Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A SYSTEMATIC AGILE FRAMEWORK FOR TEST-DRIVEN ONTOLOGY VALIDATION IN ACADEMIC PERFORMANCE ANALYTICS AND DECISION-MAKING

MOHD HAFIZAN MUSA [1,2], SAZILAH SALAM [3,4,*], MOHD ADILI NORASIKIN[5], MUHAMMAD SYAHMIE SHABARUDIN[6], UNING LESTARI[7]

[1]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Malaysia

[2]College of Computing, Informatics and Mathematics, Universiti Teknologi MARA (Segamat), Johor, Malaysia

[3]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Malaysia

[4]Faculty of Engineering and Physical Sciences, University of Southampton, United Kingdom

[5]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Malaysia

[6]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Malaysia

[7]Faculty of Science and Information Technology, Universitas Akprind Yogyakarta, Indonesia

E-mail: [3]sazilah@utem.edu.my, [4]S.Binti-Salam@soton.ac.uk

ABSTRACT

The rapid growth of educational data from diverse e-learning platforms such as Learning Management Systems (LMS) and Student Information Systems (SIS) presents challenges for universities in integrating and analyzing this data to monitor student performance, assess course effectiveness, and optimize faculty resource allocation. Ontologies provide a robust framework for enabling semantic interoperability and facilitating the integration of heterogeneous data sources for Learning Analytics (LA) and decision-making purposes. This study introduces the SPC_Academic_Performance ontology, a domain-specific ontology developed to consolidate and analyze academic performance data. To ensure the reliability and accuracy of the SPC_Academic_Performance ontology, we adopt the Test-Driven Development Ontology (TDDOnto2) methodology. TDDOnto2 systematically integrates validation techniques into the ontology development process, focusing on consistency checking and property testing. By applying TDDOnto2, this study aims to address common challenges such as logical inconsistencies and incomplete property definitions, ensuring the ontology's robustness for data integration and retrieval. The findings contribute to developing a systematic ontology validation framework that supports reliable ontology-driven analytics and informed decision-making in higher education. This approach ensures that the proposed ontology can effectively map and retrieve data from heterogeneous sources, ultimately enhancing the accuracy and utility of Learning Analytics in academic performance monitoring and resource management.

Keywords: *Learning Analytics, University Ontologies, Data Retrieval Model, Ontologies Evaluation, Ontologies Validation, Web Semantic Ontology*

1. INTRODUCTION

The fast pace of digitalisation brought advancements to higher education, which significantly changed the ways in which educational institutions collect, manage, analyse, and use data [1], [2], [3]. E-learning platforms, such as Learning Management Systems (LMS) and Student Information Systems (SIS), serve as core tools for facilitating education and producing vast amounts of heterogeneous data. This data consists of various metrics, including student grades, attendance records, course evaluations, and resource allocations. While the use of such data can enhance decision-making processing, there are technical and semantic complexities when integrating data across various platforms [2], [4], [5]. Not only SQL (NoSQL) graph databases are a type of NoSQL database that is built to handle data structures that are very complicated and highly connected [6]. These databases store the data in the form of nodes, edges, and properties, which are very convenient as a relationship within these educational datasets. By integrating NoSQL graph databases with ontologies, institutions can enhance their capacity to manage and query interconnected data, creating a robust foundation for Learning Analytics and decision-making [4]. Ontology-based systems have been seen as the answer to these problems, providing a way of defining how data is represented and improving the ability to integrate data from different systems [1], [4], [7].

The semantic web enriches the integration process by facilitating a "Web of Data." Technologies like RDF (Resource Description Framework) and OWL (Web Ontology Language) play a critical role in enabling semantic annotations and logical reasoning over data [8], [9]. OWL provides formal language constructs to define classes, properties, and relationships, enabling precise knowledge representation [10], [11]. The SPC_Academic_Performance ontology, designed within this framework, consolidates data from multiple e-learning platforms to provide a unified foundation for analyzing academic performance and optimizing resource allocation [12], [13]. However, guaranteeing the trustworthiness and precision of such an ontology is essential for its effectiveness. Logical inconsistencies, incomplete property definitions, and inadequate testing can compromise its effectiveness and usability [2], [4].

Most conventional approaches to ontology development lack thorough validation procedures and typically provide only basic validation features tools [14], which may result in undetected errors and reduced stakeholder confidence [4]. In contrast, TDDOnto2 offers enhanced accuracy over standard ontology editors [14] and introduces an agile, test-driven approach to ontology development. It allows researchers to create and execute validation tests before or during the modeling process to ensure correctness throughout. Meaning that researchers can iteratively refine their Description Logic (DL) constructs until all tests are fully passed. Furthermore, the tool integrates a regression algorithm which enhances ontology validation accuracy while producing dependable results.

To address these gaps, this research uses TDDOnto2, a test-driven development framework to check the viability of an ontology known as SPC_Academic_Performance. It is evaluated for logical content and property checking to ensure that all class taxonomical structures, well cardinalities, and object property connections exactly reflect the semantics of the relevant domain as stated. The methodology introduces a systematic structured testing process that will assess the ontology via consistency checks, property validations, and individual test cases to finalise the evaluation. This research shows how TDDonto2 can be used in practice as evidenced by the enhancement of the quality of the ontology by the framework.

The research contributes clear guidelines for practical ontology validation that offer agile regression testing to make validation processes more dependable and manageable. This research also reveals challenges and effective ways of employing TDDOnto2 in ontology validation, thereby providing other scholars and developers with similar approach. This work moves the field of ontology engineering forward by breaking the research down into the design and validation phases, making certain that the ontologies produced are correct in their semantics, as well as effective in the contexts in which they are to be used. In previous study [15], a variant of Student Performance and Course Performance (SPC) OWL ontology named SPC_Academic_Performance was introduced in learning analytics (LA) to study the heterogeneous pattern of student and course performance. Hence, this paper aims to validate the SPC_Academic_Performance ontology developed earlier. The validation will assess the ontology via consistency checks, property validations, and individual test cases to finalise the evaluation.

The following sections of this paper are structured as follows: Section 1 provides an overview of some of the ontologies validation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

techniques available, as well as how the DL is applied to integrate and link all the heterogeneous data and motivation to use the TDDonto2 in the study. Section 2 defines the structure of the SPC_Academic_Performance ontology and the configuration established for testing. Section 3 presents the results and interpretation of the test, while Section 4 offers the study's conclusion.

2. LITERATURE REVIEW

2.1 Current Practices in Ontology Validation

With an increasing number of ontologies, there is an increasing demand for innovative tools and methodologies to achieve consistency among various information representations. This is also applicable to the approach of validating the ontologies. Researchers employ various tools and techniques to validate the ontologies. The assessment may include checking the class and subclass definition errors, class redundancy errors, and class linkage errors. The class linkage error occurs when the relationship or the object properties set among the Subject, Predicate, and Object are insufficient to complete the triples.

The tool like OntoAnalyser [16], Onto Generator [16], OntoClean [16], OntoVal [17], ONE-T [16], and Ontology Pitfall Scanner! (OOPS!) [16] are quite common to be used in the validation process. A study by [16] evaluated the performance of several commonly available ontology validation tools. As for OOPS!, it operates by utilising a predefined catalog of pitfalls, which are categorised based on their severity: critical, important, and minor to allow users to prioritise issues effectively [18]. The OOPS! is also quite common to be use in the area of ontologies that belong to LA domain where it serves to validate the ontologies for the semantic representation of syllabuses ontology (OntoSyllabus) [19], Curriculum Course Syllabus Ontology (CCSO) [20] and, Ontology for Linked Open University Data (OLOUD) [21], [22].

To increase the efficiency of the ontologies produced, some studies implemented the Competency Question (CQ), which is a set of questions that are used to evaluate the quality of the ontology based on the SPARQL result generated [20], [21], [22], [23]. To determine the accuracy of the constructed ontology, the results retrieved from the data retrieval must align with the expectations of the domain expert. Studies conducted by [24], [25], [26], [27] proposed an enhanced data retrieval framework incorporating an additional layer that leverages SPARQL query generation from RDF-based ontologies, with performance assessed through evaluation metrics including accuracy, precision, recall, and F1-score.

2.2 Description Logic in Semantic Data Retrieval

To define the structure of data, a data model is needed for this purpose. The common data model that we heard about is the Entity Relationship Diagram (ERD), which is being used widely in Relational Database Management Systems (RDBMS). The concept of the ERD is somewhat analogous to OWL semantic modelling. However, the ERD and OWL semantic models serve diverse data representation and knowledge management functions [28]. ERDs are primarily utilised in database design to visually depict the relationships between entities within a system, focusing on the structure of data and how different entities interact with one another. In contrast, OWL is a formal language intended for the representation of intricate and comprehensive knowledge concerning entities, collections of entities, and their interrelations inside the semantic web.

In an OWL, Description Logic (DL) performs critical support for important reasoning services in ontology-based systems. For example, reasoning about class hierarchies, property characteristics, or the relationships between instances can be done using DL reasoners, which are unique algorithms for managing the tough semantics of DL [29], [30]. These DL reasoners can determine the satisfiability, entailment, and consistency of ontologies, which are essential for ensuring the correctness and reliability of knowledge representation in OWL [31], [32]. Additionally, it has been investigated to improve the expressiveness and application of OWL in various areas by integrating DL into other computational paradigms, such as rules and probabilistic reasoning [33], [34].

In the context of OWL, DL serves as the underlying framework that defines the semantics of the language. OWL is designed to facilitate the creation of ontologies that can be shared and reused across different applications on the Semantic Web. Specifically, OWL Lite and OWL DL are based on specific DLs, such as base SHIF(D) and a more advanced SHOIN(D), respectively [33]. In this study, all the axioms tested in the TDDOonto2 plugin use the SHIF(D) language.

Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Examples of DL axiom functions with statements

Description	DL expression	Explanation
Defining a class	Student ⊑ Person	Every Student is a type of Person (i.e., Student is a subclass of Person)
Class with a restriction	Student ⊑∃enrolledIn.Course	Every Student is a type of Person (i.e., Student is a subclass of Person)
Logical AND	GraduateStudent ≡ Student ⊓ hasDegree	A GraduateStudent is someone who is a Student and has a Degree
Logical OR	Course ≡ UndergraduateCourse ⊔ GraduateCourse	A Course is either an UndergraduateCourse or a GraduateCourse
Cardinality Restriction	Student ⊑∃ belong_to.Faculty ⊓ ≤ 1 belong_to.Faculty	Every Student belong to only one Faculty

To execute the validation testing, the axioms are required to be feed in the TDDonto2 plugin. The complete differences between TBox and Abox axioms are explained in the study of [35].

2.3 Motivation to validate ontology structures using TDDonto2 Protégé plugin

TDDonto2 is an extension explicitly created to improve the process of ontology authoring using ideas behind Test-Driven Development (TDD). It functions as a plugin for Protégé, which is often used to help make creating or modifying an ontology faster and more precise. According to [36], [37], several reasoners are available to evaluate the ontology structure. Using the automated reasoner can help in the process of ontology authoring faster while employing a test-last methodology. However, the comparison conducted in both studies demonstrates that the TDDOnto2 reasoner is superior in terms of execution speed, task completion, and result accuracy.

Furthermore, the study of [14] highlighted that the algorithms implemented in TDDonto2 significantly improve editing efficiency compared to standard ontology authoring interfaces. Their evaluation showed that users were able to complete tasks more quickly and with fewer errors when using TDDonto2, particularly in the context of medium and large ontologies. As in this study, all of the axioms utilised are followed by the TDDonto2 algorithms suggested by [36], [38].

3. METHODOLOGY

3.1. The SPC_Academic_Performance

The SPC_Academic_Performance outlined in this study is derived from the aggregation of CQs collected through interviews with a selected domain expert. According to [39], [40], the researcher must first ascertain the desired outcome or expected answer before constructing the ontology. The class and the relationship can be extracted and integrated only by understanding the anticipated response to construct the ontology and its SPARQL searches. Figure 1 below illustrates the class and the object properties relationship of the SPC_Academic_Performance.an increasing number of ontologies

Figure 1: Classes and their relationship object properties of SPC_Academic_Performance

The list of the classes and subclass of the SPC_Academic_Performance ontology is illustrated in Figure 2 below. Figure 2 shows the class hierarchy, showing the structure of a class and subclass. Starting from the most generic class, owl: Thing, which serves as the universal superclass encompassing all entities. Every class that is under another class is known as a subclass. For instance, the ‘University’ class is a subclass of the' Person' class. The class ‘Lecturer’ and ‘Student’ are two subclasses further divided from class ‘Person.’ The identical rule applied to the entire class structure. These components, including their hierarchical structure and relationships, will be systematically tested to validate consistency in subsequent stages of the research using the TDDonto2 tool to ensure the ontology's effectiveness and reliability.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

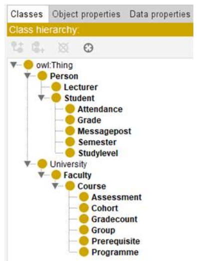

Figure 2: SPC_Academic_Performance ontology class hierarchy

The object properties carefully defined and structured within the SPC_Academic_Performance ontology, along with their domains and ranges of the triples, are comprehensively detailed in Table 2 below.

Table 2: List of class (subject, predicate) and relationship object properties utilised in SPC_Academic_Performance ontology

#	Object properties (P)	Subject (S) / Domain	Object (O) / Range
1	is_a	Person	Student
2	is_a	Person	Lecturer
3	consist_of	University	Faculty
4	has_attendance	Student	Attendance
5	has_grade	Student	Grade
6	has_messagepost	Student	Messagepost
7	has_semester	Student	Semester
8	has_semester	Course	Semester
9	undergraduate_type	Student	Studylevel
10	undergraduate_type	Course	Studylevel
11	has_cohort	Student	Cohort
12	has_cohort	Course	Cohort
13	enrols	Student	Course
14	attached_to	Student	Group
15	belong_to	Student	Faculty
16	belong_to	Lecturer	Faculty
17	teach	Lecturer	Group
18	teach	Lecturer	Course
19	offer	Faculty	Course
20	offered_in	Course	Group
21	has_assessment	Course	Assessment
22	has_programme	Course	Programme
23	has_gradecount	Course	Gradecount
24	optional_to_have	Course	Prerequisite
21	has_assessment	Course	Assessment

In the tested relationship object properties, several object properties: such as ‘is_a’, ‘has_semester’, ‘undergraduate_type’, ‘has_cohort’, ‘belong_to’ and ‘teach’ are used more than once as the exact relationship is employed to link two distinct domains to a single range, following OWL regulations. For instance, using properties such as ‘owl: ObjectProperty’ enables the establishment of relationships between instances of different classes. This means that two distinct classes can have properties that point to the same instance of a third class, effectively allowing multiple domains to connect to a single range class. This is particularly useful in scenarios where different entities share common characteristics or attributes, facilitating interoperability and integration across diverse domains [41], [42]. By utilising TDDonto2, the ontology was subjected to automated reasoning processes to detect potential logical conflicts or inconsistencies arising from incorrect class hierarchies, improperly defined relationships, or ambiguous constraints.

3.2. Testing Setup

3.2.1. Consistency Check

In this study, the consistency of the developed ontology was rigorously evaluated using the TDDonto2 plugin integrated within Protégé, a widely used ontology development environment. The consistency test aimed to ensure that the logical structure of the ontology adhered to formal reasoning principles, mainly focusing on the hierarchical relationships among classes and subclasses.

Table 3: Example test scenario for consistency check

Consistency Test	Validation Formula	Result	Comments
Class Hierarchy	$\forall x$ (Lecturer(x) → Person(x))	Consistent	Subclass relationships valid
Class Hierarchy	$\forall x$ (Student(x) → Person(x))	Consistent	Subclass relationships valid
Class Hierarchy	$\forall x$ (University(x) → Person(x))	Not Consistent	Subclass relationships are not valid

According to the example in Table 3, two hierarchy elements are consistent, as indicated in Figure 2; the class 'Lecturer' and 'Person' are subclasses of the class 'Person'. Simultaneously, the University does not constitute a subclass of Person, indicating a lack of subclass relationship.

3.2.2. Property Validation

Simultaneously, property validation is conducted to examine the correlation of object properties defined in the ontology, ensuring that these properties are accurately defined with suitable domains, ranges, and characteristics (functional and cardinality). The cardinality values employed are determined by the guidelines established during the interview with the domain expert early in the ontology's creation. Examples of functionality and cardinality are illustrated in the following example:

3.2.2.1. Functional

Table 4: Example test scenario for functional property validation

Scenario	Entities	Object Property	Relation
1	Lecture → Person	is_a	The lecturer is a Person
2	Course → Lecturer	teach	Lecturers teach the Course
3	Assessment → Course	has_assessment	The course has an Assessment

3.2.2.2. Cardinality

Table 5: Example test scenario for cardinality property validation

Scenario	Description	DL Axiom	Relation
Minimum Cardinality	Specifies the minimum number of instances allowed	Student ⊑∃ enrols.Course ⊓ ≥ 1 enrols.Course	Each student may enrol in a minimum of one course per semester
Maximum Cardinality	Specifies the maximum number of instances allowed	Lecturer ⊑∃ belong_to.Faculty ⊓ ≤2belong_to. Faculty	Each lecturer can belong to not more than two faculty
Exact Cardinality	Specifies the exact number of instances allowed	Student ⊑∃ belong_to.Faculty ⊓ =1belong_to. Faculty	Each student can belong to only one faculty

Each tested axiom will be sequentially incorporated into the TDDonto2 test list, culminating in a comprehensive evaluation of all axioms collectively.

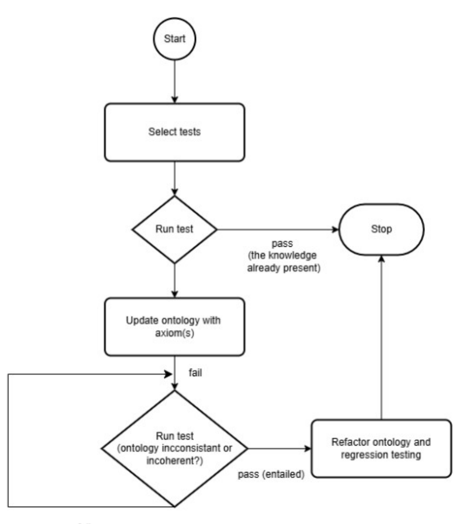

Figure 3: TDDonto2 test flow

If the axiom(s) already exists, the tool will directly ascertain their existence to prevent redundancy of the textual axioms. If axioms are logically correct, the result will indicate entailed; otherwise, it will fail (inconsistent, incoherent, absent). In TDDonto2 testing, only axiom(s) "entailed" yield a pass result, while all other cases produce failures [14].

Given consistent and coherent ontology O, and an axiom A s.t. Σ(A) ⊆ Σ(O), i.e.,

$$\textit{pre-test}_O(A) = \begin{cases} O \nvdash \top \sqsubseteq \bot & \text{ontology is consistent} \\ C \in V_C \text{ s.t. } C \sqsubseteq \neg\bot & \text{ontology is coherent} \\ \Sigma(A) \subseteq \Sigma(O) & \text{axiom vocabulary elements present in } O \end{cases}$$

Then, the result of testing O against A is:

$$\text{test}_O(A) = \begin{cases} \text{entailed} & \textit{if } O \vdash A \\ \text{inconsistent} & \textit{if } O \cup A \vdash \bot \\ \text{incoherent} & \textit{if } O \cup A \nvdash \bot \wedge (\exists C \in \Sigma_C(O)) \wedge O \cup A \vdash C \sqsubseteq \bot \\ \text{absent} & \textit{otherwise} \end{cases}$$

Figure 4: TDDonto2 regression testing logic [14]

In a more straightforward explanation, the axiom(s) must first be present in the ontology during the *pre-testO(A)*, including relevant classes, relationships, or properties, to ensure accurate reasoning and determine if the results are entailed.

The knowledge or axioms are generated based on test-based, test-last [43], and test-first approaches [44] that are often associated with test-driven development. Some axiom(s) for the tests are created before the tests are conducted, and there is a case that some of the axiom(s) are made after the test is run. If there is an error or missing condition, this axiom will be added or modified after the test [39]. The testing cycle ends when all test results fulfill the axioms and entailed.

This study's axiom(s) is tested using the HermiT 1.4.3.456 reasoner. The researcher must revise the axiom(s) if an error occurs during the configuration of the DL and axiom(s). This testing method is adapted from the study of [39]. The test is executed based on Figure 3 above.

4. RESULTS AND DISCUSSION

4.1. Result of the Consistency Check

The axioms utilised for testing the consistency check of the class and subclass structures within the ontology are comprehensively

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

listed in Table 6 below. Additionally, the axioms outlined in Table 7 are designed to validate the properties and their relationships within the ontology. Together, these tests play a critical role in verifying the structural and semantic correctness of the ontology, ensuring it adheres to its intended design principles.

Table 6: List of axioms and results for consistency check

#	Class Hierarchy DL	TDDonto2 Axiom	Relation	Result test: 1	2
1	Student ⊑ Person	Student SubClassOf : Person	“Student” is a subclass of “Person”	✓	✓
2	Lecturer ⊑ Person	Lecturer SubClassOf : Person	“Lecturer” is a subclass of “Person”	✓	✓
3	Attendance ⊑ Student	Attendance SubClassOf : Student	“Attendance” is a subclass of “Student”	✓	✓
4	Grade ⊑ Student	Grade SubClassOf : Student	“Grade” is a subclass of “Student”	✓	✓
5	Messagepost ⊑ Student	Messagepost SubClassOf : Student	“Messagepost” is a subclass of “Student”	✓	✓
6	Semester ⊑ Student	Semester SubClassOf : Student	“Semester” is a subclass of “Student”	✓	✓
7	Studylevel ⊑ Student	Studylevel SubClassOf : Student	“Studylevel” is a subclass of “Student”	✓	✓
8	Faculty ⊑ University	Faculty SubClassOf : University	“Faculty” is a subclass of “University”	✓	✓
9	Course ⊑ Faculty	Course SubClassOf : Faculty	“Course” is a subclass of “Faculty”	✓	✓
10	Assessment ⊑ Course	Assessment SubClassOf : Course	“Assessment” is a subclass of “Course”	✓	✓
11	Cohort ⊑ Course	Cohort SubClassOf : Course	“Cohort” is a subclass of “Course”	✓	✓
12	Gradecount ⊑ Course	Gradecount SubClassOf : Course	“Gradecount” is a subclass of “Course”	✓	✓
13	Group ⊑ Course	Group SubClassOf : Course	“Group” is a subclass of “Course”	✓	✓
14	Prequisite ⊑ Course	Prerequisite SubClassOf : Course	“Prequisite” is a subclass of “Course”	✓	✓
15	Programme ⊑ Course	Programme SubClassOf : Course	“Programme” is a subclass of “Course”	✓	✓

Note: Consistent = ✓, not consistent =X

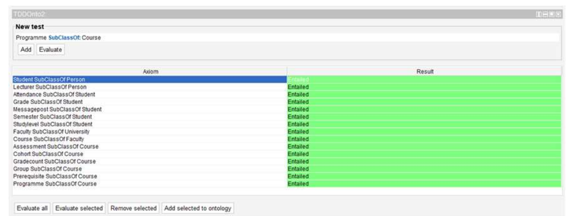

Figure 4: Axioms execution for consistency check

The tests were conducted twice for consistency check, and both tests demonstrated that all of the subclass hierarchy of the evaluated ontology is consistent. Figure 4 shows the result of the consistency check on the ontology structure, and as seen from the result, all the subclasses have ‘Entailed’ results with the superclass it links to. The series of tests will stop once all of the axiom(s) succeed.

4.2. Result of the property validation

4.2.1. Functional

To test the object properties functionality between the subject and object for each relationship, a set of 24 axioms are created and executed, and the test results are illustrated in Table 7 below.

Table 7: List of axioms and results for functional property validation

#	Object Property	TDDonto2 Axiom	Result test: 1	2
1	is_a	Person SubClassOf: is_a some Student	✓	✓
2	is_a	Person SubClassOf: is_a some Lecturer	✓	✓
3	consist_of	University SubClassOf: consist_of some Faculty	✓	✓
4	has_attendance	Student SubClassOf: has_attendance some Attendance	✓	✓
5	has_grade	Student SubClassOf: has_grade some Grade	✓	✓
6	has_messagepost	Student SubClassOf: has_messagepost some Messagepost	✓	✓
7	has_semester	Student SubClassOf: has_semester some Semester	✓	✓

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

8	has_semester	Course SubClassOf: has_semester some Semester	✓	✓
9	Undergraduate_type	Student SubClassOf: undergraduate_type some Studylevel	✓	✓
10	Undergraduate_type	Course SubClassOf: undergraduate_type some Studylevel	✓	✓
11	has_cohort	Course SubClassOf: undergraduate_type some Studylevel	X	✓
12	has_cohort	Student SubClassOf: has_cohort some Cohort	X	✓
13	enrols	Course SubClassOf: has_cohort some Cohort	✓	✓
14	attached_to	Student SubClassOf: enrols some Course	✓	✓
15	belong_to	Student SubClassOf: attached_to some Group	✓	✓
16	belong_to	Lecturer SubClassOf: belong_to some Faculty	✓	✓
17	teach	Lecturer SubClassOf: teach some Group	✓	✓
18	teach	Lecturer SubClassOf: teach some Course	✓	✓
19	offer	Faculty SubClassOf: offer some Course	✓	✓
20	offered_in	Course SubClassOf: offered_in some Group	✓	✓
21	has_assessment	Course SubClassOf: has_assessment some Assessment	✓	✓
22	has_programme	Course SubClassOf: has_programme some Programme	✓	✓
23	has_gradecount	Course SubClassOf: has_gradecount some Gradecount	✓	✓
24	optional_to_have	Course SubClassOf: optional_to_have some Prerequisite	✓	✓

Note: Pass = ✓, not pass =X

Initially, when the axioms are incorporated into the TDDonto2 test list, the outcome will be indicated as 'Absent'. The relationship must be added to the active ontology. In the initial Test 1, both object properties for 'has_cohort' axioms include spelling errors in the syntax determination. However, this error has been rectified in Test 2. Figure 5 below shows the result of the functional of the ontology structure. The result shows that all the relationship and cardinality tests have 'Entailed' results.

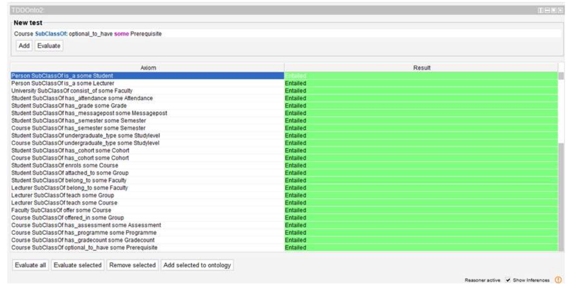

Figure 5: Axioms execution for functional validation

4.2.2. Cardinality

Meanwhile, for the cardinality, all the five cardinalities tested returned green colour passed but "Absent" results as illustrated in Table 8 and Figure 6 for Test 1.

Table 8: List of axioms and results for functional property validation

#	Object Property	TDDonto2 Axiom	Result test: 1
1	belong_to	Student SubClassOf belong_to exactly 1 Faculty	✓ (Absent)
2	enrols	Student SubClassOf: enrols some Course and enrols min 1 Course	✓ (Absent)
3	has_assessment	Course SubClassOf: has_assessment some Assessment and has_assessment min 1 Assessment	✓ (Absent)
4	teach	Lecturer SubClassOf: teach some Group and teach min 1 Group	✓ (Absent)
5	teach	Lecturer SubClassOf: teach some Course and teach min 1 Course	✓ (Absent)

Note: Pass = ✓, not pass =X

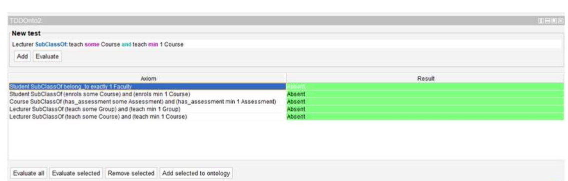

Figure 6: Axioms execution for Test 1 cardinality validation

Concerning the cardinality test from Test 1, a specific new individual is created in each of the axiom(s) classes. This individual is used and acts as an entity to fulfil the cardinality condition. The same axiom(s) are tested again in the second test, the result of which is illustrated in Table 9.

Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 9: List of Test 2 results for cardinality property validation after modification

#	Object Property	TDDonto2 Axiom	Result test: 2
1	belong_to	Student SubClassOf belong_to exactly 1 Faculty	✓ (Entailed)
2	enrols	Student SubClassOf: enrols some Course and enrols min 1 Course	✓ (Entailed)
3	has_assessment	Course SubClassOf: has_assessment some Assessment and has_assessment min 1 Assessment	✓ (Entailed)
4	teach	Lecturer SubClassOf: teach some Group and teach min 1 Group	✓ (Entailed)
5	teach	Lecturer SubClassOf: teach some Course and teach min 1 Course	✓ (Entailed)

Note: Pass = ✓, not pass =X

Figure 7 below shows the result of the functional and cardinality of the ontology structure. The result shows that all the relationship and cardinality tests have 'Entailed' results.

Figure 7: Axioms execution for Test 2 cardinality validation

4.2.3. Result Summary

The values shown in Table 9 illustrate the testing outcomes of two sets of testing performed using TDDonto2, covering the aspect of consistency check and property validation (functional and cardinality characteristics). All the 12 consistency checks were completed in Test 1 and Test 2 successfully with a 100% success rate, hence the consistency ratio of 1. This indicates that the ontology's hierarchical and logical structure of SPC_Academic_Performance is robust and error-free across iterations. While validating the properties, the functional aspect included 24 tests in total. Test 1 yields 22 tests passed on the first try, while test 2 passed all the 24 tests. This resulted into Test 1 which yielded a success rate of 95.8% and Test 2 which had a success rate of 100, and in sum the average consistency ratio of 0.92. For the Cardinality aspect, five tests were accomplished as desired for both set, with a 100% pass rate and 1.0 consistency ratio.

Table 10: Result summary

Validity Aspect	Total Test	Test 1 (Pass)	Test 2 (Pass)	SR (%)	CR
Consistency Checks	12	12	12	100	1.0
Property Validations: Functional	24	22	24	95.8	0.92
Property Validations: Cardinality	5	5	5	100	1.0

Note: Success Rate = SR, Consistency Ratio =CR

These results underline the reliability of the ontology in adhering to logical consistency and property validation requirements. The incremental improvement in the functional property validations demonstrates the effectiveness of iterative debugging and refinement, which is a hallmark of test-driven ontology development. The high levels, success rates, and consistency ratios are a testament to the ontology capability to cater to real-world LA applications.

5. CONCLUSION AND FUTURE WORKS

This study tested the proposed SPC_Academic_Performance ontology reliability and validity using TDDonto2 based on two approaches, primarily testing for consistency checks and property validations (functional and cardinality aspects). The results have confirmed the efficiency of the established ontology structure since all the consistency checks tested on two sets equal 1, showing that the structure is logical and consistent. The success rate for functional property validation also improved from 95.83% in the first test to 100% in the second, showcasing the iterative refinement of the ontology during testing. Cardinality validation consistently maintained a 100% success rate across all tests, reflecting a robust implementation of constraints such as relationships and cardinality rules. As this ontology is developed based on the thematic analysis, with classes and subclasses derived from the CQ. The data retrieved from this ontology will be validated by a domain expert to confirm its alignment with their expectations. This presents an opportunity to examine the data

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

extracted from the proposed ontology with example data sourced directly from the academic performance of students and courses at selected universities. These findings can confirm that the ontology can meet domain-specific requirements while ensuring scalability and reliability for real-world LA applications and data retrieval, especially in analysing the university student and course performance. The current study can be interpreted as the first step in data retrieval validation. Future research could explore the accuracy of data retrieval by applying real-world learner analytics (LA) data and evaluating the results using a confusion matrix to measure accuracy, precision, recall, and F1 score of the data retrieved from the SPC_Academic_Performance ontology.

The guidelines presented in this study can offer direction for ontology validation through agile test-driven methods, addressing consistency and validity tests, which can be applied not only to learning analytics but also to other data retrieval application fields like biological research, artificial intelligence, banking, and healthcare.

6. ACKNOWLEDGMENT

We express our gratitude to the Ministry of Higher Education Malaysia for their financial support of this research (Reference Number: FRGS/1/2021/ICT06/UTEM/01/1). This research was conducted by Pervasive Computing & Educational Technology (PET) Research Group, Centre for Advanced Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka (UTeM) and in collaboration with College of Computing, Informatics and Mathematics Universiti Teknologi MARA, and Web Science Institute, University of Southampton, United Kingdom.

REFERENCES:

[1] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL ontologies," *Semant Web*, vol. 2, pp. 11–21, 2011, [Online]. Available: https://api.semanticscholar.org/CorpusID:4990112

[2] J. Z. Pan and E. Thomas, "Approximating OWL-DL ontologies," in *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, in AAAI'07. AAAI Press, 2007, pp. 1434–1439.

[3] A. Wise and Y. Jung, "Teaching with Analytics: Towards a Situated Model of Instructional Decision-Making," *Journal of Learning Analytics*, vol. 6, pp. 53–69, Aug. 2019, doi: 10.18608/jla.2019.62.4.

[4] D. Vrandečić and A. Gangemi, "Unit tests for ontologies," in *Proceedings of the 2006 International Conference on On the Move to Meaningful Internet Systems: AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET - Volume Part II*, in OTM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 1012–1020. doi: 10.1007/11915072_2.

[5] I. Kotorov, Y. Krasylnykova, M. Pérez-Sanagustín, F. Mansilla, and J. Broisin, "Supporting Decision-Making for Promoting Teaching and Learning Innovation: A Multiple Case Study," *Journal of Learning Analytics*, vol. 11, pp. 21–36, Mar. 2024, doi: 10.18608/jla.2024.8131.

[6] J. Wang, Y. Yang, T. Wang, R. Simon Sherratt, and J. Zhang, "Big data service architecture: A survey," 2020, *Taiwan Academic Network Management Committee*. doi: 10.3966/160792642020032102008.

[7] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993, doi: https://doi.org/10.1006/knac.1993.1008.

[8] T. Helmy and S. Albukhitan, "Framework for Automatic Semantic Annotation of Arabic Websites," *Int J Coop Inf Syst*, vol. 25, no. 01, p. 1650001, 2016, doi: 10.1142/s0218843016500015.

[9] B. Parsia and P. F. Patel-Schneider, "Meaning and the Semantic Web," 2004, doi: 10.1145/1010432.1010513.

[10] T. Venetis, G. Stoilos, G. Stamou, and S. Kollias, "F-DLPs: Extending Description Logic Programs With Fuzzy Sets and Fuzzy Logic," 2007, doi: 10.1109/fuzzy.2007.4295599.

[11] S. Casteleyn, P. Plessers, and O. De Troyer, "Generating Semantic Annotations During the Web Design Process," p. 91, 2006, doi: 10.1145/1145581.1145600.

[12] S. Shrivastava and S. Agrawal, "An Overview of Building Blocks of Semantic Web," *Int J Comput Appl*, vol. 152, no. 10, pp. 17–20, 2016, doi: 10.5120/ijca2016911802.

[13] N. Sharma and A. K. Srivastava, "Ontological Paradigm for Focused Crawling Based on Lexical Analysis," *Int J Comput Appl*, vol. 62, no. 6, pp. 32–36, 2013, doi: 10.5120/10086-4715.

[14] C. M. Keet, K. Davies, and A. Lawrynowicz, "More Effective Ontology Authoring with Test-Driven Development," *arXiv preprint arXiv:1812.06015*, 2018.

[15] M. H. Musa *et al.*, "Integrating and Retrieving Learning Analytics Data from Heterogeneous Platforms Using Ontology Alignment:Graph-Based Approach.," *MethodsX*, p. 103092, 2024, doi: https://doi.org/10.1016/j.mex.2024.103092.

[16] T. Aruna, K. Saranya, and C. Bhandari, "A Survey on Ontology Evaluation Tools," in *2011 International Conference on Process Automation, Control and Computing*, 2011, pp. 1–5. doi: 10.1109/PACC.2011.5978931.

[17] C. V. S. Avila, G. Maia, W. Franco, T. V. Rolim, A. de O. da Rocha Franco, and V. M. P. Vidal, "OntoVal: A Tool for Ontology Evaluation by Domain Specialists.," in *ER Forum/Posters/Demos*, 2019, pp. 143–147.

[18] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa, "OOPS! (OntOlogy Pitfall Scanner!)," *Int J Semant Web Inf Syst*, vol. 10, no. 2, pp. 7–34, 2014, doi: 10.4018/ijswis.2014040102.

[19] M. , Tapia-Leon, C. , C. J. Aveiga, and Suárez-Figueroa, "Ontological model for the semantic description of syllabuses," *In Proceedings of the 9th International Conference on Information Communication and Management*, pp. 175–180, 2019.

[20] H. Kondylakis, V. Kostas, and G. Agathangelos, "Developing an Ontology for Curriculum & Syllabus," *In The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece*, pp. 55–59, 2018, [Online]. Available: www.bbc.co.uk/ontologies/curriculum

[21] R. Fleiner, B. Szász, and A. Micsik, "OLOUD - an ontology for linked open university data," *Acta Polytechnica Hungarica*, vol. 14, no. 4, pp. 63–82, 2017, doi: 10.12700/APH.14.4.2017.4.4.

[22] B. Szász, R. Fleiner, and A. Micsik, "A case study on linked data for university courses," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2017, pp. 265–276. doi: 10.1007/978-3-319-55961-2_27.

[23] G. Ashour, A. Al-Dubai, and I. Romdhani, "Ontology-based Course Teacher Assignment within Universities," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, p. 2020, 2020, [Online]. Available: www.ijacsa.thesai.org

[24] H. J. Cha, S. W. Choi, E. B. Lee, and D. M. Lee, "Knowledge Retrieval Model Based on a Graph Database for Semantic Search in Equipment Purchase Order Specifications for Steel Plants," *Sustainability (Switzerland)*, vol. 15, no. 7, Apr. 2023, doi: 10.3390/su15076319.

[25] K. H. Hwang, H. Lee, G. Koh, D. Willrett, and D. L. Rubin, "Building and querying RDF/OWL database of semantically annotated nuclear medicine images," *J Digit Imaging*, vol. 30, pp. 4–10, 2017.

[26] M. Kharrat, A. Jedidi, and F. Gargouri, "SPARQL Query Generation based on RDF Graph.," in *KDIR*, 2016, pp. 450–455.

[27] A. Pedro, A.-T. Pham-Hang, P. T. Nguyen, and H. C. Pham, "Data-driven construction safety information sharing system based on linked data, ontologies, and knowledge graph technologies," *Int J Environ Res Public Health*, vol. 19, no. 2, p. 794, 2022.

[28] A. Borgida, M. Lenzerini, and R. Rosati, "Description logics for databases," in *The Description Logic Handbook: Theory, Implementation, and Applications*, USA: Cambridge University Press, 2003, pp. 462–484.

[29] D. A. Koutsomitropoulos, D. P. Meidanis, A. N. Kandili, and T. S. Papatheodorou, "OWL-based Knowledge Discovery Using Description Logics Reasoners," in *International Conference on Enterprise Information Systems*, 2006, pp. 43–50.

[30] G. Cota, R. Zese, E. Bellodi, E. Lamma, and F. Riguzzi, "A Framework for Reasoning on Probabilistic Description Logics," in *Applications and Practices in Ontology Design, Extraction, and Reasoning*, IOS Press, 2020, pp. 127–144.

[31] B. Motik, "On the properties of metamodeling in OWL," *Journal of Logic and Computation*, vol. 17, no. 4, pp. 617–637, 2007.

[32] C. Le Duc, M. Lamolle, and O. Curé, "A Decision Procedure for with Transitive Closure of Roles," in *International Semantic Web Conference*, 2013, pp. 264–279.

[33] I. Horrocks and P. F. Patel-Schneider, "Reducing OWL entailment to description logic satisfiability," in *International semantic web conference*, 2003, pp. 17–29.

[34] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language," *Journal of web semantics*, vol. 1, no. 1, pp. 7–26, 2003.

[35] G. De Giacomo and M. Lenzerini, "TBox and ABox Reasoning in Expressive Description Logics," in *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR'96)*, Nov. 1996, pp. 37–48.

[36] K. Davies, C. Keet, and A. Ławrynowicz, "TDDonto2: A Test-Driven Development Plugin for Arbitrary TBox and ABox Axioms," Aug. 2017, pp. 120–125. doi: 10.1007/978-3-319-70407-4_23.

[37] K. Davies, C. M. Keet, and A. Lawrynowicz, "More effective ontology authoring with test-driven development and the TDDonto2 Tool," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 7, Nov. 2019, doi: 10.1142/S0218213019500234.

[38] A. Fernández-Izquierdo and R. García Castro, "Themis: a tool for validating ontologies through requirements," Mar. 2019, pp. 573–578. doi: 10.18293/SEKE2019-117.

[39] J. Potoniec, D. Wisniewski, and A. Ławrynowicz, "Incorporating presuppositions of competency questions into test-driven development of ontologies," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, Knowledge Systems Institute Graduate School, 2021, pp. 437–440. doi: 10.18293/SEKE2021-165.

[40] L. Zemmouchi-Ghomari and A. R. Ghomari, "Translating Natural Language Competency Questionsinto SPARQL Queries: a Case Study," *WEB 2013 : The First International Conference on Building and Exploring Web Based Environments*, pp. 81–86, 2013, [Online]. Available: https://www.researchgate.net/publication/273382915

[41] R. Hoehndorf, L. T. Slater, P. N. Schofield, and G. V Gkoutos, "Aber-Owl: A Framework for Ontology-Based Data Access in Biology," *BMC Bioinformatics*, vol. 16, no. 1, 2015, doi: 10.1186/s12859-015-0456-9.

[42] L. T. Slater, G. Gkoutos, and R. Hoehndorf, "Towards Semantic Interoperability: Finding and Repairing Hidden Contradictions in Biomedical Ontologies," 2020, doi: 10.1101/2020.05.16.099309.

[43] A. Tahir, S. Counsell, and S. G. MacDonell, "An Empirical Study Into the Relationship Between Class Features and Test Smells," 2016, doi: 10.1109/apsec.2016.029.

[44] S. Gundlach, R. Jung, and W. Hasselbring, "Facilitating Test-Driven Development via Domain-Specific Languages in Computational Science Software Engineering," 2023, doi: 10.21203/rs.3.rs-3753364/v1.