MANTECH Publications

Journal of Data Mining and Knowledge Engineering
Volume 5, Issue 2, July-December 2020

Ontology-based Knowledge Representation for Intelligent Systems

Kailash Rao[1], jietender Nair[2]
Assistant Professor, Associate Professor
Department of Computer Science, Sunrise Institute of Technology, India
MGM College — Indore, Madhya Pradesh
Email: *Kailashrao72Gmai.com[1], jitendernair46@yahoo.com[2],*

Abstract

Ontology-based knowledge representation has emerged as a pivotal approach in the design and development of intelligent systems. By providing a structured and formal way to represent knowledge, ontologies enable machines to reason, infer, and interact with complex datasets in a human-like manner. This paper reviews the principles, methodologies, and applications of ontology-based knowledge representation in intelligent systems. It explores ontology modeling techniques, reasoning mechanisms, semantic interoperability, and integration with artificial intelligence (AI) technologies. Additionally, the paper presents comparative analyses of ontology languages, frameworks, and tools commonly used in real-world applications. The research highlights current challenges, including scalability, dynamic knowledge updates, and ontology alignment, while proposing future directions for more adaptive and context-aware intelligent systems.

Keywords: *Ontology, Knowledge Representation, Intelligent Systems, Semantic Web, Reasoning, Ontology Languages*

INTRODUCTION

Intelligent systems rely heavily on structured knowledge to perform tasks such as decision-making, natural language understanding, and autonomous problem-solving. Traditional data representation methods, such as relational databases, often fail to capture the complexity and semantics of real-world knowledge. Ontology-based knowledge representation addresses this limitation by formally defining concepts, relationships, and constraints within a domain,

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

allowing machines to interpret and reason over information more effectively.

The term **ontology** originates from philosophy, referring to the study of the nature of being and existence. In computer science, it represents an explicit formal specification of a conceptualization, providing a shared understanding of a domain. Ontologies support semantic interoperability, knowledge sharing, and reasoning, making them crucial in fields like semantic web, expert systems, robotics, and healthcare applications.

FUNDAMENTALS OF ONTOLOGY-BASED KNOWLEDGE REPRESENTATION

Ontology-based knowledge representation provides a structured and semantic way of modeling information so that machines can understand and reason about it. Unlike traditional data models that focus mainly on storing data, ontologies aim to describe the meaning of data. They define concepts, relationships, and constraints within a particular domain in a formal and explicit manner.

In intelligent systems, ontology acts as a shared conceptual model. It helps different systems and agents to communicate using a common vocabulary. The formal nature of ontologies also enables automated reasoning, which allows systems to infer new knowledge from existing facts. This makes ontology a very important foundation for semantic technologies, expert systems, and knowledge-based applications.

1. Definition and Components

An ontology can be defined as a formal, explicit specification of a shared conceptualization of a domain. It captures domain knowledge in a machine-readable format and organizes it into meaningful structures. The main components of an ontology are discussed below in detail.

Classes (Concepts)

Classes, also known as concepts, represent general categories or types of objects within a domain. They form the backbone of an ontology and usually organized in a hierarchical structure called a taxonomy.

For example, in a healthcare ontology, *Disease*, *Patient*, and *Doctor* can be considered as classes. Classes can also have subclasses. For instance:

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

- Disease
- InfectiousDisease
- ChronicDisease

This hierarchical structure allows inheritance of properties. A subclass automatically inherits the properties of its parent class. So if *Disease* has a property like "hasSymptom," then *InfectiousDisease* will also have that property.

Classes help in organizing knowledge systematically. Without well-defined classes, ontology becomes unstructured and confusing. However, defining proper classes sometimes becomes difficult when domain concepts overlap or are ambiguous.

2. Individuals (Instances)

Individuals, also known as instances, are the actual objects or entities that belong to a class. While classes describe general concepts, individuals represent real-world examples.

For example:

- Class: Disease
- Individual: Diabetes
- Class: Patient
- Individual: Patient_001

Each individual is a member of at least one class. Individuals contain specific data values and are connected to other individuals through relationships.

In intelligent systems, individuals represent factual knowledge. For example, an expert system in medical domain may store patient records as individuals and use ontology rules to analyze their symptoms.

Instances make ontology practical and usable. Without individuals, ontology remains only as abstract structure without real-world data.

3. Attributes (Properties)

Attributes, also called properties, describe characteristics of classes or individuals. They define additional information about entities in the ontology.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

Properties are generally of two types:

- **Datatype Properties:** Link an individual to a data value (such as string, integer, date).
- **Object Properties:** Link an individual to another individual.

For example:

- Datatype Property:
- hasAge → 45
- hasName → "Ravi"
- Object Property:
- treatedBy → Doctor_01
- hasSymptom → Fever

Attributes help in providing detailed information and making ontology expressive. In knowledge-based systems, properties are very important because reasoning depends on these relationships and values.

Sometimes designing properties becomes challenging because one has to carefully define domain and range constraints. If properties are not clearly defined, it may cause inconsistencies in reasoning.

4. Relationships (Relations)

Relationships define how classes or individuals interact with each other. They connect different components of ontology and give semantic meaning to the structure.

For example:

- A Patient *hasSymptom* Fever
- A Doctor *treats* Patient
- A Student *enrolledIn* Course

Relationships are usually represented as object properties. They can also have characteristics such as:

- **Transitive:** If A is related to B and B to C, then A is related to C.
- **Symmetric:** If A is related to B, then B is related to A.
- **Inverse:** If A treats B, then B is treatedBy A.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

These characteristics allow intelligent systems to perform logical reasoning. For example, if "isPartOf" is transitive, then system can automatically infer indirect relationships.
Relations provide the semantic richness of ontology. Without relationships, knowledge representation becomes flat and lacks meaning.

5. Axioms

Axioms are logical statements or rules that define constraints and ensure consistency in ontology. They express truths about the domain that must always hold.
Examples of axioms:

- Every Patient must have at least one Doctor.
- No individual can be both Male and Female at the same time.
- InfectiousDisease is a subclass of Disease.

Axioms help in validating knowledge and detecting inconsistencies. If an individual violates an axiom, the reasoning engine can identify it as an error.
For example, if a system defines that "A Minor must have age less than 18," and an instance shows age as 25, then it will create inconsistency.

Axioms are essential for reasoning and inference. They make ontology more than just a data model; they make it logically sound and machine-understandable.

Table 1. Example of Ontology Components in Healthcare Domain

Component	Example
Class	Disease, Symptom, Patient
Individual	Flu, Fever, Patient123
Attribute	Severity, OnsetTime
Relationship	hasSymptom, treatedBy
Axiom	Every patient must have at least one symptom

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

ONTOLOGY TYPES

Ontologies are generally classified according to their scope, abstraction level, and purpose of use. This classification helps researchers and developers to choose appropriate ontology type depending on application requirements. Each type has its own advantages and limitations, and sometimes more than one type is used together in intelligent systems.

1. Domain Ontology

A **Domain Ontology** represents knowledge specific to a particular subject area or discipline. It focuses on modeling concepts, relationships, and rules that are relevant only within that domain.

For example, in medical diagnosis systems, a domain ontology may include concepts such as *Disease*, *Symptom*, *Medication*, and *Treatment*. It also defines relationships like "hasSymptom," "diagnosedBy," or "treatedWith."

Domain ontologies are highly specialized and detailed. They capture expert knowledge and are usually developed with the help of domain specialists. These ontologies are very useful in applications like:

- Clinical decision support systems
- E-learning platforms
- Financial risk assessment systems
- Agriculture advisory systems

However, domain ontologies are not easily reusable outside their specific area. If the context changes, major modifications may be required. Also, creating a domain ontology takes time because it requires deep understanding of the subject matter.

2. Task Ontology

A **Task Ontology** models knowledge related to a particular task or activity rather than a subject domain. It focuses on describing processes, methods, inputs, and outputs required to perform a task.

Examples of tasks include:

- Planning

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

- Scheduling
- Diagnosis
- Monitoring
- Decision-making

For example, in a scheduling system, a task ontology may define concepts such as *Task*, *Deadline*, *Resource*, and *Priority*. It may also include rules like "High priority tasks must be executed before low priority tasks."

Task ontologies are reusable across multiple domains. For instance, a scheduling ontology can be applied in healthcare, education, or manufacturing with small modifications.

One limitation is that task ontologies may not capture detailed domain-specific knowledge. Therefore, they are often combined with domain ontologies to create more complete intelligent systems.

3. Upper Ontology

An Upper Ontology **(also called foundational ontology) provides very general and abstract concepts that are common across all domains. These ontologies describe high-level categories such as:**

- Object
- Event
- Time
- Space
- Process
- Entity

Upper ontologies act as a foundation for building domain-specific ontologies. They ensure consistency and interoperability when integrating multiple systems.

For example, concepts like *PhysicalObject* or *AbstractEntity* can be used as parent classes for more specific domain concepts. This provides a structured and logical hierarchy.

The main advantage of upper ontologies is their ability to promote semantic interoperability. Different systems built on the same upper ontology can easily share and exchange data.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

However, designing upper ontologies is complex because the concepts must be universal and applicable to many contexts. Also, sometimes they may become too abstract, making implementation difficult.

4. Hybrid Ontology

A **Hybrid Ontology** combines elements of domain, task, and upper ontologies. It integrates abstract foundational concepts with domain-specific knowledge and task-oriented structures.
Hybrid ontologies are widely used in real-world intelligent systems because they provide:

- Flexibility
- Reusability
- Improved reasoning capability
- Better interoperability

For example, a smart healthcare system may use:

- Upper ontology concepts like *Entity* and *Event*
- Domain ontology concepts like *Patient* and *Disease*
- Task ontology concepts like *DiagnosisProcess*

By integrating all these components, the system becomes more intelligent and context-aware.
The main challenge with hybrid ontologies is complexity. Managing multiple layers of abstraction can lead to maintenance difficulties. Still, they are considered powerful for advanced intelligent applications.

ONTOLOGY TYPES

Ontologies are generally classified according to their scope, abstraction level, and purpose of use. This classification helps researchers and developers to choose appropriate ontology type depending on application requirements. Each type has its own advantages and limitations, and sometimes more than one type is used together in intelligent systems.

1. Domain Ontology

A **Domain Ontology** represents knowledge specific to a particular subject area or discipline. It focuses on modeling concepts, relationships, and rules that are relevant only within that domain.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

For example, in medical diagnosis systems, a domain ontology may include concepts such as *Disease*, *Symptom*, *Medication*, and *Treatment*. It also defines relationships like "hasSymptom," "diagnosedBy," or "treatedWith."

Domain ontologies are highly specialized and detailed. They capture expert knowledge and are usually developed with the help of domain specialists. These ontologies are very useful in applications like:

- Clinical decision support systems
- E-learning platforms
- Financial risk assessment systems
- Agriculture advisory systems

However, domain ontologies are not easily reusable outside their specific area. If the context changes, major modifications may be required. Also, creating a domain ontology takes time because it requires deep understanding of the subject matter.

2. Task Ontology

A **Task Ontology** models knowledge related to a particular task or activity rather than a subject domain. It focuses on describing processes, methods, inputs, and outputs required to perform a task.

Examples of tasks include:

- Planning
- Scheduling
- Diagnosis
- Monitoring
- Decision-making

For example, in a scheduling system, a task ontology may define concepts such as *Task*, *Deadline*, *Resource*, and *Priority*. It may also include rules like "High priority tasks must be executed before low priority tasks."

Task ontologies are reusable across multiple domains. For instance, a scheduling ontology can be applied in healthcare, education, or manufacturing with small modifications.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

One limitation is that task ontologies may not capture detailed domain-specific knowledge. Therefore, they are often combined with domain ontologies to create more complete intelligent systems.

3. Upper Ontology

An **Upper Ontology** (also called foundational ontology) provides very general and abstract concepts that are common across all domains. These ontologies describe high-level categories such as:

- Object
- Event
- Time
- Space
- Process
- Entity

Upper ontologies act as a foundation for building domain-specific ontologies. They ensure consistency and interoperability when integrating multiple systems.

For example, concepts like *PhysicalObject* or *AbstractEntity* can be used as parent classes for more specific domain concepts. This provides a structured and logical hierarchy.

The main advantage of upper ontologies is their ability to promote semantic interoperability. Different systems built on the same upper ontology can easily share and exchange data.

However, designing upper ontologies is complex because the concepts must be universal and applicable to many contexts. Also, sometimes they may become too abstract, making implementation difficult.

4. Hybrid Ontology

A **Hybrid Ontology** combines elements of domain, task, and upper ontologies. It integrates abstract foundational concepts with domain-specific knowledge and task-oriented structures. Hybrid ontologies are widely used in real-world intelligent systems because they provide:

- Flexibility

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

- Reusability
- Improved reasoning capability
- Better interoperability

For example, a smart healthcare system may use:

- Upper ontology concepts like *Entity* and *Event*
- Domain ontology concepts like *Patient* and *Disease*
- Task ontology concepts like *DiagnosisProcess*

By integrating all these components, the system becomes more intelligent and context-aware. The main challenge with hybrid ontologies is complexity. Managing multiple layers of abstraction can lead to maintenance difficulties. Still, they are considered powerful for advanced intelligent applications.

Table 2: Comparison of Ontology Types

Ontology Type	Focus Area	Reusability	Example Application
Domain Ontology	Specific subject domain	Low	Medical diagnosis
Task Ontology	Specific task or activity	Moderate	Scheduling system
Upper Ontology	General abstract concepts	High	Knowledge integration
Hybrid Ontology	Combination of multiple types	High	Smart healthcare systems

ONTOLOGY LANGUAGES AND FRAMEWORKS

To implement ontologies in intelligent systems, several standard languages and development frameworks are used. These languages provide formal syntax and semantics to represent knowledge in a machine-readable form.

1. RDF (Resource Description Framework)

The **Resource Description Framework (RDF)** is a standard model for representing information on the web. It uses a simple triple structure:

Subject – Predicate – Object

For example:

Patient_01 – hasSymptom – Fever

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

This triple-based representation allows flexible data modeling and linking between entities. RDF is simple and easy to understand, but it does not provide strong support for complex hierarchies or logical constraints.

RDF is widely used in semantic web applications and linked data systems.

2. RDFS (RDF Schema)

RDF Schema (RDFS) extends RDF by adding basic vocabulary for describing classes and properties. It allows defining:

- Class hierarchies
- Subclass relationships
- Domain and range of properties

For example:

- Doctor is a subclass of Person
- hasAge has domain Person and range Integer

RDFS provides more structure compared to RDF. However, it still has limited reasoning capability and cannot express complex constraints such as cardinality restrictions.

3. OWL (Web Ontology Language)

The **Web Ontology Language (OWL)** is a more expressive ontology language designed for complex knowledge representation and reasoning. OWL is built on top of RDF and RDFS but provides advanced features such as:

- Logical class definitions
- Cardinality constraints
- Equivalent classes
- Disjoint classes
- Complex property characteristics

OWL supports automated reasoning using description logic. Intelligent systems can use reasoners to check consistency, classify concepts, and infer new relationships.

There are different versions of OWL (such as OWL Lite, OWL DL, and OWL Full), each offering varying levels of expressiveness and computational complexity.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

OWL is widely used in healthcare, bioinformatics, and enterprise knowledge management systems.

PROTÉGÉ

Protégé is a popular open-source ontology development tool. It provides a user-friendly interface for creating, editing, and visualizing ontologies.

Key features of Protégé include:

- Graphical ontology editor
- Support for RDF, RDFS, and OWL
- Integration with reasoning engines
- Visualization plugins
- Query support using SPARQL

Protégé makes ontology modeling easier, especially for researchers and students. It reduces the complexity of manual coding and helps in validating ontology structure.

However, for very large ontologies, performance issues may arise.

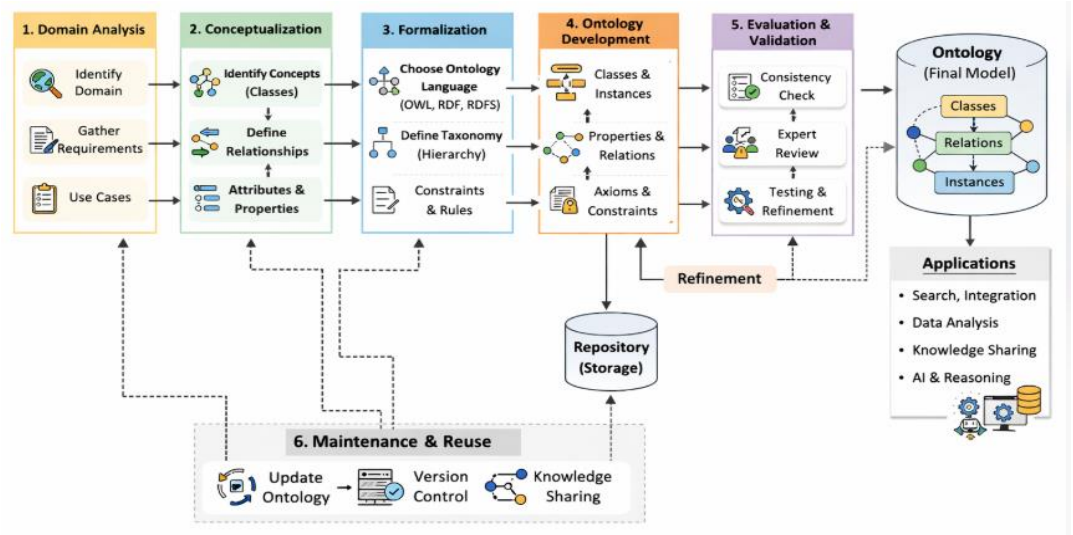


Figure 1. Ontology Modeling Process

KNOWLEDGE REPRESENTATION IN INTELLIGENT SYSTEMS

1. Role of Ontology

Ontology serves as the backbone of knowledge representation in intelligent systems. Its primary functions include:

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

- **Semantic Interoperability:** Ensures data from diverse sources can be interpreted uniformly.
- **Knowledge Sharing:** Facilitates communication between systems and users by providing a shared vocabulary.
- **Reasoning and Inference:** Enables intelligent systems to derive implicit knowledge from explicit facts.

2. Integration with AI Techniques

Ontology-based systems complement AI methodologies:

- **Expert Systems:** Ontologies encode domain knowledge for rule-based reasoning.
- **Machine Learning:** Ontologies provide semantic context to features, improving model interpretability.
- **Natural Language Processing:** Ontologies enhance entity recognition and disambiguation.
- **Robotics:** Robots use ontologies to understand environments, objects, and tasks.

ONTOLOGY CONSTRUCTION METHODOLOGIES

1. Manual Construction

Expert knowledge is encoded directly into the ontology. While precise, it is time-consuming and may not scale efficiently.

2. Semi-automatic Construction

Tools like text mining and pattern recognition extract candidate concepts and relationships from unstructured data.

3. Automatic Construction

Fully automated approaches leverage AI and machine learning to generate ontologies dynamically from large datasets.

Table 2. Comparison of Ontology Construction Approaches

Approach	Advantages	Limitations
Manual	Accurate, domain-specific	Time-consuming, labor-intensive

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

Approach	Advantages	Limitations
Semi-automatic	Faster than manual, scalable	Requires expert validation
Automatic	Scalable, adaptive to new data	May produce noisy or inconsistent ontologies

REASONING MECHANISMS

Reasoning is the process of deriving implicit knowledge from explicit facts. Common reasoning mechanisms include:

1. **Deductive Reasoning:** Uses rules to derive conclusions from known facts.
2. **Inductive Reasoning:** Generalizes from specific observations to infer broader patterns.
3. **Abductive Reasoning:** Finds the most likely explanation for observed data.

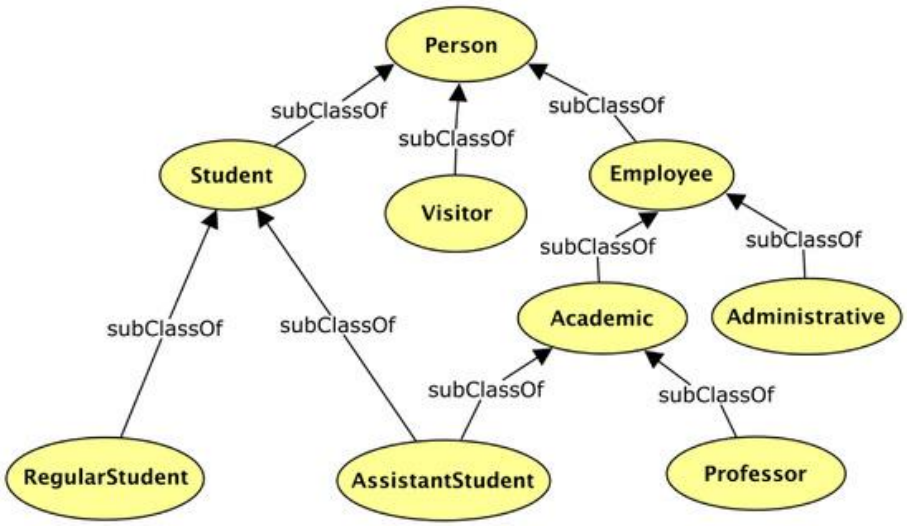


Figure 2. Reasoning in Ontology-based Intelligent Systems

APPLICATIONS

Ontology-based knowledge representation is widely applied in multiple domains:

- **Healthcare:** Clinical decision support, disease diagnosis, patient data integration.
- **E-commerce:** Product categorization, recommendation systems.
- **Smart Cities:** Traffic management, resource allocation.
- **Education:** Intelligent tutoring systems and adaptive learning platforms.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

- **Robotics:** Task planning, environment understanding, and human-robot interaction.

CHALLENGES

Despite its advantages, ontology-based knowledge representation faces several challenges:

1. **Scalability:** Handling large and complex ontologies is computationally intensive.
2. **Dynamic Knowledge Updates:** Real-time modification of ontologies is difficult.
3. **Ontology Alignment and Integration:** Merging multiple ontologies can lead to inconsistencies.
4. **Ambiguity and Vagueness:** Some domain concepts are difficult to define precisely.
5. **Tool Support Limitations:** While tools like Protégé exist, they may not scale efficiently for enterprise-level applications.

FUTURE DIRECTIONS

Future research in ontology-based knowledge representation may focus on:

- **Adaptive Ontologies:** Systems that evolve and learn from data in real-time.
- **Ontology-based Knowledge Graphs:** Integration with graph databases for complex reasoning.
- **Cross-domain Ontology Alignment:** Enhanced interoperability for multi-domain applications.
- **Integration with Deep Learning:** Combining semantic reasoning with neural networks for more intelligent systems.

CONCLUSION

Ontology-based knowledge representation provides a robust framework for building intelligent systems capable of reasoning, learning, and interacting with complex data. By formalizing domain knowledge, ontologies enhance semantic interoperability, support knowledge sharing, and enable AI-driven inference. While challenges like scalability and dynamic updates remain, ongoing research in adaptive ontologies, semantic integration, and knowledge graphs promises to make intelligent systems more context-aware, efficient, and versatile.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved

REFERENCES

1. Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, *43*(5-6), 907–928.
2. Guarino, N. (1998). Formal ontology and information systems. In *Proceedings of FOIS 1998* (pp. 3–15).
3. Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering, 25*(1-2), 161–197.
4. Horridge, M., & Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. *Semantic Web, 2*(1), 11–21.
5. Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*.
6. Maedche, A., & Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent Systems, 16*(2), 72–79.
7. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 28–37.
8. Fensel, D., & van Harmelen, F. (2007). *Unifying reasoning and search for the Semantic Web*. Springer.
9. Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2004). *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Springer.
10. Buitelaar, P., Cimiano, P., & Magnini, B. (2005). *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.

Page 117-133 © MANTECH PUBLICATIONS 2020. All Rights Reserved