International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 1515-1527
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

A Learning Ontology in Computer Programming Approach

Ahmet Doğukan Sarıyalçınkaya[1]*, Uğur Ercan[2]

[1]Ondokuz Mayis University, Vezirköprü Vocational School, Computer Programming Programme, 55900, Samsun-Turkiye
* **Corresponding Author Email:** dogukan.sariyalcinkaya@omu.edu.tr - **ORCID:** 0000-0002-1388-5114

[2]Akdeniz University, 07000, Antalya-Turkiye
Email: ugurercan@akdeniz.edu.tr - **ORCID:** 0000-0002-9977-2718

Article Info:

DOI: 10.22399/ijcesen.1355
Received : 03 January 2025
Accepted : 10 March 2025

Keywords :

Ontology,
Learning ontology,
Computer programming.

Abstract:

Advances in science and technology have made computer programming an inseparable part of our lives and have raised users' expectations from software. This situation has led to an increase in the complexity of computer programming and software development processes. To manage this complexity, models are increasingly adopted as the main structure of computer programming. On the other hand, developments in the field of linked data has spurred the use of ontologies—concepts not new to computer science—in various domains. In computer programming approaches that consider models as primary structures, it is important to formally represent requirements and ensure traceability between requirements and lower-level analysis and design models. Additionally, adapting or extending existing ontologies is one of the methods that can be employed to reduce the costs of computer programming activities. To achieve this, it is necessary to examine the differences in computer programming and the fundamentals of ontologies. These differences can be categorized under the headings of layered architecture, open-closed world approaches, and interoperability approaches. Taking into consideration the ease of incorporating ontologies in computer programming process and the difficulties reported in the scientific literature, this study proposed a model of knowledge discovery based on computer programming strategy with analogies and obtained a set of patterns for possible scenarios that can be used with a classification of the ontology in learning levels by the topics in computer programming paradigm. The aim of this research is to determine the impact of ontological learning paradigm in computer programming process by drawing a basic ontological learning map by computer programming features.

1. Introduction

Education plays a pivotal role in modern society and is gaining even greater importance in our digital era. The swift evolution of Information and Communication Technologies (ICT) has fundamentally changed the ways we acquire and disseminate knowledge[1]. Learning ontologies, which provide structured frameworks for organizing knowledge and concepts, offer a powerful means of enhancing educational practices by enabling clearer communication and shared understanding among stakeholders [2]. As educational environments become more interconnected, the need for effective data interoperability has never been more crucial. By establishing common vocabularies and relationships among various educational resources, learning ontologies facilitate seamless data sharing and collaboration among educators, thereby enriching the teaching and learning experience [3]. Furthermore, these ontologies play a pivotal role in curriculum development, allowing for the creation of interconnected curricula that reflect the diverse needs of students. As education moves toward more personalized approaches, the integration of learning ontologies supports tailored learning experiences that cater to individual students' strengths and preferences.

Collaboration and knowledge sharing among educators and researchers in the Learning ontology offer numerous benefits to the academic community [3]. By working together, professionals can combine

their expertise and resources to tackle complex research questions and develop innovative solutions. Improved communication resulting from this collaboration can significantly enhance the quality of research and educational outcomes. When educators and researchers share their knowledge and insights, they can build on each other's ideas, leading to a more comprehensive understanding of the subject matter. Moreover, the learning ontology provides a platform for interdisciplinary cooperation, allowing experts from different fields to come together and contribute their unique perspectives. This interdisciplinary approach fosters innovation and creativity, ultimately advancing the academic field as a whole. Through enhanced collaboration and knowledge sharing, educators and researchers in the learning ontology can collectively drive progress and make significant contributions to the advancement of knowledge and technology [4].

Learning ontologies provide a structured framework that aligns curriculum goals with learning outcomes by offering a clear vocabulary and metadata for learning objects. Moreover, ontology facilitates the identification of knowledge gaps and highlights areas where students may require additional support. This enables educators to make data-driven decisions, optimizing curriculum development and tailoring teaching methods to meet students' specific needs. While many educational institutions have successfully implemented informatisation tools to manage entities like "curriculum" or "course syllabus," these systems often focus primarily on formal attributes—such as study duration, assessment methods, and course placement within the curriculum. At most, such systems may offer a list of topics covered in a course; however, this information is generally not structured in a way that allows for meaningful computer-based analysis [5].

Ontology in computer programming refers to the formal representation of knowledge within a domain, typically using a structured vocabulary of entities and the relationships between them [6]. It serves as a foundational framework for organizing and categorizing information in a way that is understandable by both humans and machines. Ontologies define classes, properties, and instances, offering a shared understanding of a particular domain that can be used to facilitate knowledge sharing, data integration, and reasoning across different applications and systems. By providing a common semantic structure, ontologies play a crucial role in enhancing interoperability and enabling more intelligent and context-aware software applications [7].

Teaching and learning computer programming is a complex endeavor, and evidence suggests that instructors often fail to foster reflective problem-solving processes in their students [8]. This is particularly concerning given that many students lack prior experience or training in programming [9]. While this methodology offers significant potential for personalizing learning experiences, it also introduces challenges, including the complexity involved in developing and maintaining precise educational ontologies, the necessity for effective reasoning algorithms, and the critical need to protect student data privacy and security in the context of computer programming [10]. Furthermore, this article provides practical examples demonstrating the application of ontology-based knowledge representation in real-world educational environments.

Taking into consideration the ease of incorporating ontologies in computer pro-gramming process and the difficulties reported in the scientific literature [11], this study proposed a model of knowledge discovery based on computer programming strategy with analogies and obtained a set of patterns for possible scenarios that can be used with a classification of the ontology in learning levels by the topics in computer programming paradigm.

In the collection of information, a semi-structured online questionnaire was con-ducted with 100 individuals who give computer programming courses to classify com-puter programming. The proposed knowledge ontology has three stages: classification of the computer programming, the owl of the ontology and mind map of the ontology.

2. Related Work

2.1 Ontology

Ontology is the term "for the philosophical field that tries to answer the query 'what are there?'" [12]. It is a part of philosophy that studies "diverse kinds and structures of things, qualities, events, processes, and links" in every realm of reality. The piece provides readers with some insights regarding basic ontology, highlights its meaning, clarification, and relevance. It is needed to state that this type of philosophy plays an important role because it helps people grasp the main queries associated with reality and existence. In this respect, philosophers worry "to provide a final and comprehensive classification of entities in all spheres of being" [13]. Another goal that is attained is "to give reply to such queries as what groups of entities are required for a total description and elucidation of the world". Metaphysics, epistemology, and logic are other disciplines that are impacted by ontology because all of them are related to being-specific features of knowledge. Ontologies, whose main purpose is information sharing and

reuse, are used in many fields, including knowledge management, information retrieval, natural language processing, and artificial intelligence [14]. Ontology and ontology development concepts, which have existed for a long time, have garnered more attention with Tim Berners-Lee's vision of the semantic web, and the number of studies in this area has rapidly increased [15].

As the number of ontologies has grown, there has been a need for resources that facilitate access to these ontologies, allowing users to search and use them more easily. Ontology search engines like Swoogle and Watson have indexed and provided search capabilities for ontologies on the web [16]. Beyond search engines, web-based systems known as ontology repositories or ontology libraries have emerged, housing a collection of ontologies and enabling users to find and utilize them [17]. These systems typically contain similar ontologies within a specific scope and domain. For example, ontology libraries such as BioPortal, OBO Foundry, and OLS are important resources for accessing medical ontologies. In addition to these resources, large ontologies hosted on their own servers, such as DBpedia, YAGO, and GeoNames, are also available. These ontologies offer access through methods such as SPARQL endpoints and direct ontology file downloads.

Maedche and Staab addressed ontology as a web ontology [18]. Each data point has a descriptor, and the data in the ontology is characterized and finitely fixed. This includes the meanings of the data, the relationships between the data, the similarities and differences among the data, and their sequential and ordering relationships. For a concept set to have an ontology in the linked data, it must possess certain attributes:

- There must be a boundary for the data set, allowing for the interpretation of data in relation to one another, progressing from parts to the whole.
- The logical and mathematical operations between data sets and meaningful data must be atomic, meaning they should have a single answer.
- It should use the class structure of OWL (Web Ontology Language) [18].

According to Maedche and Staab, in addition to these, for the ontology to be more understandable and easily usable, it should:

- Contain a simple example that is easily comprehensible,
- Possess class attributes similar to those in object-oriented programming,
- Have criteria for comparing mathematical functions,
- Have criteria for comparing logical functions [18].

Ontology, the branch of philosophy that deals with the nature of being and exist-ence, plays a crucial role in education by shaping student learning and development. The different ontological perspectives held by educators can significantly impact how students perceive knowledge, themselves, and the world around them. This essay will delve into the influence of ontological perspectives on student learning, exploring how varying beliefs about the nature of reality can impact educational outcomes. Additionally, we will examine the relationship between ontology and curriculum design, as educators' ontological beliefs can influence the content, structure, and delivery of educational materials. Furthermore, we will analyze how ontological beliefs shape educational philosophies and approaches, impacting teaching methods, assessment strategies, and overall educational goals. By understanding the significance of ontology in education, we can gain insight into how different perspectives on reality can shape the learning experiences of students and inform the practices of educators.

The existential learning method concentrates on gaining awareness about their learning and existence in general. In education, applying this notion is crucial as it ena-bles learners to know more regarding their beliefs, values, and assumptions. When individuals undertake existential learning, they become more self-aware, and they can know more about themselves. It also permits them to improve in diverse areas of learning such as critical thinking. Learning this way requires knowing more about how rational one is. Meanwhile, people need to know they are not emotionally harmed when undergoing existential learning. Potential issues or criticism that might arise from using this form of learning within the conventional educational system are possible. Some modes of teaching and curriculum may fail to accommodate existential learning as it deviates from the norm. The regular educational framework may not allow such an approach as existential learning because it may be subjective.

If we examine the layered structure of the Linked data, as seen in Figure 1, the bot-tom layer contains XML (Extensible Markup Language). XML, developed after HTML, is a language used for data storage and data exchange between web pages and other software. Following XML is RDF (Resource Description Framework), which forms the data model of the Linked data, essentially establishing the rules of the database. The next layer contains Ontology, which represents the entity-relationship system within the database and is written in OWL (Web Ontology Language). The logic layer consists of standards used to strengthen the ontology

language, creating a more stable and robust structure. The proof layer enables the interaction of the linked data structure with other web languages and facilitates the transition from the whole to the particular. The trust layer is the final and most crucial feature, encompassing the security aspects [19].

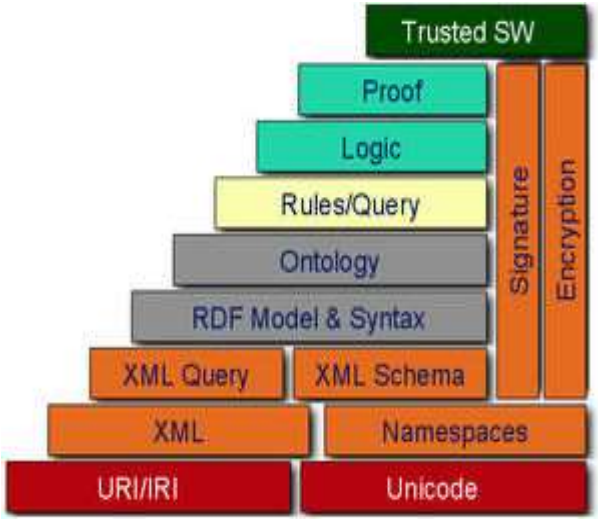

Figure 1. *Layered structure of the Linked data*

Ontology Creation Tools

Ontology creation tools are software applications used for managing information and ensuring data integrity. These tools define the data structure and relationships within a specific domain or topic, and are utilized in semantic web technologies, information management, natural language processing, and artificial intelligence applications. Table 1 shows some popular ontology creation tools: According to Table 1 the basic features of this applications are shown below.

•Protégé:

Description: An open-source ontology development tool developed by Stanford University. It offers both a graphical user interface and support for OWL (Web Ontology Language).

Features: Supports OWL, RDF, RDFS, extensibility with plugins, extensive docu-mentation.

•TopBraid Composer:

Description: A tool developed by TopQuadrant for enterprise-level ontology de-velopment and management.

Features: User-friendly interface, strong integration capabilities, supports RDF, SPARQL, SHACL.

•Ontology Editor (OBO-Edit):

Description: An open-source tool specifically developed for biomedical ontologies. It is commonly used in the Gene Ontology (GO) projects.

Features: Easy-to-use interface, biomedical data management, powerful filtering and search capabilities.

•Web Protege:

Description: The web-based version of Protégé, providing collaborative ontology development over the internet.

Features: User-friendly web interface, collaborative features, version control.

•Apollo:

Description: An open-source bioinformatics software that facilitates the annotation and visualization of biological data.

Features: Biological data management, powerful visualization tools, integrated data analysis.

•VocBench:

Description: A web-based ontology and vocabulary creation and management tool developed by the FAO (Food and Agriculture Organization)

Table 1. *Some popular ontology creation tools*

Tool	Description	Features
Protégé	Open-source ontology development tool by Stanford University.	Supports OWL, RDF, RDFS; extensible with plugins; extensive documentation.
TopBraid Composer	Enterprise-level ontology development and management tool by TopQuadrant.	User-friendly interface; strong integration capabilities; supports RDF, SPARQL, SHACL.
Ontology Editor (OBO-Edit)	Open-source tool for biomedical ontologies; used in Gene Ontology projects.	Easy-to-use interface; biomedical data management; powerful filtering and search capabilities.
WebProtege	Web-based version of Protégé for collaborative ontology development.	User-friendly web interface; collaborative features; version control.
Apollo	Open-source bioinformatics software for annotation and visualization of biological data.	Biological data management; powerful visualization tools; integrated data analysis.
VocBench	Web-based ontology and vocabulary management tool by FAO.	User-friendly interface; rich collaboration features; supports RDF, SKOS.
Ontorion Fluent Editor	Open-source tool by NeOn Project for ontology engineering and lifecycle management.	Modular architecture; extensive plugin support; integration with

Features: User-friendly interface, rich collaboration features, supports RDF, SKOS.

•Ontorion Fluent Editor:

Description: An open-source tool developed by the NeOn Project for ontology en-gineering and lifecycle management.

Features: Modular architecture, extensive plugin support, integration with OWL, RDF, SKOS.

These tools simplify the processes of ontology creation and management, facilitating better understanding and utilization of data. To select the most suitable tool for your needs and projects, it is beneficial to review their features and capabilities in detail.

The Web Ontology Language(OWL)

The Web Ontology Language (OWL) is a language used to define ontologies for the linked data [20]. OWL is designed not only for information to be understood by users but also to enable computers to perform interpretation and transformation of data. With OWL, the linked data indexing process is accomplished by converting web content and web pages written in XML and RDF into the Linked data. Linked data indexing refers to the organization of web pages according to the ontology based structure using languages like XML, OWL, and RDF. There are several editor programs for creating OWL, in the research Ontorion fluent editor was chosen to make this ontology in computer programming approach.

The Web Ontology Language(OWL)

The creation and implementation of ontologies offer numerous advantages, such as facilitating a shared understanding of information structures among individuals and software agents, and promoting the reuse of domain-specific knowledge [21]. As a result, various ontologies have been developed to improve comprehension and querying within the programming domain. Many of these ontologies focus on representing elements of the C language or object-oriented concepts found in Java and C#. One such example is PasOnto, an ontology designed to conceptualize Pascal exercise solutions, aiming to enhance learners' ability to understand and query Pascal programs. PasOnto is intended to encompass exercise descriptions along with all Pascal language constructs necessary to define a Pascal program [21]. Source code plays a dominant role in understanding software and, in many cases, is the only structural resource available to programmers for maintenance activities. In the software industry, the ability to query source code underpins a wide range of activities, from obtaining basic information about the software to automatically generating complex inferences. In computer programming, the ontology in use generally offers the researchers with a distinct vocabulary that enables them to share the data within a domain. It consists of the machine-understandable definitions of the core concepts and the connections that exist between the concepts. By sharing the ontology, people and software agents can connect. Ontologies like RDF (Resource Description Framework) and the DARPA Agent Markup Language are being built to enhance the encoding of knowledge on the web pages. The usage of the ontology in the coding language is part of the semantic web development. By providing the shared background for data organized on the web pages, coding language exploits ontologies in declaring the ontology-related variables inside the program. By sharing a common idea of the structures of the information, providing the knowledge reuse, separating the domain knowledge, and accounting the operational knowledge, the ontology of the coding language in the structure of the coding language facilitates the effectiveness of the coding language. Thus, method to show knowledge in a realm, plays a key job in programming and software development. Method gives a way to show knowledge in a standard style— that is clear because the same method structure is shared among people or software agents. It provides a vocabulary of strange concepts and connections in a formal way, so the high-level declarative logic programming is possible for automatic knowledge analysis and use. Program analysis based on method allows using declarative programming with writing logic inquiries rather than impedance with the complex compiler code for knowledge regarding program properties. The use of method for standardizing the definitions of concepts and versatile knowledge representation is the significant strength of method based program analysis as differentiated to the need for machine-oriented code programming in the situations of conventional declarative program analysis. The search results display the web pages containing the expression they typed, ranked based on popularity. However, to distinguish whether these are indeed the pages the student is looking for, semantically annotated resources, such as web pages, will be presented. This will enable students to access the most accurate information without wasting time. Consequently, students will utilize the remaining time to construct the information they seek. Individuals plan their learning according to their own needs. One dimension of this classification evaluates the use of ontologies at runtime or during software development. The other dimension is based on whether the ontology models the problem domain of the software or the infrastructure of the development environment. In this dimension, the area modeled by the ontology can be the real-world problem domain of the software or various

infrastructure areas within the software development process (such as modeling a software component library with ontologies). Programming languages are diverse, with each designed to fulfill specific tasks and support various paradigms. For developers, grasping the distinctions between these language categories is key to improving coding efficiency and selecting the right tool for a given project. While there are multiple ways to classify programming languages, they generally fall into five primary categories.:

- Functional programming languages, like Haskell and Lisp, treat computation as the evaluation of mathematical functions,
- Object-oriented programming (OOP) languages, including Python, Java, and C#, organize code around objects, encapsulating data and behavior,
- Procedural Languages, Procedural programming languages, like Pascal and Fortran, follow a linear, top-down approach to code organization,
- Scripting languages, such as Python, JavaScript, and Ruby, are designed for automating the execution of tasks,
- Adding another layer to our exploration, logic-based programming lan-guages like Prolog introduce a different paradigm.

We opted for the knowledge domain of computer programming, given that the learning of at least one programming language is the common requisite by drawing an ontology map to computer programming to address the basic computer programming procedure, which can draw an ontological map to make computer programming approach easier according to the contents of the programming language contents.

The Hypoteses

The research problem was defined to draw an ontology map to computer pro-gramming to address the basic computer programming procedure, which can draw an ontological map to make computer programming approach easier according to the contents of the programming language contents.

3. Material and Methods

3.1. Research Design

In the research, an online survey was used to ontologically illustrate learning to classify topics in computer programming. The use of online questionnaires has increased in research due to the spread of internet use and recent COVID-19 pandemic. Online questionnaires provide advantages in terms of cost, speed, and convenience. Descriptive research, due to its quantitative nature, is the most commonly used and most accurate form of survey research. Unlike exploratory research methods, descriptive research employs pre-planned, structured, and closed-ended questions. This research method is based on the principle of deduction, meaning the structure and questions of the survey are predetermined according to existing theories or areas of study. The collected data is then used to test hypotheses or assumptions. The purpose of descriptive research is to measure and categorize the opinions, attitudes, or beliefs of a specific population on a particular topic. For instance, multiple-choice questions that provide predefined response options can be used for a descriptive survey. While these standardized questions may not provide the depth offered by qualitative insights, they yield data that can be statistically analyzed and used to draw inferences. By grouping responses into categories, you can measure how prevalent certain views or behaviors are within your target audience. This allows you to identify changes in attitudes and trends by comparing changes over time.

3.2. Participants

The research was conducted with 100 individuals who give computer programming courses. The sample of the research was selected through purposive sampling, which is one of the probability sampling methods. Probability sampling involves selecting units from the population with equal probability each time, and its distinguishing feature is the random selection of elements from the population. In such selections, the investigated groups are divided into homogeneous groups as long as they have similar characteristics Table 2 shows the demographic characteristics of the survey participants.

Table 2. *The demographic characteristics of the survey participants.*

	Female	**Male**	**Total**
Prof.	5	20	25
Assoc.Prof.	5	20	25
Lecturer	10	15	25
Programmer	5	20	25
	25	75	100

Based on Table 2, 100 experts participated in the survey, of which 25 were female and 75 were male who hold the titles of Professor, Assoc. Prof., Lecturer and Programmer.

3.3. The Digital Questionnare

Programming experience provides the technical foundation, problem-solving mindset, and toolset

required to efficiently construct, debug, and integrate ontologies within various systems. Improving programming skills requires practicing rules, basics, and core concepts of computer programming (how to declare variable, to use selective or loop structure, to define functions and procedures, …). We are conducting a survey to understand how programming levels are determined based on different topics. The participants fill out the following information by GoogleForms for follow-up purposes:

- Name:
- Email Address:

Profession: Please select one from the options below or specify another profession if not listed:

- Software Developer
- Data Scientist
- Teacher/Instructor
- Student
- Other (please specify):

Programming Experience: Please indicate your experience in years by selecting one of the following options:

- 1-3 years
- 3-5 years
- 5+ years

Beginner Level Topics: In your opinion, which topics should be included to define the beginner level in programming? Possible topics include:

- Variables and Data Types
- Control Structures (if, else)
- Loops (for, while)
- Functions/Methods
- Other (please specify any additional topics):

Intermediate Level Topics: When defining the intermediate level, which topics do you consider essential? Possible topics include:

- Advanced Control Structures (switch, case)
- Object-Oriented Programming (OOP)
- File Operations
- Database Connections
- Other (please specify any additional topics):

Advanced Level Topics: For the advanced level, which topics do you believe are important? Possible topics include:

- Algorithms and Data Structures
- Parallel Programming
- API Integration
- Security
- Other (please specify any additional topics):

3.4. The Ontology

The ontology of the study was created by using Ontorion fluent editor ontology creation software. Accordingly, two objects were created: "thing" and “level”. In Figure 2, under the "user" object, the user's name and their level obtained from the on line questionnaire are entered as sub-values. Under the "site" object, the addresses of formal sites predetermined by expert instructors and their difficulty levels are entered as sub-values as it is shown on the taxonomy tree on the right drop down menu as it is categorized. As a result of these linked data operations, sites corresponding to the user's name and level will automatically appear. In other words, under the user, there are sub-classes of the name and level. Under the level class, there are sub-classes such as beginner, intermediate, and advanced. Under the "site" object, there are sub-classes of site address and level. Under the level class, there are common values such as beginner, intermediate, and advanced, similar to the sub-classes in the user object's level. The user object and the site object are mathematically processed through intersection sets based on the level. Thus, if the user's level is beginner, they are directed to sites with beginner-level content.

4. Results and Discussion

In the research, the data obtained from on line questionnaire after the implementation of materials prepared in accordance with the ontological learning process in computer programming were analyzed. The reliability of the online scale was tested by using Cronbach's alpha analysis. The Cronbach's alpha analysis was conducted by using the SPSS 21 program, which is shown in Table 3.

Table 3. *Reliability Statistics*

Cronbach’s Alpha	Number of Questions
,78	6

According to Table 3, the scale's Cronbach's alpha reliability was found to be 78%, indicating that this reliability value is quite sufficient. Cronbach's alpha values range from 0 to 1. A value above 0.7 is generally considered acceptable for research purposes, indicating good internal consistency.

The contents of the computer programming paradigm is divided into three levels, which is shown in Table 4.

According to Table 4, the ontology for programming levels is divided into three main categories: Beginner, Intermediate, and Advanced. At the Beginner Level, learners are introduced to basic programming concepts such as variables, control structures, loops, and functions. Topics under this level include Variables, which cover the basic concepts of variables and data types; Control Structures, focusing on basic control structures like if statements and loops; Loops, which include basic

loops such as for and while loops; and Functions, which discuss basic concepts of functions and methods.

At the Intermediate Level, learners delve into more complex topics such as ad-vanced control structures, object-oriented programming, file operations, and database connections. This level includes Advanced Control Structures, covering advanced control structures such as switch statements and recursion; Object-Oriented Programming, which explores concepts of object-oriented programming including classes and objects; File Operations, focusing on file handling and operations such as reading and writing files; and Database Connections, which involve connecting to and interacting with databases.

At the Advanced Level, learners examine topics such as algorithms and data structures, parallel programming, API integration, and security in depth. Topics under this level include Algorithms, focusing on the in-depth study of algorithms and their complexities; Data Structures, which cover advanced data structures such as trees and graphs; Parallel Programming, which discusses techniques and concepts for parallel and concurrent programming; API Integration, focusing on integrating and using APIs in software development; and Security, which covers security concepts and practices in programming.

According to the, the ontology for programming levels, which is divided into three main categories, the owl of this ontological learning process is shown in Figure 3.

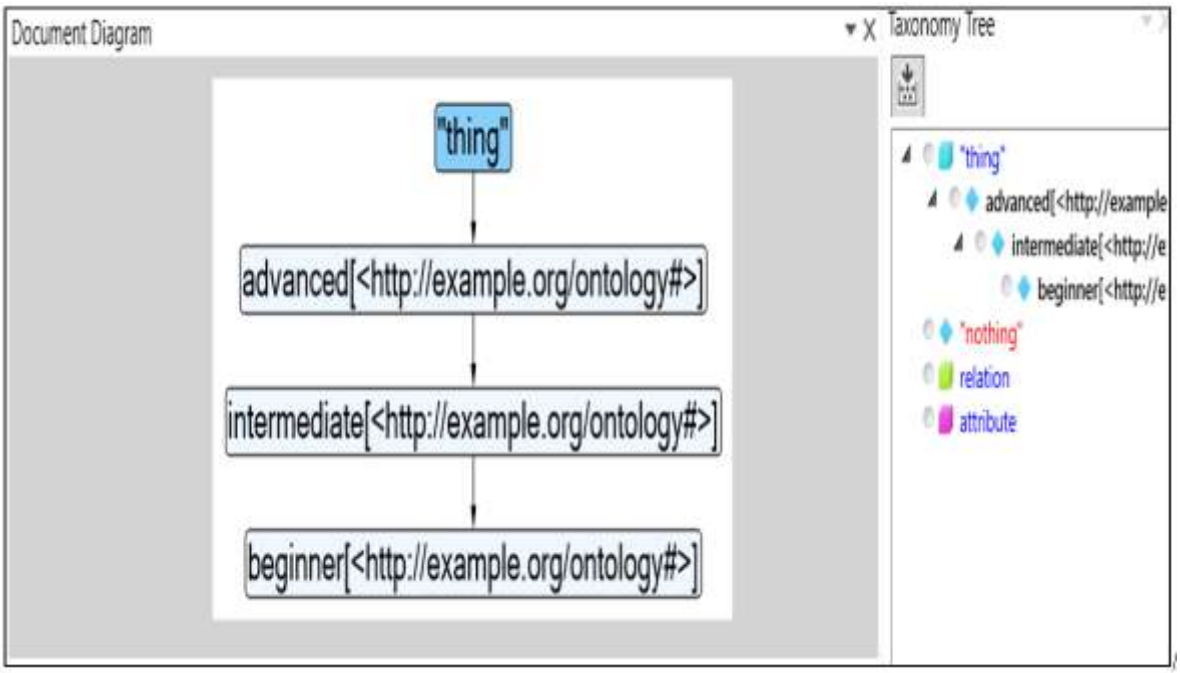

Figure 2. *The Ontology Map*

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# Classes defining programming levels
<http://example.org/ontology#Beginner> rdf:type rdfs:Class ;
    rdfs:label "Beginner Level" .

<http://example.org/ontology#Intermediate> rdf:type rdfs:Class ;
    rdfs:label "Intermediate Level" .

<http://example.org/ontology#Advanced> rdf:type rdfs:Class ;
    rdfs:label "Advanced Level" .

# Relationships between programming levels
<http://example.org/ontology#Beginner> rdfs:subClassOf <http://example.org/ontology#Intermediate> .
<http://example.org/ontology#Intermediate> rdfs:subClassOf <http://example.org/ontology#Advanced> .

# Properties of programming levels
<http://example.org/ontology#Beginner> rdfs:comment "Learning level for basic programming concepts such as
variables, basic control structures, loops, and functions." .
<http://example.org/ontology#Intermediate> rdfs:comment "Level where advanced control structures,
object-oriented programming, file operations, and database connections are understood." .
<http://example.org/ontology#Advanced> rdfs:comment "Level where algorithms and data structures, parallel
programming, API integration, and security topics are deeply examined." .
```

Figure 3. *The owl of the computer programming ontology*

Table 4. *The classification of the computer programming paradigm according to the questionnaire*

Level	Number of Classification	Label	Comment
Beginner		Beginner Level	Learning level for basic programming concepts such as variables, basic control structures, loops, and functions.
- Variables	90	Variables	Basic concepts of variables and data types.
- Control Structures	95	Control Structures	Basic control structures such as if statements and loops.
- Loops	85	Loops	Basic loops such as for and while loops.
- Functions	80	Functions	Basic concepts of functions and methods.
Intermediate		Intermediate Level	Level where advanced control structures, object-oriented programming, file operations, and database connections are understood.
- Advanced Control Structures	80	Advanced Control Structures	Advanced control structures such as switch statements and recursion.
- Object-Oriented Programming	75	Object-Oriented Programming	Concepts of object-oriented programming including classes and objects.
- File Operations	90	File Operations	File handling and operations such as reading and writing files.
- Database Connections	80	Database Connections	Connecting to and interacting with databases.
Advanced		Advanced Level	Level where algorithms and data structures, parallel programming, API integration, and security topics are deeply examined.
- Algorithms	65	Algorithms	In-depth study of algorithms and their complexities.
- Data Structures	75	Data Structures	Advanced data structures such as trees and graphs.
- Parallel Programming	80	Parallel Programming	Techniques and concepts for parallel and concurrent programming.
- API Integration	55	API Integration	Integrating and using APIs in software development.
- Security	60	Security	Security concepts and practices in programming.
Beginner		Beginner Level	Learning level for basic programming concepts such as variables, basic control structures, loops, and functions.
- Variables	90	Variables	Basic concepts of variables and data types.
- Control Structures	95	Control Structures	Basic control structures such as if statements and loops.
- Loops	85	Loops	Basic loops such as for and while loops.
- Functions	80	Functions	Basic concepts of functions and methods.
Intermediate		Intermediate Level	Level where advanced control structures, object-oriented programming, file operations, and database connections are understood.
- Advanced Control Structures	80	Advanced Control Structures	Advanced control structures such as switch statements and recursion.
- Object-Oriented Programming	75	Object-Oriented Programming	Concepts of object-oriented programming including classes and objects.
- File Operations	90	File Operations	File handling and operations such as reading and writing files.
- Database Connections	80	Database Connections	Connecting to and interacting with databases.
Advanced		Advanced Level	Level where algorithms and data structures, parallel programming, API integration, and security topics are deeply examined.
- Algorithms	65	Algorithms	In-depth study of algorithms and their complexities.
- Data Structures	75	Data Structures	Advanced data structures such as trees and graphs.
- Parallel Programming	80	Parallel Programming	Techniques and concepts for parallel and concurrent programming.
- API Integration	55	API Integration	Integrating and using APIs in software development.
- Security	60	Security	Security concepts and practices in programming.

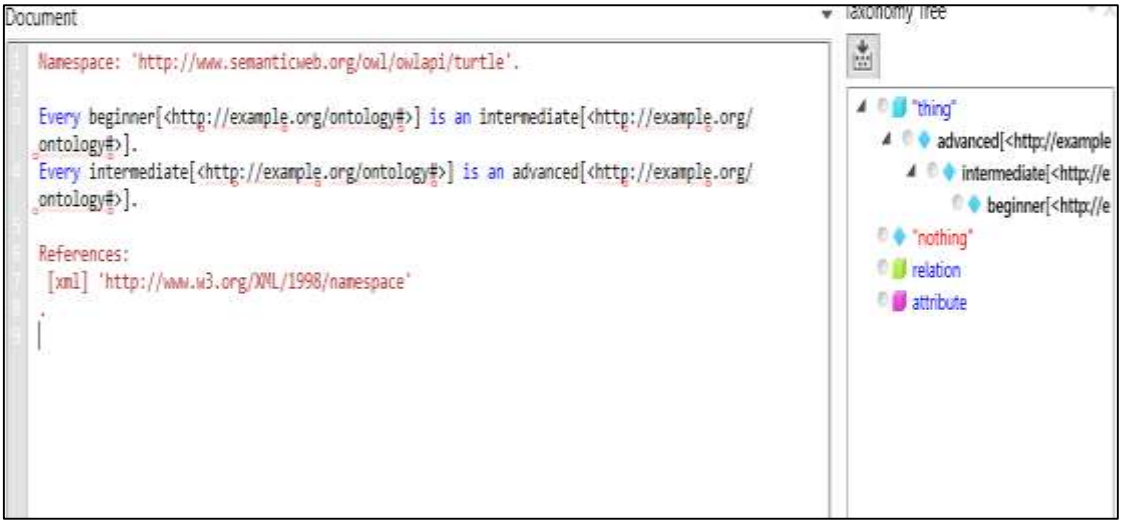

Figure 4. *The terminology of the computer programming*

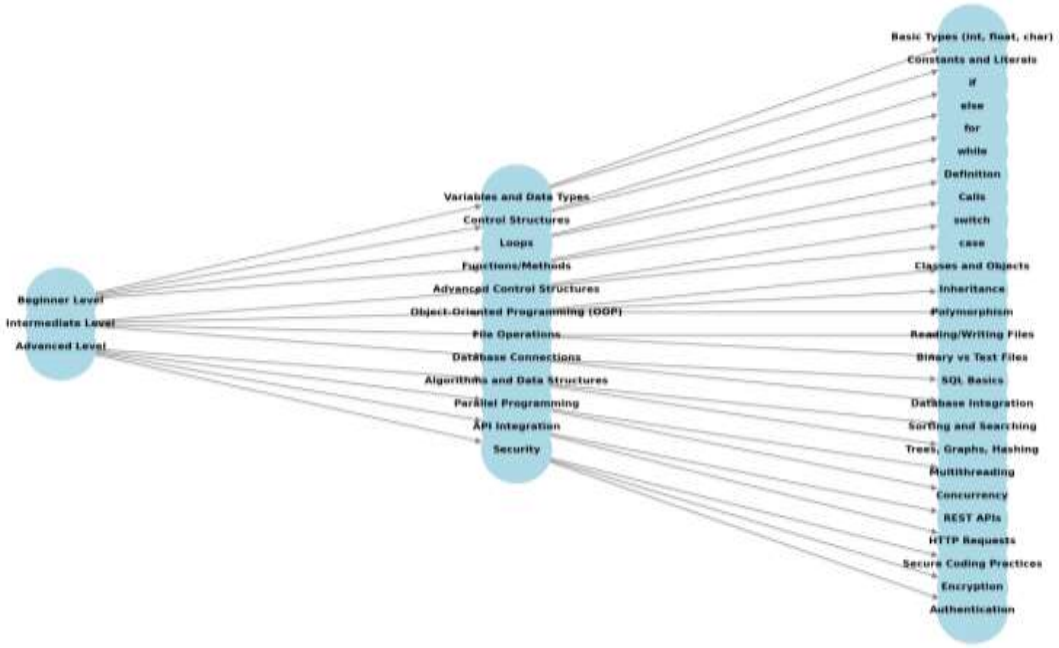

Figure 5. *The mind map of computer programming ontology*

The ontology is structured into three levels of programming expertise: beginner, intermediate, and advanced. Each level encompasses specific topics pertinent to that level of proficiency. At the beginner level, learners focus on fundamental programming concepts such as variables, basic control structures (like if statements and loops), loops (for and while), and basic functions. Intermediate level topics delve into more complex areas such as advanced control structures (including switch statements and recursion), object-oriented programming (OOP) principles, file operations (handling reading and writing of files), and database connections. The advanced level includes in-depth studies of algorithms and their complexities, advanced data structures (like trees and graphs), parallel programming techniques, API integration, and security practices in programming. This structured approach helps in systematically advancing the learner's knowledge and skills in programming. The terminology of this ontology is shown on Figure 4.

According to Figure 4, the given owl notation defines an ontology with a base namespace http://www.semanticweb.org/owl/owlapi/turtle, which serves as the default URI for the ontology. Within this ontology, it specifies hierarchical relationships among three classes: Beginner, Intermediate, and Advanced, all prefixed with http://example.org/ontology#. Specifically, it states that every instance of the Beginner class is also an instance of the Intermediate class (Beginner is a subclass of Intermediate), and every instance of the Intermediate class is also an instance of the Advanced class (Intermediate is a subclass of Advanced). Additionally, the notation references the XML namespace

http://www.w3.org/XML/1998/namespace, often used in XML standards, indicating that the ontology may incorporate or align with XML-related elements. The ontology map structured into three levels of programming expertise: beginner, intermediate, and advanced as it is shown on Figure 5. This ontology illustrates a mind map for programming skills, divided into three key levels: Beginner, Intermediate, and Advanced. The structure is hierarchical, and each level has its own branches detailing relevant subtopics. The core concept is labeled as Skill, from which all other nodes branch out. This map serves as an educational tool or roadmap for learning programming, showing how concepts build on each other as you progress from beginner to advanced level.

Ontology in computer programming refers to the structured framework that defines the entities, concepts, and relationships within a domain. Originating from philosophical ontology—the study of the nature of being—its application in computer programming serves as a foundation for organizing knowledge in a way that both humans and machines can process effectively. This discussion explores the significance, challenges, and applications of ontology in programming.

Ontologies enable programmers to formalize and communicate complex systems. By providing a shared vocabulary and a structured schema, they ensure consistency across different systems and stakeholders. For instance, in domains like e-commerce, healthcare, and artificial intelligence, ontology facilitates interoperability by defining a standard representation of concepts such as "product," "patient," or "algorithm."

Moreover, ontologies play a critical role in semantic computing. Technologies like the Semantic Web rely on ontologies to make the web's content machine-readable and understandable. Through the use of languages such as OWL (Web Ontology Language) and RDF (Resource Description Framework), programmers can create knowledge graphs and enable intelligent search, data integration, and reasoning systems. Ontology development involves defining classes, properties, and relationships to create a structured vocabulary that enhances knowledge representation and interoperability. This guide aims to empower novice developers with essential methodologies and tools for creating and managing ontologies tailored to specific domains. On the other hand, in The Role of Ontologies and Linked Data," ontologies are seen as pivotal in the AI, facilitating the integration and expression of data across diverse sources. It emphasizes the role of ontologies in enhancing semantic search, automated reasoning, and interoperability with technologies like RDF and SPARQL [14]. Together, these perspectives illustrate how ontologies not only organize information but also enhance the intelligence and interoperability of software applications within and beyond traditional programming contexts.

Fundamental is the exercise of ontological exploration in programming, settling coding matters to be deciphered. Coders sense such deeds and employ commonplace language processes to fortify their capability in structuring data, which is an ontological process of producing facts within a given extent. Investigating ontology is a tactic contemplated for augmenting puzzle-solving potentialities [15]. The capacity of coders to resolve puzzles ameliorates, and their actions for unraveling intricate problem-solving challenges are partially facilitated. The study of producing facts on a given focus point assists in this process. The evolution of comprehension can benefit future code scribes, enabling them to share data about data layouts. This might even permit for the reuse of data from certain fields as part of the learning process, or for deliberating operational specifics apart from field-based data. The study of producing facts on a given focus point could be employed in programming pedagogy to address the style of code scripts performed by coders in a more structured fashion.

Programming languages are diverse, with each designed to fulfill specific tasks and support various paradigms. For developers, grasping the distinctions between these language categories is key to improving coding efficiency and selecting the right tool for a given project. While there are multiple ways to classify programming languages, they generally fall into five primary categories. In that regard, our programming ontology proposes the curriculum ranging from basic to more complex exercises. So, student have the possibility to practice programming by visualizing solutions and comparing them with their work. Nevertheless, proposed solutions are not semantically structured to be queried efficiently.

5. Conclusion

There is an approach of being kept in mind in the world of coding so-called ontological learning, which promises notable code analysis and development boost. In the sphere of code analysis, where the ontology standardizes trusty field rules, links, and ideas, coders may unite the code builds and related knowledge into a single depiction.

Knowledge-map directs the way we represent and grasp wisdom. It exists as a philosophical idea or used in the field of wisdom representation. It allows computers to "think" correctly and make good choices in the domain of artificial intelligence. Its

sig-nificance can be found in areas from computer science to ontology philosophy. Fur-thermore, knowledge-map arranges the structure of wisdom. In other words, it precisely describes what we call wisdom and the reasons for calling it wisdom. Generally, what we refer to as knowledge-map is a formal knowledge-map. Other wide classifications include metaphysical knowledge-map and descriptive knowledge-map. It is hard to imagine science devoid of classifications or proper information structuring. If taken to a wider setting, it is hard to imagine philosophy devoid of proper code to represent concepts. That code is provided by knowledge-map.

The benefits of spreading the method in computer coding into ontological stages are clearly evident. Below these situations, it is possible to enhance the overall quality of software development. In particular, separating the usefulness from the user view contributes to the creation of more long-lasting and handy software. The ontological stage system gives a good framework for handling difficulty. One layer performs its job, interacting with the other layer as per the appointed scheme. Many people are convinced that it is vital to introduce the method to using the ontological stage system in computer programming. This tactic will allow enhancing the quality of software and devote more time to mindful work on the program. Since the need for appropriate and high-quality software will only increase in the future, the ontological stage system will be the right fix.

An ontology map like this one is primarily helpful for planning and learning rather than direct coding. Here's how it can be beneficial:

Learning Path: The map helps beginners, intermediate coders, and advanced programmers see the sequence and scope of topics they should cover as they progress, which is valuable for self-study or course design.

Curriculum Planning: In educational settings, the map aids instructors in struc-turing lessons, creating a syllabus, or organizing course materials by skill level. Skills Assessment: For developers, it provides a reference to evaluate their knowledge and identify areas for improvement. Project Planning: For complex projects, knowing which level of knowledge each team member should have can help in assigning tasks that match their expertise. However, when it comes to actual coding tasks, the map is a high-level reference rather than a coding aid. Practical coding requires detailed syntax knowledge, hands-on experience, and familiarity with specific tools, libraries, and development environments. For future work, we will design a recommendation system that will guide the professor in the approach of an analogy to address a fundamental concept of a CS1 and develop a visualization tool that supports the process of abstraction in computing environments.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Villegas-Ch, W., & García-Ortiz, J. (2023). Enhancing learning personalization in educational environments through ontolo-gy-based knowledge representation. *Computers*, 12(10), 1994.

[2] Dou, D., Wang, H., & Liu, H. (2015, February). Semantic data mining: A survey of ontology-based approaches. *In Proceedings of the 2015 IEEE 9th international conference on semantic computing (IEEE ICSC 2015)* (pp. 244-251

[3] Stancin, K., Poscic, P., & Jaksic, D. (2020). Ontologies in education–state of the art. *Education and Information Technologies*, 25(6), 5301-5320.

[4] Tapia-Leon, M., Rivera, A. C., Chicaiza, J., & Luján-Mora, S. (2018, April). Application of ontologies in higher education: A systematic mapping study. *In 2018 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1344-1353).

[5] Husáková, M., & Bureš, V. (2020). Formal ontologies in information systems development: a systematic review. Information, 11(2), 66.

[6] Wawrzik, F., Rafique, K. A., Rahman, F., & Grimm, C. (2023). Ontology learning applications of knowledge base con-struction for microelectronic systems information. *Information,* 14(3), 176.

[7] Somodevilla García, M., Vilariño Ayala, D., & Pineda, I. (2018). An overview of ontology learning tasks. *Computación y Sistemas,* 22(1), 137-146.

[8] Nazyrova, A., Milosz, M., Bekmanova, G., Omarbekova, A., Mukanova, A., & Aimicheva, G. (2023). Analysis of the Con-sistency of Prerequisites and Learning Outcomes of Educational Programme

Courses by Using the Ontological Approach. *Applied Sciences*, 13(4), 2661.

[9] Qaswar, F., Rahmah, M., Raza, M. A., Noraziah, A., Alkazemi, B., Fauziah, Z., ... & Sharaf, A. (2022). Applications of ontology in the internet of things: A systematic analysis. *Electronics*, 12(1), 111.

[10] Guizzardi, R., Carneiro, B. G., Porello, D., & Guizzardi, G. (2020, November). A core ontology on decision making. In *Pro-ceedings of the XIII Seminar on Ontology Research in Brazil and IV Doctoral and Masters Consortium on Ontologies* (ON-TOBRAS 2020) (Vol. 2728, pp. 9-21). CEUR-WS.

[11] Effingham, N. (2013). An introduction to ontology. *John Wiley & Sons*.

[12] Smith, B. (2012). Ontology. In The furniture of the world (pp. 47-68). *Brill.*

[13] Staab, S., & Maedche, A. (2001). Knowledge portals: Ontologies at work. *AI magazine*, 22(2), 63-63.

[14] Demirli, C., & Kütük, Ö. F. (2010). Semantic Web (Web 3.0) and an overview of its ontologies. *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, (9/18), 95-105.

[15] Sevinc, O., Huang, L., Loughzang, L., & Kilic, E. (2015). Geospatial information retrieval base on query expansion and semantic indexing. *Journal of Engineering and Fundamentals*, 2(2), 51-68.

[16] Diatta, B., Basse, A., & Ouya, S. (2019, April). PasOnto: Ontology for learning Pascal programming language. *In 2019 IEEE Global Engineering Education Conference (EDUCON)* (pp. 749-754). IEEE.

[17] Hussain, F. (2012). E-Learning 3.0= E-Learning 2.0+ Web 3.0?. *International Association for Development of the Information Society*.

[18] Brickley, D. and Guha, RV (1999). Resource Description Framework (RDF) Schema Specification. *Proposed Recommendation, World Wide Web Consortium:* http://www.w3.org/TR/PR-rdf-schema.

[19] Zhao, Y., Chen, G., Liao, C., & Shen, X. (2016). Towards ontology-based program analysis. *In 30th European Conference on Object-Oriented Programming (ECOOP 2016).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[20] Băjenaru, L.; Smeureanu, I. Learning Style in Ontology-Based E-Learning System. In *Proceedings of the International Confer-ence on Informatics in Economy, Cluj-Napoca, Romania,* 2–3 June 2018; Volume 273.

[21] García-García, J., Gil, M. Á., & Lubiano, M. A. (2024). On some properties of Cronbach's α coefficient for interval-valued data in questionnaires. *Advances in Data Analysis and Classification*, 1-24.