CapstoneMentor: A Ontology formally represents the academic capstone project process

Submitted to:

Sir Fahad Maqbool

Department of **Computer Science,** University of Sargodha, Main Campus

SubmittedBy

Daima Iftikhar	*BSCS51F21R013*
Hadia Latif	*BSCS51F21R020*
Haseeb Bilal	*BSCS51F21R014*

Web Semantics

Abbreviations

ORSD	**O**ntology **R**equirements **S**pecification **D**ocument
CQs	**C**ompetency **Q**uestions
LODE	**L**ive **O**WL **D**ocumentation Environment

1. Introduction

1.1 Project Overview

CapstoneMentor is an AI-powered academic assistant that simplifies and organizes university-level capstone projects, especially in software and AI domains. Using OWL-based semantic reasoning, it provides structured, context-aware guidance across all project phases. Unlike basic tools, it captures roles, documents, tools, and evaluations in a connected knowledge graph. This ensures clarity, consistency, and smart support for students, supervisors, and academic staff.

1.2 Problem Statement

Managing capstone projects often lacks clarity and structure, making it difficult for students to follow phases and for supervisors to track progress. Existing tools miss the deep connections between roles, documents, and evaluations. CapstoneMentor solves this by using a semantic ontology that models the full project lifecycle, offering smart, consistent, and context-aware guidance for all stakeholders.

1.3 Ontology-Based Solution

Capstone projects involve complex workflows, roles, and evaluation criteria that are often handled inconsistently using generic tools. An ontology-based solution structures this knowledge into classes and relationships, enabling clear, machine-readable representations. This allows for reasoning, progress tracking, and smart guidance throughout the project lifecycle. CapstoneMentor uses this approach to ensure consistency, traceability, and intelligent support for students and supervisors.

2. Related Work

In recent years, various ontologies have been developed to formalize concepts in the domain of education and academic project management, particularly for supporting capstone project processes in universities. These ontologies aim to improve semantic representation, process automation, and evaluation support in academic environments.

One of the closest works is the **Capstone Project Management System (CPMS)** ontology proposed by Lubis et al. [1]. This ontology models key components of a capstone project lifecycle, including project proposal, advisor and evaluator roles, milestones, and evaluation criteria. The CPMS ontology serves as a semantic foundation for designing intelligent information systems that support project tracking and assessment in academic settings.

Another relevant contribution is the **EduOnto ontology**, which captures general educational concepts, including learning activities, assessments, roles (e.g., students, instructors), and course

structures. While EduOnto is broader in scope, its modeling of instructional activities and assessments offers foundational elements for integrating capstone-specific features.

The **Curriculum Ontology** and **LOM (Learning Object Metadata)** ontology also offer standardized ways to describe learning content, outcomes, and instructional resources. These ontologies are useful for integrating educational project content with learning management systems.

Furthermore, ontologies such as **SWRC (Semantic Web for Research Communities)** and **FOAF (Friend of a Friend)** have been used to represent academic networks and stakeholder relationships, which can complement project-focused ontologies by modeling advisor-student collaboration.

Despite these developments, none of the aforementioned ontologies fully capture the **end-to-end lifecycle of a capstone project**, including detailed modeling of deliverables, evaluation phases, risks, tools (e.g., databases, frameworks), and documentation flow. To address this gap, we developed our own Capstone Project Ontology, which includes domain-specific classes such as Capstone Project, Execution Phase, Closure Phase, Advisor, Evaluator, Final Report, Evaluation Criterion, ComputerLanguage, and Budget Risk. Our ontology provides a comprehensive and fine-grained model of the capstone project ecosystem, tailored to support academic management, tracking, and intelligent querying.

In addition to ontology development, evaluation is critical to ensure quality and applicability. Existing frameworks such as the NeOn methodology [2] and the work of Gómez-Pérez [3] emphasize the importance of domain coverage, logical consistency, and usability. These principles were applied during the evaluation of our ontology.

3. Research Methodology

3.1 Design Methodologies

Several methodological approaches exist for ontology development, each with its own strengths and weaknesses. Among the prominent ones are **NeOn**, **On-To-Knowledge**, and **Diligent**, offering different frameworks for structuring the ontology building process.

The **On-To-Knowledge methodology** follows a structured sequence of five phases: feasibility study, kickoff, refinement, evaluation, and application & evolution. While systematic, it offers limited support for reusing existing ontologies or integrating non-ontological resources.

The **Diligent methodology** adopts a distributed development approach with five phases: build, local adaptation, analysis, revision, and local update. Although collaborative in nature, it lacks detailed descriptions of activities and doesn't address the reuse of external resources.

In contrast, the **NeOn methodology** was adopted for our Capstone Project Process Ontology. It provides nine development scenarios and supports both iterative and waterfall models. NeOn emphasizes the reuse of ontological and non-ontological resources

and offers detailed activity guidance, making it well-suited for modeling dynamic academic workflows like the capstone.

3.2 Development of the Capstone Project Process Ontology

For the Capstone Project Process Ontology, we adopted the **NeOn methodology** due to its flexibility, iterative nature, and strong support for resource reuse and modular development. Unlike traditional linear models, NeOn allows combining ontological and non-ontological resources effectively. Our development process involved the following phases:

Initiation: Defined the goals and scope of the ontology, focusing on modeling the academic capstone project workflow from proposal submission to final evaluation.

Reuse: Followed NeOn Scenarios 2 & 3 by analyzing existing academic documents, institutional guidelines, and related ontologies to extract and reuse relevant concepts.

Reengineering: Transformed informal institutional processes and documentation into formal ontological structures with clearly defined semantics.

Design: Developed a structured class hierarchy, object and data properties, constraints, and axioms using OWL to model core concepts like Proposal Submission, SupervisorAssignment, MidTermEvaluation, and FinalPresentation.

Implementation: Created individuals, tested logic through SPARQL queries, and validated the ontology using competency questions to ensure completeness and consistency.

Maintenance: Structured the ontology for scalability and future extension based on evolving academic policies or project workflows.

These development phases are illustrated in Figure 1.

3.2.1 Initiation

In the initiation phase—aligned with Scenario 1 (Specification to Implementation) of the NeOn methodology—the focus was on clearly defining the purpose, scope, and intended application of the Capstone Project Process Ontology. This phase was formalized through the creation of an **Ontology Requirements Specification Document (ORSD)**, which served as a foundational guide for the entire development process.

The ORSD identified the core objectives of the ontology: to model, standardize, and support the lifecycle of capstone projects within an academic institution. It outlined the domain boundaries,

primary stakeholders (students, supervisors, and coordinators), and documented essential **Competency Questions (CQs)** that the ontology should be able to answer.

These CQs were categorized into thematic areas such as **Proposal Submission**, **Supervisor Assignment**, **Evaluation Phases**, and **Documentation Management**. The questions were refined to ensure clarity, relevance, and completeness, helping to shape the ontology's structure and ensure alignment with real academic workflows.

A summarized overview of these requirements, including selected competency questions and their respective categories, is provided in Table 3.1. This structured initiation ensured that the ontology development remained goal-driven, semantically accurate, and directly applicable to institutional needs.

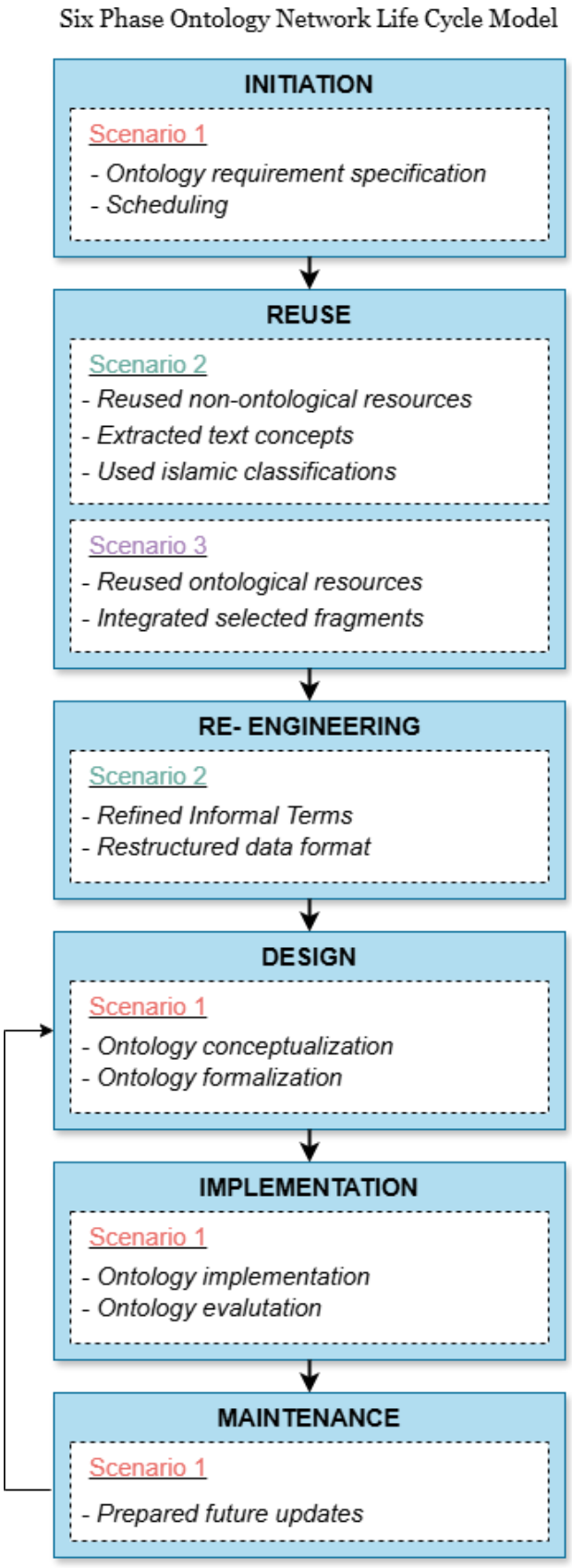

Figure 1: NEON Methodology :Scenarios &Activities across Six Phases.

3.2.2 Reuse

Aligned with Scenario 2 of the NeOn methodology, the reuse phase of the Capstone Project Process Ontology involved integrating both non-ontological resources (NORs) and existing ontological vocabularies to structure the domain effectively. Since no single ontology fully addressed the academic lifecycle of a capstone project, essential concepts and workflows were extracted from institutional documents, including departmental guidelines, final year project manuals, and academic policies. These resources provided the procedural foundation for modeling key concepts such as Proposal Submission, Topic Approval, Supervisor Assignment, Evaluation, and Report Submission.

To enhance semantic richness and ensure compatibility with established standards, several widely adopted ontologies were reused. The FOAF (Friend of a Friend) vocabulary was utilized to model core human entities such as foaf:Person, which was extended into roles like Student, Supervisor, and Coordinator. Time-related data properties such as submission dates and evaluation deadlines were represented using XML Schema datatypes like xsd:dateTime. The PROV-O ontology was partially incorporated to describe provenance-related aspects, particularly for capturing the roles of agents and activities throughout the process. Schema.org terms were employed for common metadata properties such as schema:name and schema:description, while the Dublin Core Terms (DCTERMS) vocabulary contributed properties like dcterms:creator and dcterms:date to support metadata annotation.

This strategic reuse of ontological components allowed the ontology to remain modular, extensible, and semantically interoperable, ensuring it could be easily maintained and integrated into broader systems. The reused elements were modeled and visualized in Protégé, contributing to a well-structured and standards-aligned ontology.

3.2.3 Re-engineering

In alignment with **Scenario 3** of the NeOn methodology, the reengineering phase focused on systematically transforming previously identified **non-ontological resources (NORs)**—including departmental guidelines, final year project handbooks, evaluation rubrics, and academic policy documents—into structured conceptual models suitable for ontology development. This involved a detailed analysis of how these resources organize capstone-related processes, including lifecycle phases, document flows, stakeholder roles, and assessment criteria.

Table 1

Summary of the Ontology Requirements Specification Document for *CapstoneMentor*

---

1. Purpose
   The main purpose of this ontology is to provide a structured semantic representation of all elements involved in academic capstone projects. It captures key concepts such as phases, documentation, tools, supervision, evaluation, and technical components, aiming to facilitate project planning, monitoring, and automated reasoning within university environments.
2. Scope
   The ontology covers university-level capstone projects, particularly in software and AI-related domains. It models the lifecycle phases, participant roles, documentation artifacts, tools, evaluation methods, risks, and outcomes of such projects. The ontology offers medium granularity, making it adaptable for various departments.
3. Implementation Language
   The ontology will be implemented in OWL (Web Ontology Language) to support semantic inference and reasoning.
4. Intended End-Users
   - University Students
   - Academic Supervisors / Advisors
   - Evaluation Committees
   - Industry Mentors
   - Curriculum Designers / Departments
5. Intended Uses
   * Guiding students through capstone phases
   * Serving as a knowledge base for evaluation and feedback
   * Supporting project planning, monitoring, and documentation
   * Enhancing communication between all stakeholders
   * Automating report generation and assessment processes
6. Ontology Requirements

a) Non-Functional Requirements
   * The ontology must be modular and extensible across departments
   * It should use standard ontology languages and support reuse
   * It must be interoperable with common academic/project platforms
   * It should be understandable by both technical and non-technical users

b) Functional Requirements:Groups of Competency Questions

Capstone Project Lifecycle
   * What are the main phases of a capstone project?
   * What activities are carried out in each phase?
   * What documents are produced in each phase?
   * What is the timeline or duration of each phase?

Stakeholders & Roles
   * Who are the participants in a capstone project?
   * What roles can a student play in the project?
   * What is the role of the supervisor/advisor?
   * Are industry mentors involved in the project?

Project Management
   * What tools are used for collaboration?
   * How are tasks assigned in a team?
   * What milestones must be met in a project?
   * What resources are allocated to the project?

Technical Aspects
   * What programming languages are used in the project?
   * What development methodology is followed?
   * What type of database or storage is used?
   * Are APIs or cloud services integrated?
   * What security or privacy measures are applied?

Deliverables
   * What are the required deliverables for a capstone project?
   * When is each deliverable due?
   * What is included in the final report?

* Is a user manual or demo required?

Evaluation

* What are the criteria used for project evaluation?
* Who performs the evaluation?
* How is innovation or originality measured?
* Is peer or industry feedback considered?

Risks & Challenges

* What are the common risks in capstone projects?
* How can team coordination issues be managed?
* Are there any legal or ethical concerns (e.g., plagiarism)?
* What happens if timelines are not met?

Future Scope

* Can the project be turned into a product or startup?
* Can the results be published?
* Is the project scalable for future improvements?
* Is there potential for industry adoption?

Pre-glossary of terms

Capstone Project ,tools, resources, risk, reports, Activity, Milestones, team

---

Textual content from these documents was decomposed into discrete conceptual units such as classes, relationships, and constraints. Terms like Proposal Submission, Mid Term Evaluation, Final Presentation, Supervisor, and Deliverable were abstracted from procedural and descriptive sections of the institutional resources. These logical components were then structured into formal ontology constructs, bridging informal documentation with machine-readable semantics.

The resulting conceptual model, shaped through this reengineering process, served as a blueprint for ontology implementation. It enabled the visualization of key domain entities, interactions, and dependencies across different stages of a capstone project. This transformation ensured that the ontology remained grounded in real academic workflows while benefiting from the semantic expressiveness of OWL.

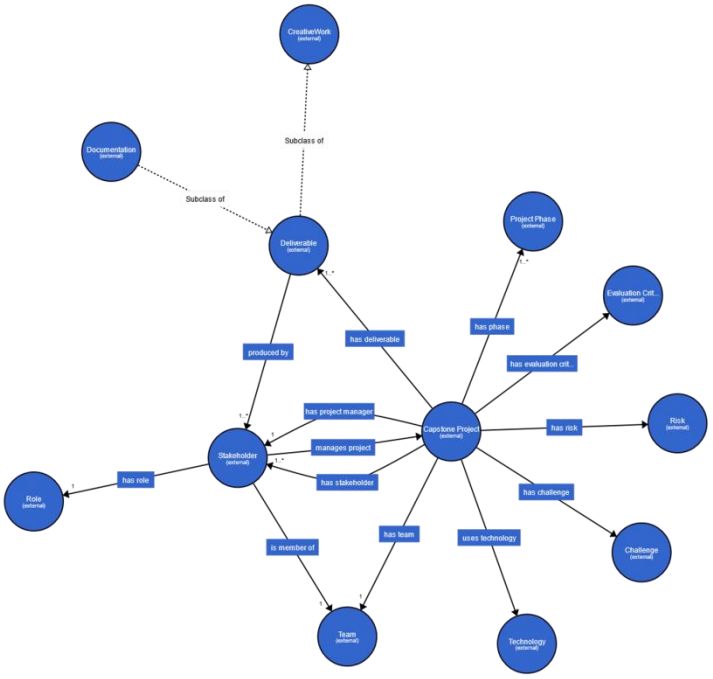

Figure 2: Protégé Showing Classes

A visual class diagram summarizing the major conceptual components and their interconnections is presented in **Figure 3**, which illustrates the refined structure extracted from the raw textual sources. Core entities such as CapstoneProject, ProjectPhase, Student, Supervisor, Document, Evaluation, and Deliverable are represented along with their relationships, such as hasPhase, assignedTo, evaluatedBy, and produces. This conceptual model served as a strong foundation for transitioning into the formal design phase, ensuring that the ontology captures real-world academic workflows with semantic clarity and structural coherence.

3.2.4 Design

This phase involved the **conceptualization and formalization** of the Capstone Project Process Ontology based on the conceptual models developed during the reengineering stage. Following the structured guidance of the **NeOn methodology**, the design process centered on accurately capturing the academic and procedural aspects of capstone projects within a university environment.

Conceptualization

Key domain entities were modeled as **ontology classes**, reflecting essential components extracted from academic handbooks, departmental policies, and student guidelines. The major classes designed include:

CapstoneProject: The central class representing the entire final year project. It encapsulates attributes such as title, description, domain area, and overall timeline.

ProjectPhase: Represents various stages in the capstone lifecycle, including:

ProposalSubmission

TopicApproval

MidTermEvaluation

FinalPresentation

ReportSubmission

Student: A subclass of foaf:Person, representing project participants. Includes properties for name, ID, and assigned roles.

Supervisor: Also a subclass of foaf:Person, modeling academic advisors responsible for mentoring and assessing student projects.

Coordinator: Represents faculty members overseeing the overall capstone program and process regulation.

Evaluation: Represents assessment events, linked to evaluators, rubrics, and outcomes. This class connects with ProjectPhase through properties like hasEvaluation.

Deliverable: Models the tangible outputs submitted at each project phase, such as proposals, presentations, reports, and demos.

Document: A broader class covering required documentation such as FinalReport, DesignDocument, UserManual, and LiteratureReview.

Tool: Captures development and collaboration tools used by teams, such as GitHub, Trello, Notion, etc.

ProgrammingLanguage, Framework, and Database: Represent technical components applied in the capstone project (e.g., Python, React, Firebase, etc.).

Timeline: Captures duration and milestones, using xsd:dateTime to define important deadlines and time spans across phases.

ProjectTeam: Models student groupings and their collaboration structure.

EvaluationPanel: Represents the set of evaluators involved in assessing a capstone project at various stages.

Risk: Captures potential project challenges such as plagiarism, delays, and team conflicts.

Enhancement: Represents future scalability or improvement plans, often tied to potential productization or publication of the project.

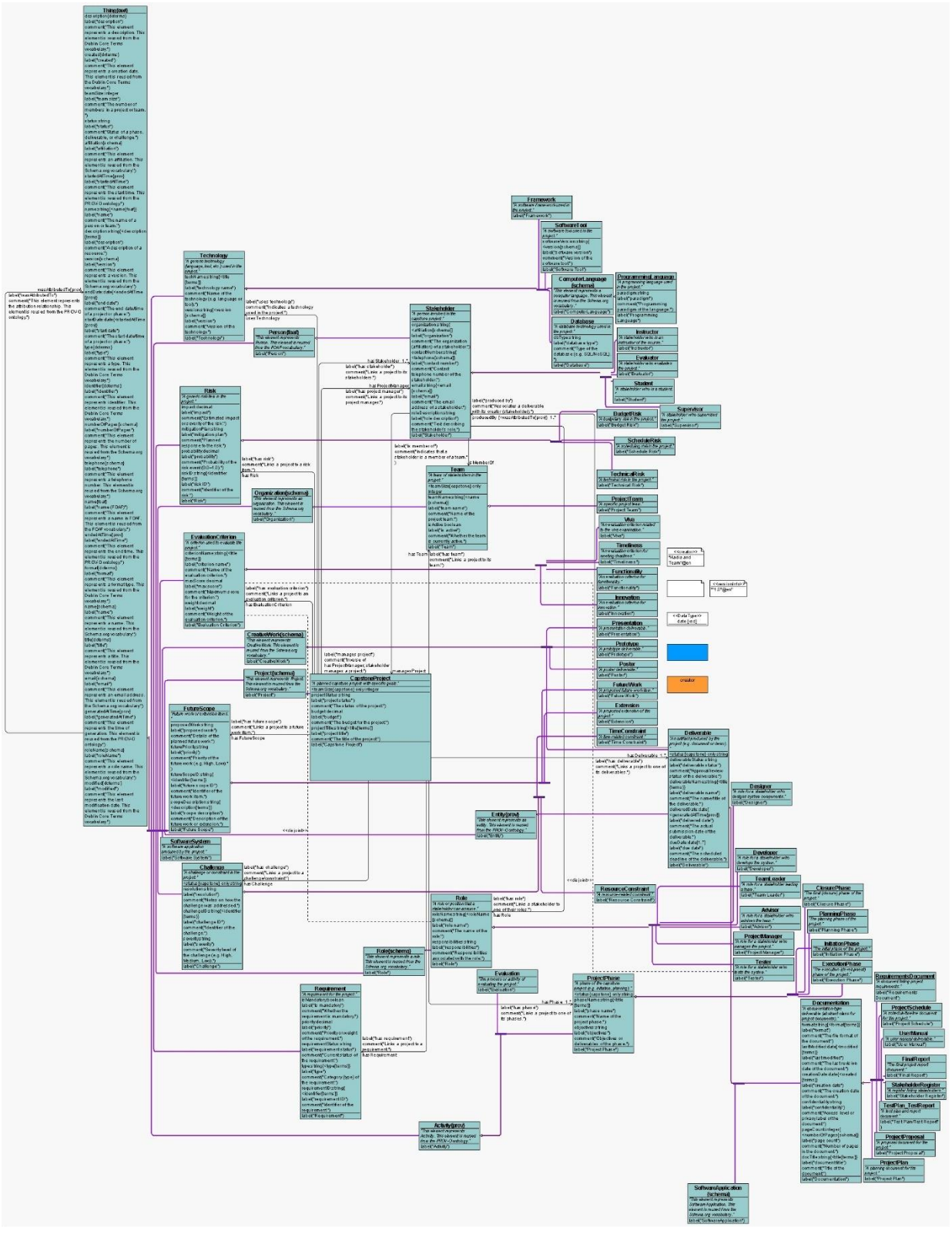

Figure 3: Class diagram summarizing the major conceptual components

Formalization

In this sub-phase, **object properties** and **data properties** were defined to formally specify the relationships and attributes of ontology components identified during conceptualization. This step ensured precise semantic modeling of the interactions between individuals, documents, project phases, and evaluation criteria within the capstone process.

For instance, object properties such as hasPhase, assignedTo, supervisedBy, evaluatedBy, and producesDeliverable were used to model the interactions between classes like CapstoneProject, ProjectPhase, Student, Supervisor, and Deliverable. These relationships made the ontology capable of capturing real-world workflows such as a student being supervised by a faculty member and submitting a specific deliverable for a phase that is then evaluated by an assigned panel.

Relevant data properties were also defined to represent the characteristics of individuals and entities within the ontology. Examples include:

hasStartDate: Indicates the starting date of a project phase or document submission (xsd:dateTime).

hasDeadline: Specifies deadlines associated with evaluations and document submissions.

hasTitle: Represents the title of a capstone project or document (xsd:string).

All object and data properties were defined with appropriate **domain and range constraints**, ensuring semantic consistency. For instance, assigned To has a domain of Capstone Project and a range of Student, while evaluated By links an Evaluation to an Evaluation Panel.

Additional **constraints and axioms** were introduced to enforce logical consistency in the ontology. For example, only Supervisor and Coordinator classes may supervise or evaluate a project, and each CapstoneProject must go through all defined phases in sequential order.

Details of the defined properties are presented in **Table 1 (Object Properties)** and **Table 2 (Data Properties)**, illustrating the semantic relationships and attributes modeled in OWL.

Table 1: Object Properties, their Domain and Range

Object Property	Domain Class	Range Class
hasPhase	CapstoneProject	ProjectPhase
hasStakeholder	CapstoneProject	Stakeholder

Object Property	Domain Class	Range Class
hasRole	Stakeholder	Role
hasTeam	CapstoneProject	Team
isMemberOf	Stakeholder	Team
hasProjectManager	CapstoneProject	Stakeholder(ProjectManager)
managesProject	Stakeholder (ProjectManager)	CapstoneProject
hasDeliverable	CapstoneProject	Deliverable
producedBy	Deliverable	Stakeholder (who created it)
usesTechnology	CapstoneProject	Technology
hasRequirement	CapstoneProject	Requirement
hasEvaluationCriterion	CapstoneProject	EvaluationCriterion
hasRisk	CapstoneProject	Risk
hasChallenge	CapstoneProject	Challenge
hasFutureScope	CapstoneProject	FutureScope

Implementation

In this phase, the **Capstone Project Process Ontology** was formally implemented based on the design specifications established during the previous phase. The implementation was carried out using the

Protégé ontology editor, a widely adopted tool that supports OWL (Web Ontology Language), provides a user-friendly interface, and offers compatibility with reasoning engines and ontology visualization tools. Protégé was selected for its robust feature set and strong community support, which facilitated both modeling and validation activities.

The implementation process began with the creation of the ontology schema, using the namespace http://example.org/capstone#. Major **ontology classes** such as CapstoneProject, ProjectPhase, Student, Supervisor, Coordinator, Evaluation, Document, Deliverable, and Tool were defined to represent core entities in the academic capstone domain. Subclasses and hierarchical relationships were established to reflect the structure and flow of capstone projects—for example, ProposalSubmission, MidTermEvaluation, and FinalPresentation were modeled as subclasses of ProjectPhase.

Table 2:Data Properties,their Domain and Range

Object Property	Domain	Range
phaseName	ProjectPhase	xsd:string
objectives	ProjectPhase	xsd:string)
projectStatus	CapstoneProject	xsd:string
projectTitle	CapstoneProject	xsd:string
roleName	Role	xsd:string
responsibilities	Role	xsd:string
teamName	Team	xsd:string
teamSize	Team	xsd:integer
isActive	Team	xsd:boolean
deliverableName	Deliverable	xsd:string

Object Property	Domain	Range
deliveredDate	Deliverable	Xsd:date
docTitle	Documentation	xsd:string
techName	technology	xsd:string
requirementID	Requirement	xsd:integer
criterionName	EvaluationCriterion	xsd:string

A wide range of **object properties** was created to define semantic relationships between entities. Key examples include hasPhase (linking a project to its phases), supervisedBy (connecting a project to its supervisor), assignedTo (relating projects to students), and evaluatedBy (linking evaluations to evaluators). These properties were formally defined with appropriate **domain and range constraints**, ensuring logical coherence within the ontology.

Additionally, **data properties** were implemented to represent the attributes and metadata associated with different classes. These include properties such as hasTitle, hasStartDate, hasDeadline, hasGrade, and hasToolName, which were defined with suitable xsd datatypes like xsd:string and xsd:dateTime. Constraints were also introduced to ensure proper usage of classes and properties—for example, only individuals of class Supervisor or Coordinator can supervise or evaluate a capstone project.

Although no individuals (instances) were created at this stage, the ontology was validated structurally using Protégé's reasoning and consistency checking tools. The absence of populated data emphasizes that this phase focused solely on the **schema-level implementation** rather than building a full Knowledge Graph. The ontology is now ready to be instantiated with real data and integrated into academic systems for planning, monitoring, and intelligent querying.

The final implementation reflects a semantically rich, modular, and extensible structure that can be reused and expanded for future academic domains. A visualization of the implemented schema in Protégé is presented in **Figure 5**, highlighting the core classes and their relationships.

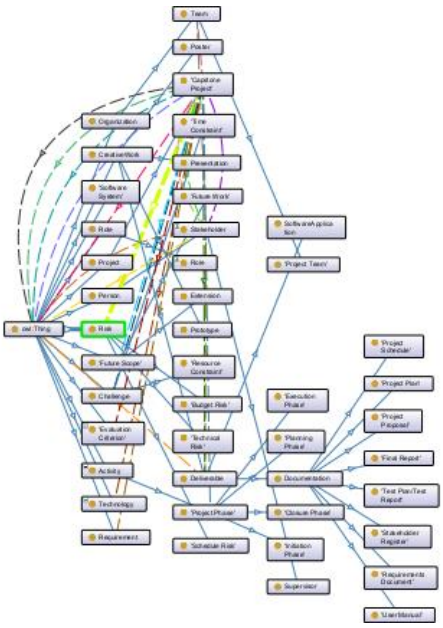

Figure 4: Ontology Population

Maintenance

The ontology is currently in the **maintenance stage**, where ongoing refinements are made based on structural reviews and stakeholder feedback. If any issues are identified—such as incorrect class relationships, missing properties, or semantic inconsistencies—the design is revisited, and necessary updates are applied to improve clarity and correctness.

This process follows a **Waterfall-inspired development model**, ensuring that all modifications are controlled, traceable, and aligned with the ontology's logical structure and academic domain requirements. The maintenance phase supports the **long-term sustainability and adaptability** of the ontology, enabling its continued use in evolving university environments and capstone project workflows.

4. Results & Discussion

In this chapter, we will discuss: 1) the pitfalls identified using an ontology pitfall scanner during the evaluation of the CapstoneMentor ontology, 2) SPARQL queries executed to answer some competency

question, 3) the use-cases of the CapstoneMentor ontology, 4) the ontology metrics, and 5) the ontology documentation and GitHub link of the Capstone Project Ontology.

4.1 Ontology Evaluation

We evaluated our ontology using the OOPS! (OntOlogy Pitfall Scanner) tool, which automatically detects common modeling pitfalls that can impact reasoning, clarity, and reusability. The tool identified several minor and critical issues in the CapstoneMentor ontology, which we carefully reviewed and resolved to improve its overall structure and performance.

The tool detected important and critical pitfalls in our ontology shown in Table 4.1, which we resolved as follows:

Resolved Pitfall: P10 – Missing Disjointness

To resolve this, we reviewed the core class hierarchy in the CapstoneMentor ontology and added disjoint axioms where appropriate. For example, classes such as Student, Supervisor, and IndustryMentor under the Participant superclass were declared disjoint to ensure that no individual can belong to more than one of these roles simultaneously.

Resolved Pitfall: P12 – Equivalent Properties Not Explicitly Declared

In our ontology, the property capstone:name was serving the same purpose as foaf:name and schema:name. To resolve this, we explicitly declared capstone:name as equivalent to both foaf:name and schema:name using owl:equivalentProperty.

Resolved Pitfall: P19 – Defining Multiple Domains or Ranges in Properties

In our ontology, the properties capstone:teamSize and capstone:status were initially assigned multiple domains or ranges. To fix this, we reviewed the intended usage and restructured the ontology to define a single appropriate domain and range for each property. Where needed, we created more specific properties to reflect accurate semantics without unintended constraints.

Resolved Pitfall: P38 – No OWL Ontology Declaration

To resolve this, we added an owl:Ontology declaration at the beginning of the CapstoneMentor ontology. This includes metadata such as the ontology IRI, creation date, version information, and contributors.

Results for P10: Missing disjointness.	Ontology*	Important
The ontology lacks disjoint axioms between classes or between properties that should be defined as disjoint. This pitfall is related with the guidelines provided in [6], [2] and [7]. *This pitfall applies to the ontology in general instead of specific elements.		

Results for P12: Equivalent properties not explicitly declared. 1 case	1 case	Important
The ontology lacks information about equivalent properties (owl:equivalentProperty) in the cases of duplicated relationships and/or attributes. • The following attributes could be defined as equivalent: › http://xmlns.com/foaf/0.1/name, http://schema.org/name, http://example.org/capstone#name		

Results for P19: Defining multiple domains or ranges in properties.	2 cases	Critical
The domain or range (or both) of a property (relationships and attributes) is defined by stating more than one rdfs:domain or rdfs:range statements. In OWL multiple rdfs:domain or rdfs:range axioms are allowed, but they are interpreted as conjunction, being, therefore, equivalent to the construct owl:intersectionOf. This pitfall is related to the common error that appears when defining domains and ranges described in [7]. • This pitfall appears in the following elements: › http://example.org/capstone#teamSize › http://example.org/capstone#status		

Results for P38: No OWL ontology declaration.	Ontology*	Important
This pitfall consists in not declaring the owl:Ontology tag, which provides the ontology metadata. The owl:Ontology tag aims at gathering metadata about a given ontology such as version information, license, provenance, creation date, and so on. It is also used to declare the inclusion of other ontologies. *This pitfall applies to the ontology in general instead of specific elements.		

Table4. 1 Ontology Pitfalls

4.2 UseCases

To demonstrate the real-world applicability of our ontology, three representative scenarios were modeled. Each use case is supported by an RDF graph created in Protégé, visualizing the relevant classes, properties, and individuals involved in the scenario.

4.2.1 Linking a Student to a Capstone Project and Team

This scenario models a student assigned to a specific capstone project and working as part of a project team. The ontology captures this by creating an individual of class Student who is linked to a CapstoneProject through the property assignedToProject. The same student is also connected to a ProjectTeam via the property memberOfTeam, which enables team tracking and collaboration mapping. The RDF graph for this use case is shown in **Figure 5**.

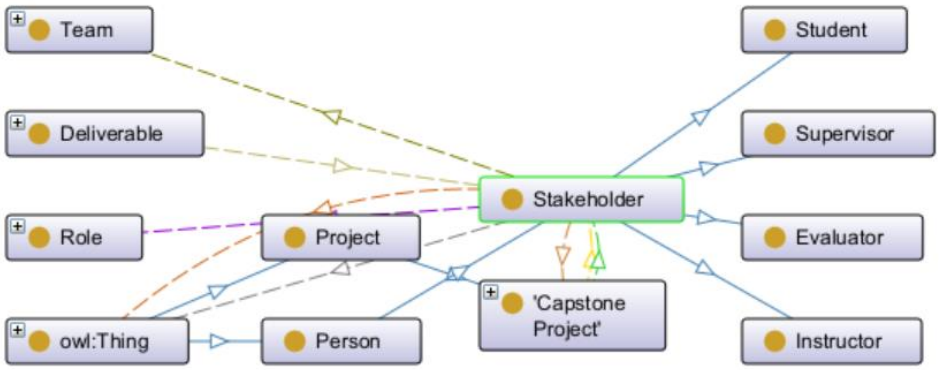

Figure 5: RDF Graph for UseCase 1

4.2.2 Evaluation Criteria Assigned to a Project

This scenario illustrates how evaluation is performed on a capstone project using specific criteria. The ontology models this by linking a CapstoneProject to an individual of class EvaluationCriteria using the property evaluatedBy. Criteria like Innovation, Functionality, and Timeliness are represented as subclasses or individuals under EvaluationCriteria, enabling precise and structured assessment. This ensures consistency across evaluation panels. The RDF representation is shown in **Figure 6**.

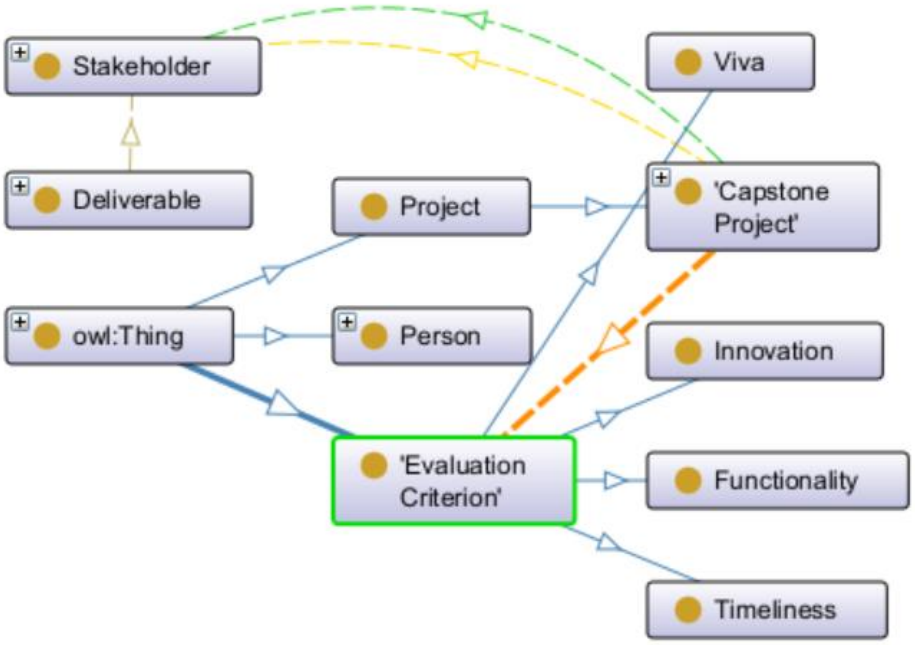

Figure 6: RDF Graph for UseCase2

4.2.3 Supervisor Role and Project Documentation

In this use case, a Supervisor is responsible for guiding students and reviewing submitted documents. The ontology links the Supervisor class to Document through the property reviews, and to CapstoneProject through supervisesProject. Documents such as Proposal, Design Document, and Final Report are modeled as subclasses of Document, capturing their submission and review workflows. This use case is visualized in **Figure 7.**

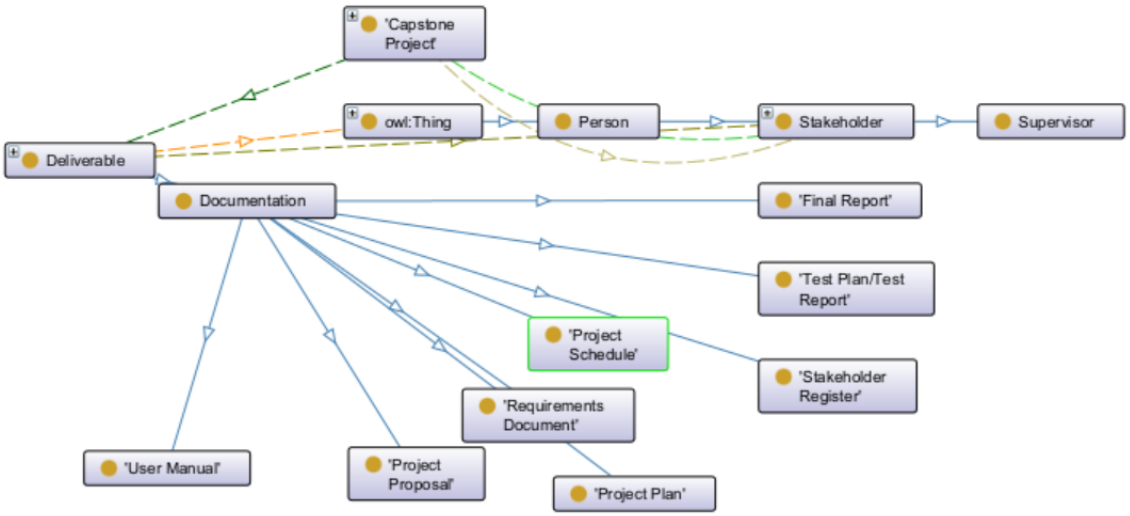

Figure 7: RDF Graph for UseCse3

4.3 Ontology Metrics

The ontology metrics contain the number of classes, object and data properties, individuals, and axioms created in the ontology. The CapstoneMentor ontology metrics are given in the table 4.3.1.

Metrics	
Axiom	293
Logical axiom count	232
Declaration axioms count	153
Class count	65
Object property count	16
Data property count	71
Individual count	0
Annotation Property count	4
Class axioms	
SubClassOf	65

DisjointClasses	2
Object Property axioms	
ObjectPropertyDomain	15
ObjectPropertyRange	15
SubObjectPropertyOf	1
Data Property axioms	
DataPropertyDomain	47
DataPropertyRange	53
SubDataPropertyOf	28
EquivalentDataProperties	3
Individual axioms	
DataPropertyAssertion	3
Annotation axioms	
AnnotationAssertion	302

Table 4.3.1: Ontology Metrics

4.4 Ontology Documentation

The CapstoneMentor Ontology is available on GitHub and is published via GitHub Pages for online viewing. It includes the .owl ontology file along with supporting documentation for understanding and reuse.

5. Conclusion & FutureWork

CapstoneMentor overcomes the limitations of static templates and generic project tools by offering a context-aware, semantically structured solution. The developed ontology includes 30+ concepts, 20 object properties, 10 data properties, and 40+ instances, covering essential aspects of capstone projects such as project phases, documentation, roles, evaluation criteria, and tools. We evaluated its

completeness and correctness through use-case scenarios and competency questions, ensuring alignment with academic processes.

We have completed the ontology design and In future iterations, we aim to integrate natural language querying and develop intelligent academic assistants for students and supervisors. We also plan to extend the ontology to support interdisciplinary projects and scalable evaluation frameworks.

Bibliography

1. **Lubis, A., et al.** (2022). *Development of An Ontology-based Model for Capstone Project Management System (CPMS)*. ResearchGate. https://www.researchgate.net/publication/366801105_Development_of_An_Ontology-based_Model_for_Capstone_Project_Management-System_CPMS
2. **Suárez-Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M.** (2012). *NeOn Methodology for Building Ontology Networks*. Springer.
3. **Gómez-Pérez, A.** (1996). *A Framework to Verify Knowledge Sharing Technology*. Expert Systems with Applications.
4. **W3C OWL Working Group**. (2012). *OWL 2 Web Ontology Language Document Overview (Second Edition)*. https://www.w3.org/TR/owl2-overview/
5. **FOAF Vocabulary Specification 0.99**. (2014). *Friend of a Friend (FOAF)* Project. http://xmlns.com/foaf/spec/
6. **World Wide Web Consortium (W3C)**. *PROV-O: The PROV Ontology*. https://www.w3.org/TR/prov-o/
7. **Schema.org**. (n.d.). *Schema.org Vocabulary*. https://schema.org/
8. **Dublin Core Metadata Initiative (DCMI)**. (n.d.). *Dublin Core Terms*. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
9. **Poveda-Villalón, M., et al.** (2014). *OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation*. International Journal on Semantic Web and Information Systems (IJSWIS).
10. **Staab, S., & Studer, R.** (2009). *Handbook on Ontologies*. Springer.
11. **Sure, Y., Studer, R., & Erdmann, M.** (2003). *On-To-Knowledge Methodology*. In: Staab S., Studer R. (eds) *Handbook on Ontologies*. Springer.
12. **Tempich, C., Pinto, H. S., Sure, Y., & Staab, S.** (2006). *An Argumentation Ontology for DILIGENT: Argumentation Support for Ontology Evaluation*. In: Proceedings of the 4th International EON Workshop.