources are used in a process. The *providesResource* property indicates the source of resources used. For the connection between concepts and knowledge, *containsKnowledge* shows what knowledge is associated with a concept, while *createsDocument* marks the generation of new documents. Fig. 3 details object properties hierarchy.



```
Class hierarchy: owl:Thing
owl:Thing
    Actor
        Administrator
        Professor
        Student
    Knowledge
        ExplicitKnowledge
            Document
                Article
                CaseStudy
                Manual
                PolicyAndProcedure
                Presentation
                Report
                Slide
            OtherExplicit
                Diagram
                GraphicalRepresentation
                Metadata
                StructuredData
        TacitKnowledge
            DirectCollaboration
            Experience
            Mentoring
            Skills
    MetricAndEvaluation
        Feedback
        ImpactEvaluation
        TestScore
    Processe
        KnowledgeCapture
        KnowledgeCreation
        KnowledgeDistribution
            Course
            Lab
            Project
            Seminar
        KnowledgeReuse
        KnowledgeUpdating
    Resource
        DigitalPlatforms
            AssessmentPlatform
            CollaborationPlatform
            ContentManagementSystem
            DataRepository
            E-LibrarySystem
```

```
Object property hierarchy: owl:topObjectProperty
owl:topObjectProperty
    containsKnowledge
    createsDocument
    evaluatesProcess
    hasParticipant
    isEvaluatedBy
    isManagedByProcessSECI
        hasDirectCollaborationParticipant
        isDerivedFrom
        isLearnedThrough
        isPartOfCombination
        isReusedBy
        isUpdatedBy
    isProvidedBy
    isProvidedByActors
    isUsedInProcess
    managesProcessSECI
        combinesExplicitKnowledge
        enablesLearning
        isTransformedInto
        participatesInDirectCollaboration
        reusesKnowledge
        updatesKnowledge
    measuresKnowledge
    participatesInProcess
    producesExplicitKnowledge
    providesResource
    usesResource
```

```
Data property hierarchy:
owl:topDataProperty
    actorProperties
        actorAffiliation
        actorBirthdate
        actorCreatedAt
        actorEmail
        actorExperienceYears
        actorGender
        actorID
        actorName
        actorNationality
        actorRole
    knowledgeProperties
        knowledgeAccessibilityLevel
        knowledgeAuthor
        knowledgeCreatedAt
        knowledgeDescription
        knowledgeIdentifier
        knowledgeSource
        knowledgeTitle
        knowledgeType
        knowledgeUpdatedAt
        knowledgeVersion
    MetricsAndEvaluationProperties
        metricCreatedAt
        metricDescription
        metricIdentifier
        metricName
        metricStatus
        metricType
        metricUnit
        metricUpdatedAt
        metricValue
        metricVersion
    processProperties
        processDescription
        processEndDate
        processIdentifier
        processName
        processOutcome
        processOwner
        processPriority
        processStartDate
        processStatus
        processVersion
    resourceProperties
        resourceAvailability
        resourceCreatedAt
        resourceDescription
        resourceIdentifier
        resourceLocation
        resourceName
        resourceOwner
        resourceType
        resourceUpdatedAt
        resourceVersion
```

Fig. 3. Class, Object Property and Data Property Hierarchies of the Ontology

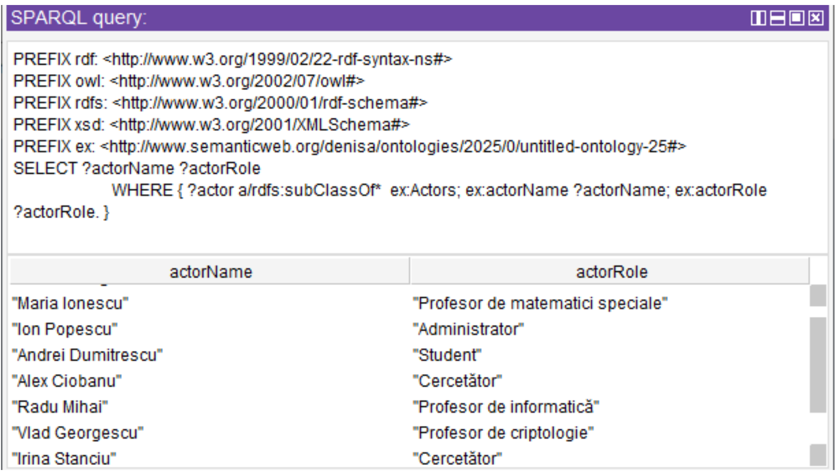
Process evaluation is done through *evaluatesProcess*, which links the evaluator to the analyzed process, and *isEvaluatedBy*, which shows who evaluates a specific process. The properties *isManagedByProcessSECI* and *managesProcessSECI* show who manages processes according to the SECI model. Actor interaction is described through *hasDirectCollaborationParticipant* and *participatesInProcess*, which show who participates in collaborations or processes. Knowledge reuse and combination are indicated by *isReusedBy* and *combinesExplicitKnowledge*. The *enablesLearning* property shows how a process contributes to learning, while *isLearnedThrough* shows how knowledge is acquired. Knowledge updating is described through *updatesKnowledge* and *isUpdatedBy*. Knowledge conversion and integration are reflected by *isTransformedInto* and *isPartOfCombination*. For performance measurement, *measuresKnowledge* links evaluation to the analyzed knowledge.

Data properties. Data properties describe attributes of ontology instances. Each entity type has specific properties (see Fig. 3 – data property hierarchy).

- Knowledge: title, description, type, author, accessibility level, unique identifier, version, and timestamps.
- Actors: name, role, demographics (birth date, gender, nationality), affiliation, unique ID, experience, email, and creation date.
- Metrics: name, description, type, identifier, unit, timestamps, status, value, etc.
- Processes: name, description, identifier, outcome, owner, priority, start/end dates, status, and version.
- Resources: name, description, identifier, availability, owner, type, location, timestamps, and version.

Each property belongs to a specific class. Identifier properties (actorID, knowledgeIdentifier, etc.) are functional, meaning that each instance has one unique ID. Cardinality constraints exist: processes have exactly one start date and version, but can have multiple participants or generate multiple knowledge items. Some properties are defined as inverse relationships.

All entities follow a structured format with name, description, unique identifier, version, and relevant metadata for easy identification and access.



SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ex: <http://www.semanticweb.org/denisa/ontologies/2025/0/untitled-ontology-25#>
SELECT ?actorName ?actorRole
        WHERE { ?actor a/rdfs:subClassOf*  ex:Actors; ex:actorName ?actorName; ex:actorRole
?actorRole. }
```

actorName	actorRole
"Maria Ionescu"	"Profesor de matematici speciale"
"Ion Popescu"	"Administrator"
"Andrei Dumitrescu"	"Student"
"Alex Ciobanu"	"Cercetător"
"Radu Mihai"	"Profesor de informatică"
"Vlad Georgescu"	"Profesor de criptologie"
"Irina Stanciu"	"Cercetător"

Fig. 4. SPARQL Query in the Ontology

Testing the ontology through execution of SPARQL queries obtains valid information such as the name and role of each actor in the system (Fig. 4). Also, the use of the reasoner to verify the logical consistency of the ontology and the correctness of the relationships defined between concepts. The results confirmed that this information can be correctly extracted, for example displaying the associated role for each person, such as professor, student, or administrator.

After validating the ontology's consistency, it has been integrated into the application using the Owlready2 library in the Django backend. The ontology file is loaded using the *get_ontology().load()* function, then traversed the defined classes to build a structure of nodes and links. This data is returned as a response to an API request, allowing the frontend to graphically show the ontology structure. Thus, the application can access and display in real-time the concepts and relationships defined in the ontology.

The Owlready2 library is used in Python, integrates well with Django, and allows manipulation and querying of ontologies directly from code. Owlready2 offers support for complex operations and SPARQL queries, making it suitable for web applications. The ontology is loaded into the application backend using the Owlready2 library. Then it traverses all classes defined in the ontology and build two lists: one with nodes, each node representing a class, and one with links (edges) that show the "is_a" type relationships between classes and their parents. For each class and parent, the name and description are added. Finally, this data is returned as a JSON response, so the frontend displays the hierarchical structure.

Once integrated into the application, the ontology is used in the backend to respond to requests from the frontend, to extract information about classes, relationships, and properties, and to generate graphical visualizations of it. The ontology is used to correlate data from the database with defined concepts, to answer complex queries, and to provide users with logically organized information directly in the application interface.

6. Conclusions

The web application can be used in practice in real-world usage scenarios and case studies that highlight how ontology integration helps education and demonstrates improved knowledge organization in universities.

These research results demonstrate that ontology-based KMSs significantly enhance information organization and retrieval in university environments through semantic technologies. The proposed web application successfully integrates recommendations with intuitive user interfaces, providing personalized access to educational resources while maintaining data consistency and improving overall user satisfaction. The system uses symbolic AI techniques, based on ontologies, to generate personalized recommendations and support semantic reasoning. Users can visualize and search information about courses, professors, and competencies. The system can be extended to data analitics reports, student progress analysis, and feedback management.

REFERENCES

[1] *L. Dobrica*, Knowledge in action towards better knowledge management in organizations, in Management Research And Practice, Vol. **13**, Iss 2, Pages 26-35, 2021

[2] *L. Dobrica*, A study on practical approaches in building domain ontologies, in Management Research and Practice, Vol. **13**, Iss. 4, Pages: 41-50, 2021

[3] *L. Dobrica*, Ontology-based tools for architecture analysis of cyber-physical systems, in Management Research And Practice, 15 (3) , pp.22-30, Sept. 2023

[4] *P. F. Drucker*, The rise of the knowledge society, unpublished manuscript, 1993. [Online]. Available: http://pinguet.free.fr/drucker93.pdf

[5] *N. F. Noy, D. L. McGuinness*, Ontology development 101: A guide to creating your first ontology, Stanford Knowledge Systems Lab., Technical Report KSL-01-05, March 2001.

[6] *A. Miles, S. Bechhofer*, SKOS simple knowledge organization system reference, W3C Recommendation, World Wide Web Consortium (W3C), Aug. 18, 2009.

[7] *I. Nonaka, H. Takeuchi H*: The Knowledge-Creating Company, Oxford University Press, 1995

[8] *G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, B. Wielinga,* Knowledge engineering and management: the CommonKADS methodology, MIT press, 2010

[9] ***, The Open University, OpenLearn – Free learning from The Open University, 2024. [Online]. Available: https://www.open.edu/openlearn (accessed August 2025)

[10] *S. Ahmed, D. Parsons, H. Ryu*, Supporting adaptive learning interactions with ontologies, in Proc. 11th ACM SIGCHI New Zealand Chapter Int. Conf. Computer-Human Interaction (CHINZ), Auckland, New Zealand, Jul. 8–9, 2010, pp. 17–24.

[11] *R. McGreal*, Learning objects:A practical definition, Int. J. Instructional Technology and Distance Learning, Vol. **2**, Iss.. 1, pp. 29, Jan. 2005.

[12] ****, IEEE Learning Technology Standards Committee, IEEE standard for Learning Object Metadata," IEEE Std 1484.12.1-2002, 2002. (accessed August 2025)

[13] ***, IEEE Learning Technology Standards Committee, Draft standard for Learning Technology—Reusable Competency Definitions," IEEE P1484.20.1/D1, Draft 1, May 2007.

[14] *D. Fensel,* Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag, Berlin, 2001

[15] ***, W3C, SPARQL 1.1 Overview, W3C Recommendation, Mar. 21, 2013.

[16] ***, W3C, OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation, Dec. 11, 2012

[17] ***, W3C, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, Feb. 25, 2014.

[18] *Y. Wang, Y. Wang*, A survey of ontologies and their applications in e-learning environments, Journal of Web Engineering, Vol. **20**, Iss. 6, pg. 1675-1720, 2021.

[19] *L. Dobrica, A. S. Cernian, A*. Bertesteanu, An Ontological Approach for the Activity in An Automatic Control Faculty, in Procs of the ICSOFT 2011 -6th International Conference on Software and Data Technologies, Vol. **2**, pg. 423-427, 2011.

[20] *A Balderas, J. A. Calallero-Hernandez, J.M. Dodero, M. Palomo-Duarte, I. Ruiz-Rube*, Model-driven skills assessment in kowledge management systems, in Journal of web engineering, Vol. **18**, Iss. 4-6, 2019.

[21] *J.-H. Lin, E. J.-L. Lu*, SPARQL generation with NMT-based approach, in Journal of web engineering, Vol. **21**, Iss. 5, 2022.

[22] *P. Chaudhary, B.B. Gupta, A.K. Singh*, Adaptive cross-site scripting attack detection framework for smart devices security using intelligent filters and attack ontology, in Soft Computing, Vol. **27**, 4593-4608, 2023.

[23] *J. Liebowitz, M. Frank*, Knowledge management and e-Learning, CRC press, 2016.

[24] *V. Opranescu, AD Ionita*, Towards a recommendation system for an educational profile in systems engineering, UPB Sci. Bull., Series C, Vol. **86**, Iss. 1, 2024.