University of Ibadan
Journal of Science and Logics in ICT Research

An Activity Ontology for the Conceptualization of Exploratory Software Testing

[1]Ibitowa F. O., [2]Ayorinde I. T., [3]Akinola S. O. and [4]Oyedeji O. A.

[1]Department of Computer Studies, The Polytechnic, Ibadan, Ibadan., Nigeria
[2,3,4]Department of Computer Science, University of Ibadan, Ibadan, Nigeria.

[1]ibitowafolashade@yahoo.com, [2]temiayorinde@yahoo.com
[3]solom202@yahoo.co.uk [4]seyioyedeji31@gmail.com

Abstract
Exploratory Software Testing (EST) permits testers to interact with a system based on knowledge, skill, creativity, expertise, inspiration, experience and intuition in order to find bugs/defects while ontology explicitly specifies the terms in a domain and the relationship between them. The limitation of EST experiences to individual testers and the inability to share the tester's knowledge within the software organisation have resulted in conceptual ambiguities and a low reuse rate of EST knowledge. Therefore, this study develops a formal activity ontology for different EST knowledge in software organisations for uniform vocabulary and knowledge management. The ontology engineering approach was used in modelling the ontology. Elicitation of knowledge for EST was carried out in an experimental test environment using 150 testers. Through the use of Description Logic, knowledge is transformed through knowledge formalization into a logical form (axioms). A web ontology language was used to implement the logical axioms. The EST ontology was evaluated with formulated competency questions that validated its correctness. It can be used as a reference model in software organisations as well as a knowledge base for software testers. It can also be reused by professionals in the domain of software testing.

Keywords: *Exploratory software testing, Formal ontology, Description logic, Web ontology language*

1. Introduction

Ontology is defined as an explicit specification of a conceptualization [1]. Ontologies describe concepts in a specific field of knowledge along with their properties and constraints. To facilitate communication, integration, storage, search, sharing, and reuse of knowledge representation, an ontology offers a precise description of knowledge in a formal language [2]. The need for greater interoperability and reuse of information between systems and people within an organization is one of the main interests in ontologies [3]. Ontologies are important for knowledge management. According to Davenport and Prusak [4], knowledge management is a process that streamlines the procedure of distributing, sharing, capturing, creating and understanding of knowledge. It has been viewed as a source of competitive advantage for organizations.

Software testing is an activity that evaluates a program's quality and also improves it by identifying and correcting defects and problems [5]. Exploratory Software testing involves human testers interacting with a system in accordance with their knowledge, skill, creativity, expertise, inspiration, experience and intuition with the ability to detect bugs and this has been referred to as concurrent learning, test design, and test execution [6]. Exploratory software testing makes use of human judgment to determine whether a feature is functional and it is possible to concentrate testing efforts on areas of an application that are thought to be more likely to have bugs.

Ibitowa F. O., Ayorinde I. T., Akinola S. O. and Oyedeji O. A. (2023). An Activity Ontology for the Conceptualization of Exploratory Software Testing, *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 10 No. 1, pp. 69 – 84.

Exploratory software testing is accepted in the industry and regarded as a successful method of detecting bugs [7]. Exploratory software testing, according to practitioner literature, also lowers the burden of creating and maintaining documentation, aids team members in understanding the features and behavior of the application being developed, and enables testers to immediately concentrate on useful areas during testing [8].

The main issues in software organizations are low reuse rates of exploratory software testing knowledge, serious loss of exploratory software testing knowledge, poor sharing of exploratory software testing knowledge, and incomplete transfer of knowledge. These occur because knowledge in organizations is not treated, making it difficult to articulate it.

This study develops a formal activity ontology for an exploratory software testing approach for knowledge reuse and an easy retrieval system. This is achieved by building a knowledge base of software exploratory testing, building an activity ontology for it and validating the ontology's accuracy using knowledge based reasoner. Knowledge reuse in the area of EST is highly needed for organizational growth. This helps in shared knowledge representation of the entities, activities, relationships and every action required to carry out exploratory software testing.

2. Related Works

Olszewska [9] in Software Testing Ontology For AI-Based Systems develops an ontological framework for AI-Software Testing (AI-T). This domain includes both software testing and comprehensible artificial intelligence; the objective is to create an ontology that guides the testing of AI software, in a way that is efficient and operative. For this purpose, AI-T ontology includes temporal interval logic modeling of the software testing process as well as the ethical principle of formalization and has been built using the Enterprise Ontology (EO) methodology resulting AI-T ontology proposes both conceptual and implementation models and contains 708 terms and 706 axioms.

Hassnain *et.al.* [10] in an Ontology Based Test Case Prioritization Approach in Regression Testing explained that regression testing has received a lot of research attention, in an effort to address the issues with software system quality. The study provides insight into proposing the ontology-based TCP (OTCP) approach, with the intention of using fewer resources to maintain and improve the quality of software systems.

The suggested method analyzes the behavior of various classes of software systems using software metrics. To ensure that predictions of the faulty and non-faulty classes of software systems are accurate, Binary Logistic Regression (BLR) and AdaBoostM1 classifiers are used. Code metrics and class attributes are matched using a reference ontology. They looked into five Java programs to evaluate the suggested strategy for achieving code metrics. An average percentage of faults was discovered as a result of this study (APFD) value of 94.80%, which is higher when compared to other TCP approaches. They suggested that large sized programs in different languages can be used to evaluate the scalability of the proposed OTCP approach in the future.

Chimalakonda and Nori [11] in an Ontology based modeling framework for the design of educational technologies stated that despite the fact that educational technology has developed quickly, strong instructional design knowledge is still lacking, which has an impact on how well instruction is designed. Based on domain patterns, the research recommends an ontology-based framework for systematic modeling of different facets of instructional design knowledge.

As part of the framework, they offered ontologies for modeling instructional objectives, instructional procedures, and instructional content. They demonstrated the ontology framework by presenting instances of the ontology for the large-scale case study of adult literacy in India (287 million learners spread across 22 Indian Languages), which requires the creation of numerous identical but distinct eLearning systems based on flexible instructional designs. The proposed framework could be used to represent instructional design proficiency for academic learning, career skills, and other purposes.

Ayorinde [12] in a Formalised Ontology of Musical Instruments stated that an aspect of

knowledge representation known as formal ontology deals with the formal conceptualization of domains. The sharing of a common understanding of the structure of information is improved by ontology, it is a tool that can be used in training. Hence, the article creates a formalized ontology of musical instruments based on Hornbostel and Sach's classification system. Information about the families, groups, and characteristics of musical instruments, such as their shape and playing style, is provided by the ontology, among others. On the basis of the user's preferences, it also suggests which instrument to learn. Predicate logic is used to formalize ontology concepts and their relationships, and the prolog programming language is used to implement them.

Tebes *et. al.* [13] explained that testing is one of the areas of Software Engineering (SE) that supports quality assurance. Given that specific software testing processes, artifacts, methods, and ultimately strategies involve a lot of domain concepts, It is advantageous to have a solid conceptual base, that is, a software testing ontology that explicitly and unambiguously defines the terms, properties, relationships, and axioms. After examining both the findings of a primary study's Systematic Literature Review (SLR) on software testing ontologies, and the state-of-the-art of test-related standards, they chose to create a top-domain ontology that meets their objectives. In the framework of a four-layered ontological architecture, TestTDO was created, which takes into account domain, instance, core, and foundational ontologies. In the paper, the development, assessment, verification, and validation of the TestTDO conceptualization were topics of discussion.

Mårtensson *et.*al [14] interviewed 20 people from four case study companies. The paper listed a number of crucial elements that make efficient and productive exploratory testing of substantial software systems possible. The four themes are comprised of the nine factors: “The testers’ knowledge, experience and personality”, “Purpose and scope”, “Ways of working” and “Recording and reporting”. According to the interviewees, exploratory testing allows testers to be more creative in their work, and was therefore thought to utilize the testers more effectively. A series of follow-up interviews with 20 interviewees and a cross-company workshop with 14 participants confirmed the key factors that had been identified. This improves the findings' generalizability, proving that a large segment of the software industry can use the list of key factors to guide projects. The study also includes 129 publications related to exploratory testing as part of its systematic literature review. There is no article that lists the essential elements that make exploratory testing efficient and effective, which affirms that the findings are novel.

Ayorinde and Oyedeji [15] in an Ontology for Intra-Campus Transport System (ICTS) stated that the efficiency and effectiveness of an organization can be greatly increased by knowledge representation, business or firm if its advantages are properly tapped. The Intra-Campus Transport System (ICTS), which manages the movement of people inside an institution, can act as a model for transportation systems.

The study develops a formal ontology for the intra-campus transport system which improves the availability of knowledge and a quick retrieval system. The research work used an extremely thorough knowledge engineering approach. It entails gathering knowledge, also referred to as ontology capture, which provides significant factual information about the ICTS domain. Knowledge analysis and refinement follow this, which involves classifying the gathered knowledge into classes, properties and individuals. With the aid of description logic tools, knowledge formalization transforms the refined knowledge into a logical form. Lastly, Protégé 5.0 was used to implement the logical axioms in web ontology language (OWL) format.

3.0 Methodology

In the first stage, the exploratory software testing knowledge was captured, filtered, refined and classified into facts, definitions and rules/constraints. This is followed by knowledge formalization, which involves the representation of the analysed facts, definitions and constraints using mathematical method called description logic.

Web Ontology Language (OWL) was used to represent the logical expression that was formed

based on the information gathered in a machine-readable form. The inference engine was used to help the ontology to make explicit statements and infer new knowledge and deductions. The ontology was tested with competency questions for effectiveness / accuracy. The competency questions formulated for the ontology are problems that are expected to be solved by the ontology. Answering these questions by the ontology showed the intelligence ability of the ontology. The knowledge can then be shared, reused and applied. It can also be transferred to the testers as new knowledge. Figure 1 shows the architecture of the system while the specific processes are broken down into the goal and scope definition of the ontology, information gathering and elicitation, initial structuring, formalisation, deployment and evaluation.

3.1 The Goal and Scope of the Ontology

This phase is regarded as the starting point of any ontology development cycle. It is largely a preparatory phase concerned with the identification of the domain and subject area being studied. It also lists the aims and objectives as well as the high-level specifications of the exploratory test ontology. The exploratory software testing (EST) activity ontology developed in this study is based on representing the knowledge involved in the exploratory software testing domain. It specifically represents the knowledge involved in the process of carrying out the exploratory software testing.

3.2 Information Gathering and Elicitation

This phase helps to achieve a deeper understanding of the exploratory testing domain. Software Testers came together to brainstorm and engage in discussion through conversations, e-chatting, e-messaging, teleconferencing and meetings. During the process, exploratory software testing knowledge was gathered and captured using a specific software that was developed.

3.2.1 Establishment of Facts about the Processes involved in Exploratory Software Testing

The facts gathered constitute the knowledge of the exploratory software testing domain. Some are highlighted below:

1. A tester is obligated with the duty of carrying out a software test on a given software in question.
2. The specific type of testing in relation to this research work is the exploratory software testing.
3. Exploratory software testing is a kind of testing that makes use of the tester's cognitive ability to detect errors/defects in the testing stage of software development
4. The details of exploratory software testing are critically reviewed in this research procedure
5. An exploratory software testing is preceded with a set of instructions and guidelines
6. An exploratory software tester carries out the exploratory software testing which relies solely on the discretion of the tester.

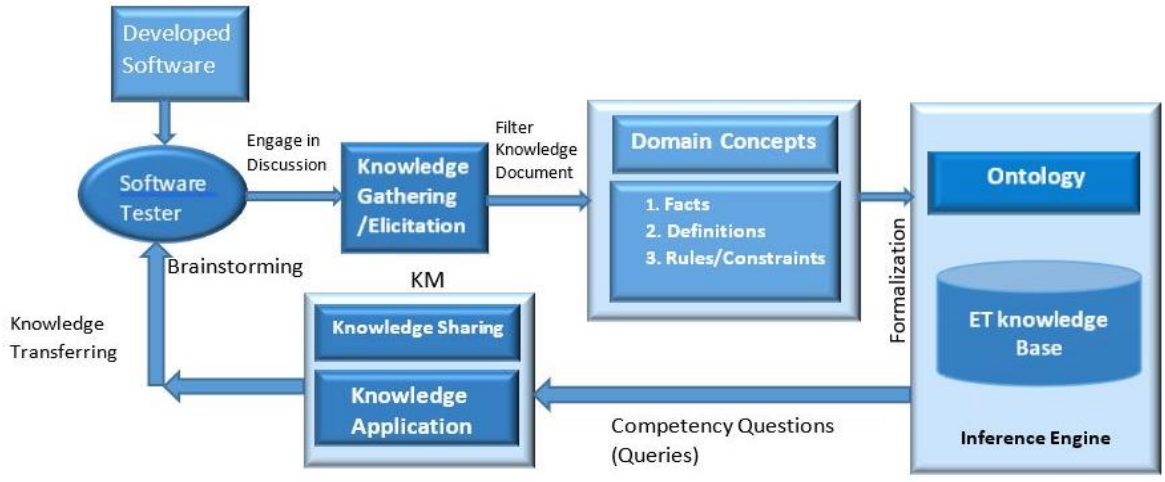

Figure 1: Architecture of the EST Ontology

7. The exploratory software testing consists of three basic sessions which are: Pre-survey, Main exploratory software test and Post-survey
8. An exploratory software testing consists of/makes use of a test charter
9. The pre-survey session consists of a pre-survey form filled by the exploratory software tester
10. The pre-survey form captures information such as: Tester name, Sex, Age, Level of education, Current working role, Years of working, Area of specialization, Programming language specialization, Familiarity of exploratory software testing, Knowledge of software testing, Training on software testing and How frequent is software testing.
11. The post-survey session consists of a post-survey form filled by the exploratory software tester
12. The post-survey form captures information such as: Name, Number, Testing Coverage, Overall software quality, Ease of exploratory software test approach, Usefulness of test charter for structuring and guiding, Usefulness of test charter for finding defects, Effect of allotted time and Challenges faced.
13. The test survey is used to describe the pre-survey and the post-survey aspects of the test.
14. The test charter consists of a defect report and test log
15. The test charter is also referred to as the test plan
16. The Exploratory software tester fills the test charter during the test
17. The test charter consists of a charter form filled by the Exploratory software tester
18. The charter form captures information such as: Charter's name, Tester's name, Tester's number, Start date, Start time, Duration, Purpose, Test reference, Priority, Data, Testing notes/ideas, Defect report, Issues/challenges
19. The defect report consists of a defect report sheet which is filled by the exploratory software tester
20. The defect report sheet captures information such as: Description of defect found, Defect severity, Defect type and Defect mode.

3.3 Initial Structuring

This phase is about getting the necessary knowledge out of the gathered information and using information organization methods for finding trends, rationalizing and structuring the information. Methods used involve the extraction of concepts from the established facts and establishing relationships between the concepts.

3.3.1 Extraction of Concepts in the Established Facts E.G.

The various concepts established from the EST domain are: Person, Tester, Human resource, Software test, Software resource, Software analyst, Programmer, Analyst, End-user, Hardware resource, Exploratory software test, Exploratory software tester, Exploratory software testing, Test instructions and guidelines, Test Session, Test survey, Test plan, Error, Bug, Defect, Pre-survey, Main Exploratory software test, Post-survey, Pre-survey form, Post-survey form, Charter form, User specification, User guide, Test charter, Defect report, Defect report sheet, Test log, User manual, Software specification, Test environment, Actual result, Expected result, Exploratory Software Testing Activity, Test execution, Test result, Tester ID, Tested features, Detected defect, Test deliverables, Fatal severity, System usage and Data loss among others.

3.3.2 Extraction of Relationships between Concepts in the Established Facts

Some of the relationships are given below:

1 Exploratory software test is a type of software testing method
2 Exploratory software tester carries out Exploratory software test
3 Exploratory software tester detects defects/bug in an Exploratory software test
4 A tester is a software tester
5 A software tester is a person
6 An exploratory software tester is a software tester
7 A bug is also known as a defect
8 A defect is also known as an error
9 An exploratory software tester makes use of hardware resource during an exploratory software test

10 An exploratory software tester performs main exploratory software testing
11 A programmer is a tester
12 A tester loads the software to be tested during an exploratory software test
13 Main exploratory software test requires the test charter during exploratory software testing.
14 A tester produces a test plan as artefact
15 A test charter defines at least one Test activity
16 The test charter is represented with the charter form
17 Charter form is a testing artefact
18 Defect is described in the defect report
19 Defect report is represented with a defect report sheet
20 Defect report sheet is a testing artefact
21 Specification document is a testing artefact
22 Test data is a testing artefact
23 Test instruction is a testing artefact
24 Test log is a testing artefact
25 Test plan is a testing artefact
26 Test survey is represented with test survey form
27 Pretest survey is represented with pretest survey form
28 Posttest survey is represented with posttest survey form
29 Pretest survey form is a test survey form
30 Posttest survey form is a test survey form
31 Test survey form is a testing artefact
32 User guide is a testing artefact
33 User manual is a testing artefact
34 User specification is a testing artefact
35 Tester must prepare a test charter
36 Test charter must have a name
37 Test survey form consist of pretest survey form and posttest survey form -
38 Tester requires an academic qualification
39 The academic qualification must be in computer science or in computer related field
40 The academic qualification must be in computer science or in computer related field
41 The academic qualification must be a Tertiary Qualification
42 A tertiary qualification can be National Diploma, Higher National Diploma, Bachelors Degree or Masters degree
43 Testing process needs a Testable Entity as input.
44 Test environment composed of Test hardware resource
45 Test environment composed of Test software resource
46 Test environment composed of human resource

3.4 Formalisation

In the formalization stage, the facts and relationships are transformed into description logic axioms and web ontology language (owl). This stage also involves the use of a suitable ontology development environment (ontology editor, rule engine, etc.) to encode, refine and test the initial structures as a formal model expressed in a chosen knowledge representation formalism. Classes, relations and logic among others are captured.

3.4.1 Formalization of Domain Knowledge using Description Logic

Some of the description logic axioms are shown below:

1. Exploratory software test is a type of software testing method
   ExploratoryTesting ⊆ SofwareTesting
2. Exploratory software tester carries out Exploratory software test
   ExploratoryTester ≡ Person ⊓ ∃carriesOut.ExploratoryTesting
3. Exploratory software tester detects defects/bug in an Exploratory software test
   ExploratoryTester ≡ Person ⊓ ∃detects.Bug
4. A tester is a software tester
   Tester ⊆ SofwareTester
5. A software tester is a person
   SofwareTester ⊆ Person
6. An exploratory software tester is a software tester
   ExploratoryTester ⊆ SofwareTester
7. A bug is also known as a defect
   Bug ≡ Defect
8. A defect is also known as an error
   Defect ≡ Error
9. An exploratory software tester makes use of hardware resource during an exploratory software test
   ExploratoryTester ≡ Person ⊓ ∃makesUseOf.Hardware

10. An exploratory software tester performs main exploratory software test
    ExploratoryTester ≡ Person ⊓ ∃performs.MainExploratoryTest
11. An analyst is a Tester
    Analyst ⊆ Tester
12. A programmer is a tester
    Programmer ⊆ Tester
13. An end user is a tester
    EndUser ⊆ Tester
14. A software analyst is a tester
    SoftwareAnalyst ⊆ Tester
15. A tester loads the software to be tested during an exploratory software test
    ExploratoryTester ≡ Person ⊓ ∃loads.Software
16. Main exploratory software test requires the test charter during exploratory software testing
    MainExploratoryTest ≡ ∃requires.TestCharter
17. The test charter is represented with the charter form
    TestCharter ≡ ∃representedWith.CharterForm
18. Charter form is a testing artefact
    CharterForm ⊆ TestingArtefact
19. Defect is described in the defect report
    Defect ≡ ∃describedIn.DefectReport
20. Defect report is represented with a defect report sheet
    DefectReport ≡ ∃representedWith.DefectReportSheet
21. Defect report sheet is a testing artefact
    DefectReportSheet ⊆ TestingArtefact
22. Specification document is a testing artefact
    SpecificationDocument ⊆ TestingArtefact
23. Test data is a testing artefact
    TestData ⊆ TestingArtefact
24. Test instruction is a testing artefact
    TestInstruction ⊆ TestingArtefact
25. Test log is a testing artefact
    TestLog ⊆ TestingArtefact
26. Test plan is a testing artefact
    TestPlan ⊆ TestingArtefact
27. Test survey is a form of survey that is represented with tested survey form
    TestSurvey ≡ Survey ⊓ ∃representedWith.TestSurveyForm
28. Pretest survey is represented with pretest survey form
    PreTestSurvey ≡ ∃representedWith.PreTestSurveyForm
29. Posttest survey is represented with posttest survey form
    PostTestSurvey ≡ ∃representedWith.PostTestSurveyForm
30. Pretest survey form is a test survey form
    PreTestSurveyForm ⊆ TestSurveyForm
31. Posttest survey form is a test survey form
    PostTestSurveyForm ⊆ TestSurveyForm
32. Test survey form is a testing artefact
    TestSurveyForm ⊆ TestingArtefact
33. User guide is a testing artefact
    UserGuide ⊆ TestingArtefact
34. User manual is a testing artefact
    UserManual ⊆ TestingArtefact
35. User specification is a testing artefact
    UserSpecification ⊆ TestingArtefact

3.4.2 Ontology Coding with Web Ontology Language (OWL)

The ontology representation begins with the heading of the ontology code. It describes the syntax of the Resource Description Framework (RDF), Web Ontology Language (OWL) and Extensible Markup Language (XML) schema that is being used to build the ontology. This consists of the ontology International Resource Identifier (IRI) which points to the location of the ontology.

3.4.2.1 Class Representation

This is the first step in the ontology coding. The concepts that have been identified in the previous stages are encoded. Example is shown below:

Concept: **Person**

```
<Declaration>
    <Class IRI="#Tester"/>
</Declaration>
```

3.4.2.2 Class Hierarchy Representation

This is used to model sub classes and super class i.e. to represent divisions and categories that exists between concepts.

Class hierarchy: **Person (software tester, Programmer etc.)**

```
<SubClassOf>
        <Class IRI="#Tester"/>
        <Class IRI="#Person"/>
</SubClassOf>
```

3.4.2.3 Object Property Representation

The object properties provide links between concepts and classes e.g.

Object property: **Defect found in**

```
<SubObjectPropertyOf>
    <ObjectProperty IRI =
"#defectFoundIn"/>
    <ObjectProperty
abbreviatedIRI="owl:topObjectProperty"/>
</SubObjectPropertyOf>
```

Object property: **Must include**

```
<SubObjectPropertyOf>
    <ObjectProperty IRI="#mustInclude"/>
    <ObjectProperty
abbreviatedIRI="owl:topObjectProperty"/>
</SubObjectPropertyOf>
```

3.4.2.4 Data Type Representation

Relationship between classes and data values e.g.

Data property: **Charter name**

```
<Declaration>
    <DataProperty IRI="#charterName"/>
</Declaration>
```

Data property: **Tester name**

```
<Declaration>
    <DataProperty IRI="#testerName"/>
</Declaration>
```

4. Implementation and Result

4.1 Deployment

This phase is concerned with ontology publishing and release as well as scaling into an ontology-driven system. In the deployment phase, knowledge graphs are built and necessary outputs are generated. Protégé 5.5.0 is used to implement the ontology.

4.1.1 Class Implementation

The exploratory software testing ontology classes and its hierarchy is shown in Figure 2.

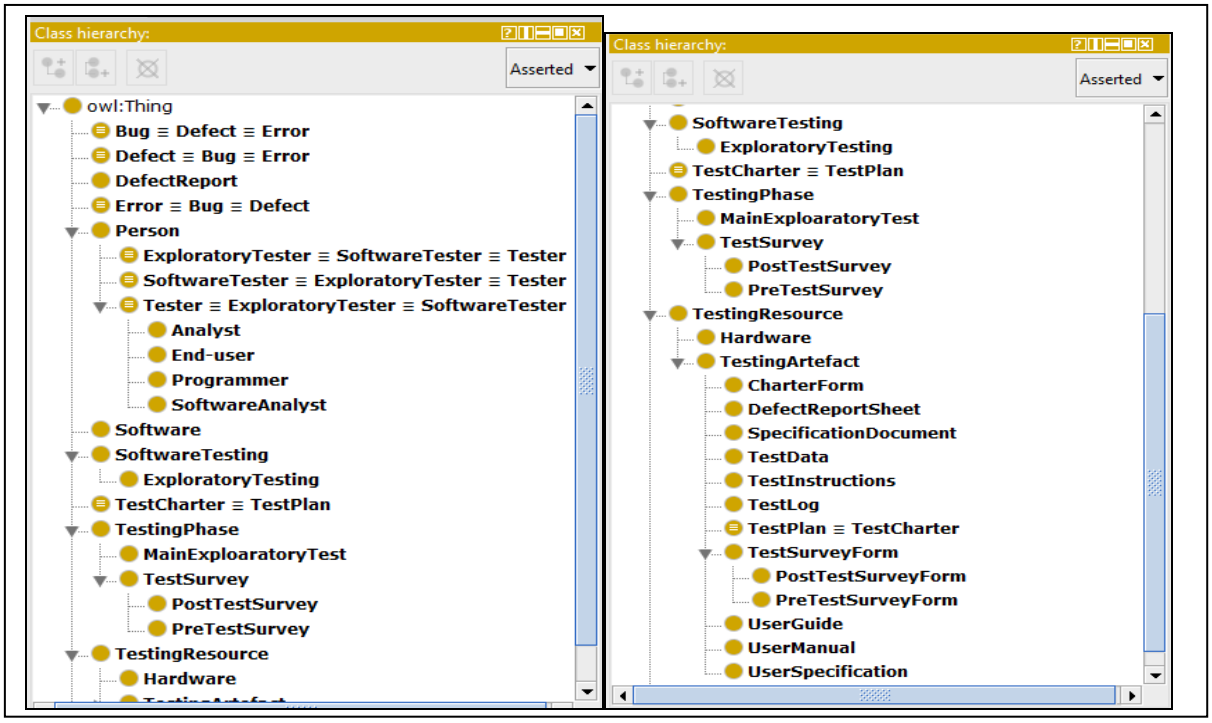

Figure 2: Exploratory software testing ontology class view/class hierarchy

4.1.2 Object Properties Implementation

The exploratory software test object properties is shown in Figure 3.

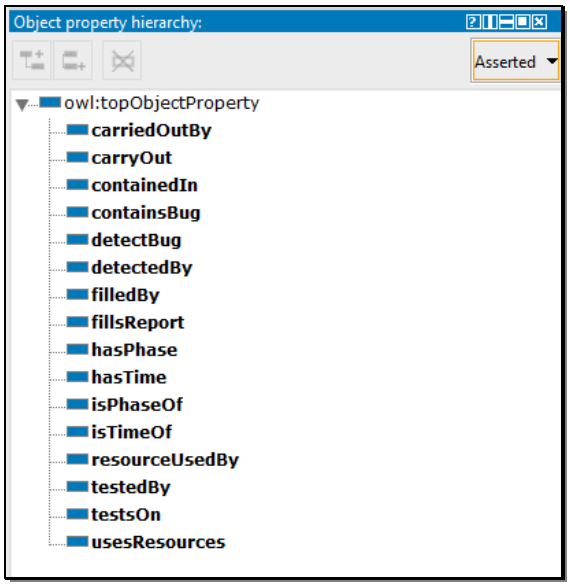

Figure 3: Exploratory software testing ontology object properties representation

4.1.3 Data Properties Implementation

The exploratory software testing data properties representation is depicted in Figure 4.

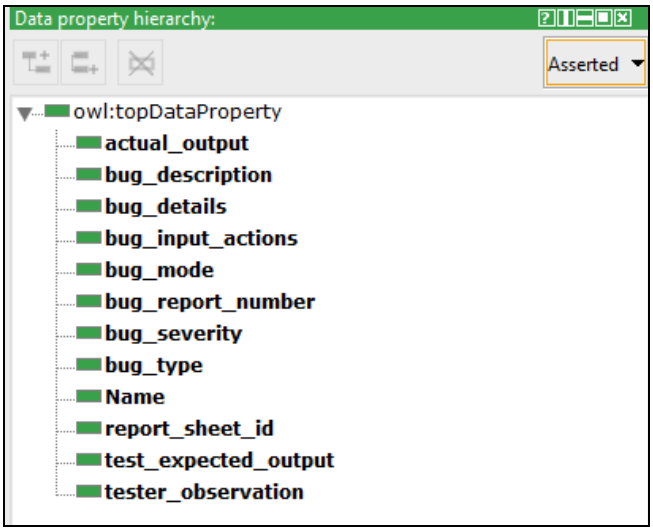

Figure 4: Exploratory software testing ontology data properties representation

4.1.4 Individual Implementation

The exploratory software test instances is depicted in Figure 5

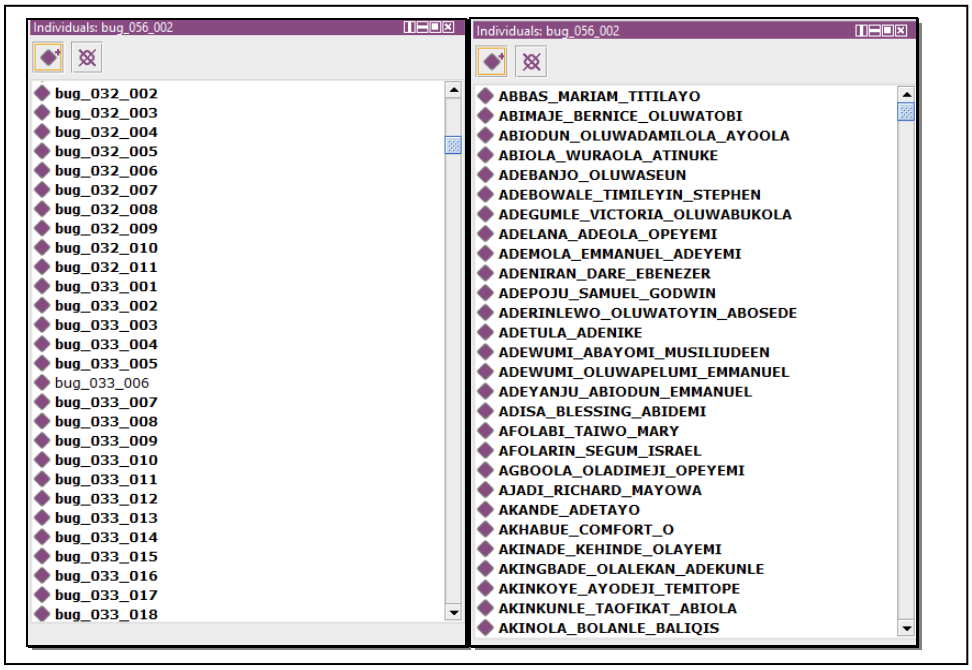

Figure 5: Exploratory software testing ontology with Instances

4.1.5 Ontograph

Two of the graphical views of the exploratory software test ontology is shown in Figures 6 and 7.

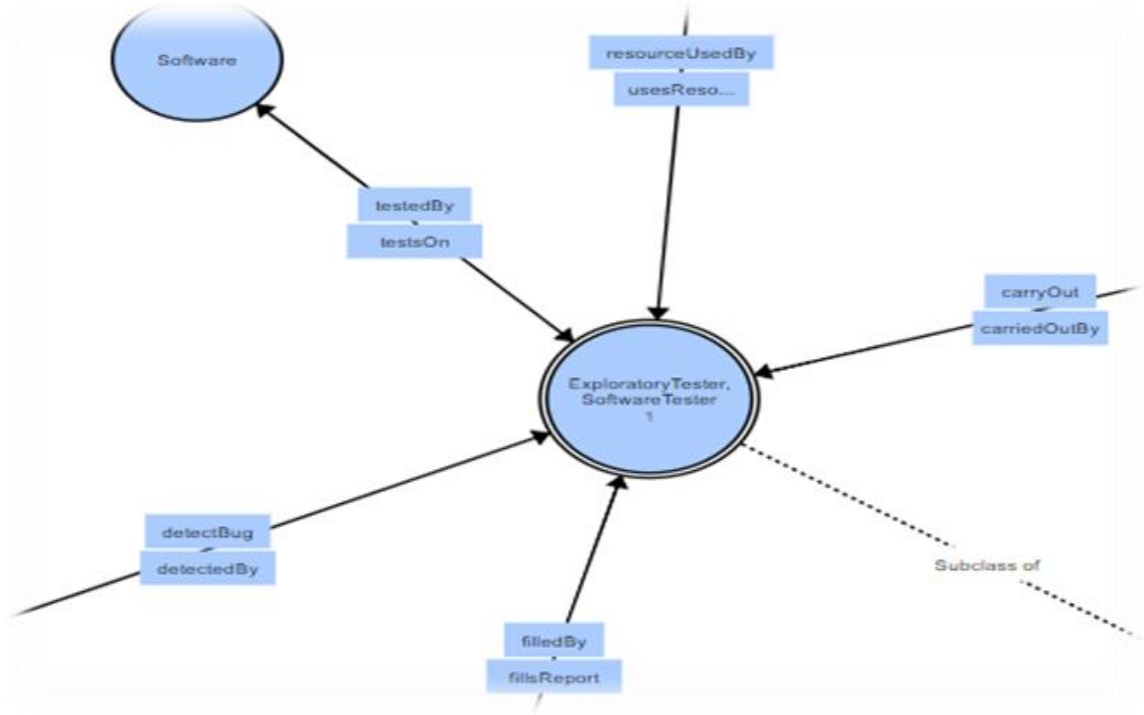

Figure 6: Exploratory tester Ontograph

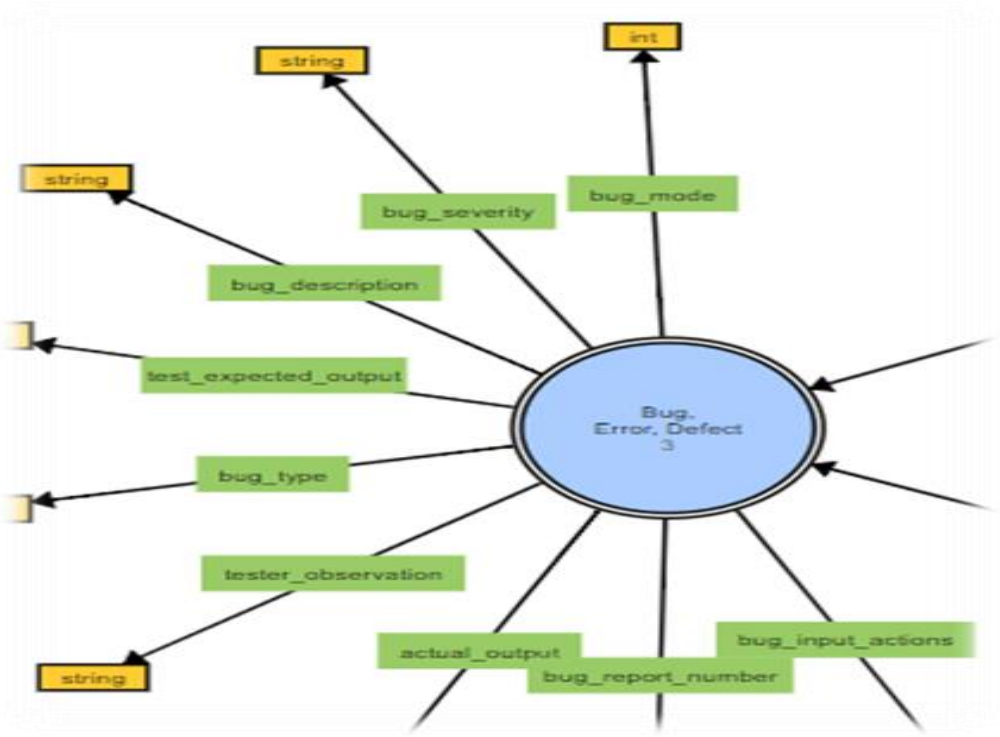

Figure 7: Bug Ontograph

4.2 Evaluation

This phase looks back at the goal and scope definition of the ontology and assesses the extent to which the aims and objectives have been fulfilled and the requirements met in the context of the established scope.

4.2.1 Queries in form of Competency Questions

Competency Questions (CQ) are user generated questions that are consciously created in order to probe the ontology. The purpose of this is to eventually retrieve certain answers which are then checked against the actual knowledge that was used in creating the ontology and this can be likened to the popular Turing test in the artificial intelligence domain. Examples of the competency questions used include but not limited to the following:

i. What are the different categories of testers involved in the exploratory software testing?
ii. What are the various types of people that can be found within the exploratory software test context?
iii. In any order, what are the major stages involved in the Exploratory Software Testing procedure?
iv. What are the types of test survey involved in the test process?
v. What are the various artefacts involved in the testing process?
vi. Who are the specific testers within the research and which bugs (represented with their respective bug codes) were detected by each of them?
vii. Outline the descriptions of the bugs detected by the explorative testers for the bugs during the test procedure.

4.2.2 Results

The Software/Module used for implementing the competency questions within the protégé application are DL QUERY and SPARQL. The DL QUERY supports quite simple query syntax while SPARQL supports more complex query patterns similar to the structured query language (SQL). The Prefix used for building the query is shown in Figure 8:

```
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX io: http://www.semanticweb.org/ibito/ontologies/2019/11/exploratory-test-ontology
```

Figure 8: The Prefix Used for Building the Query

4.2.2.1 Result for the Query: What are the different categories of testers involved in the exploratory software testing? DL QUERY: Tester

The result for this query is shown in Figure 9.

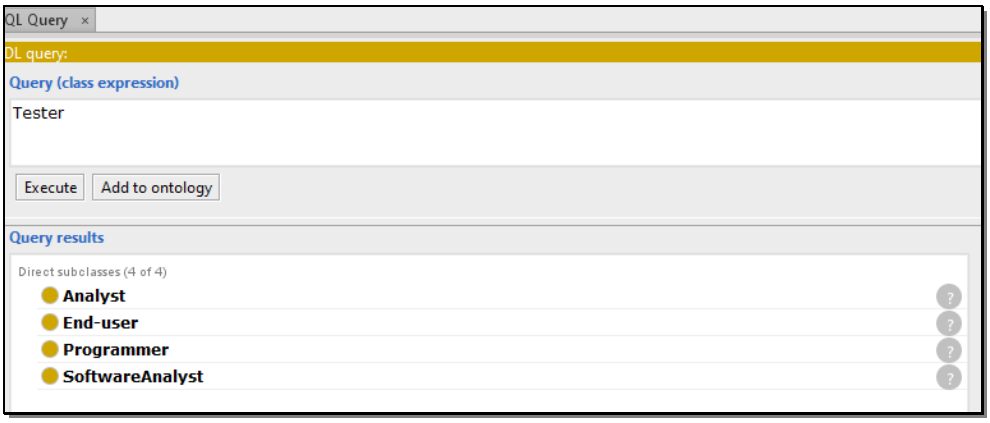

Figure 9: The result containing tester categories
Remark: Result is accurate.

4.2.2.2 Result for the Query: What are the various types of people that can be found within the exploratory software test context? DL QUERY: Person

The result for this query is shown in Figure 10.

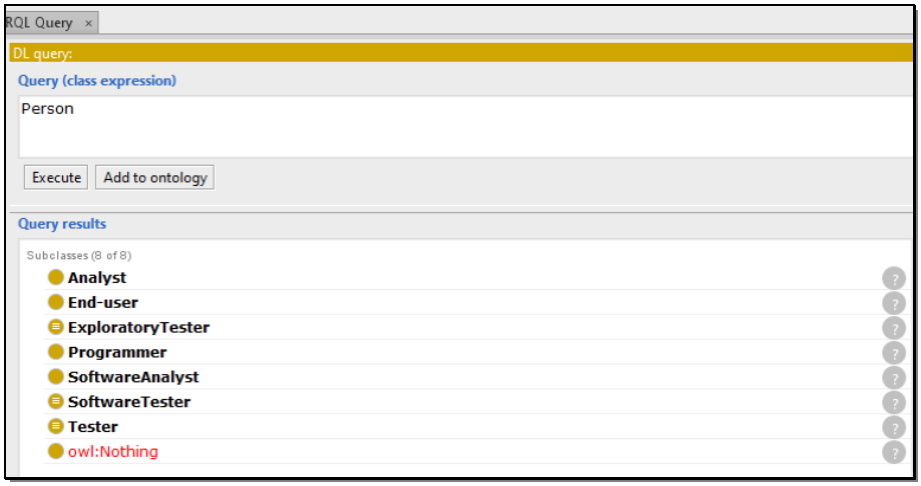

Figure 10: The result containing person categories
Remark: Result is accurate.

4.2.2.3 Result for the Query: In any order, what are the major stages involved in the Exploratory Software Testing procedure? DL QUERY: TestingPhase

The result for this query is shown in Figure 11.

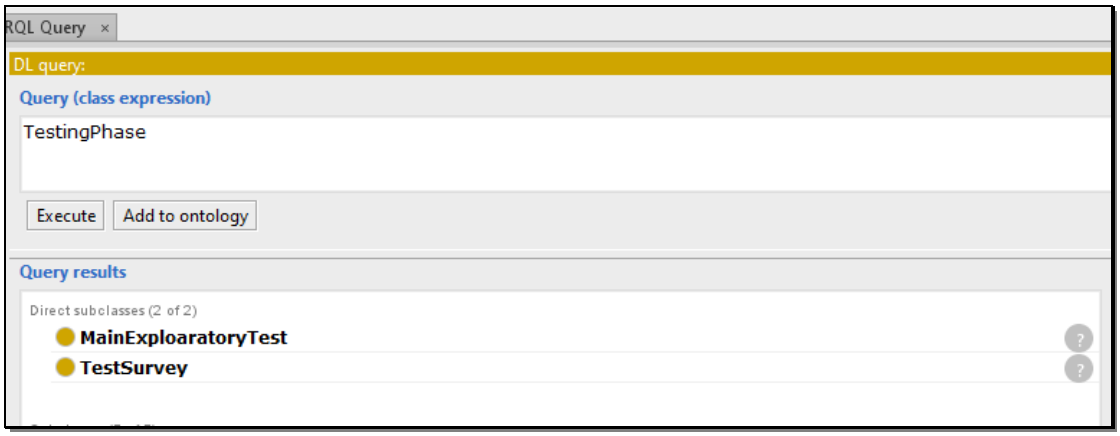

Figure 11: The result containing test phase
Remark: Result is accurate

4.2.2.4 Result for the Query: What are the types of test survey involved in the test process? DL QUERY: TestSurvey

The result for this query is shown in Figure 12.

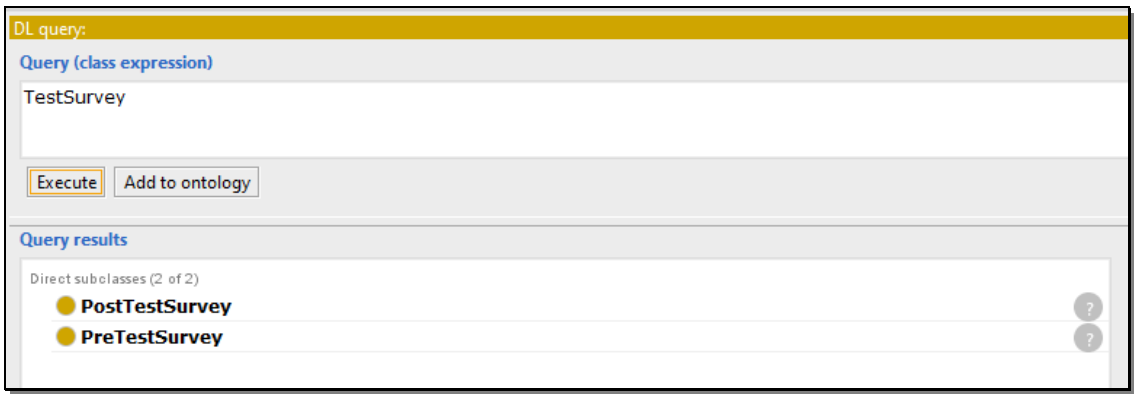

Figure 12: The result containing types of test survey
Remark: Result is accurate.

4.2.2.5 Result for the Query: What are the various artefacts involved in the testing process? DL QUERY: TestingArtefact

The result for this query is shown in Figure 13.

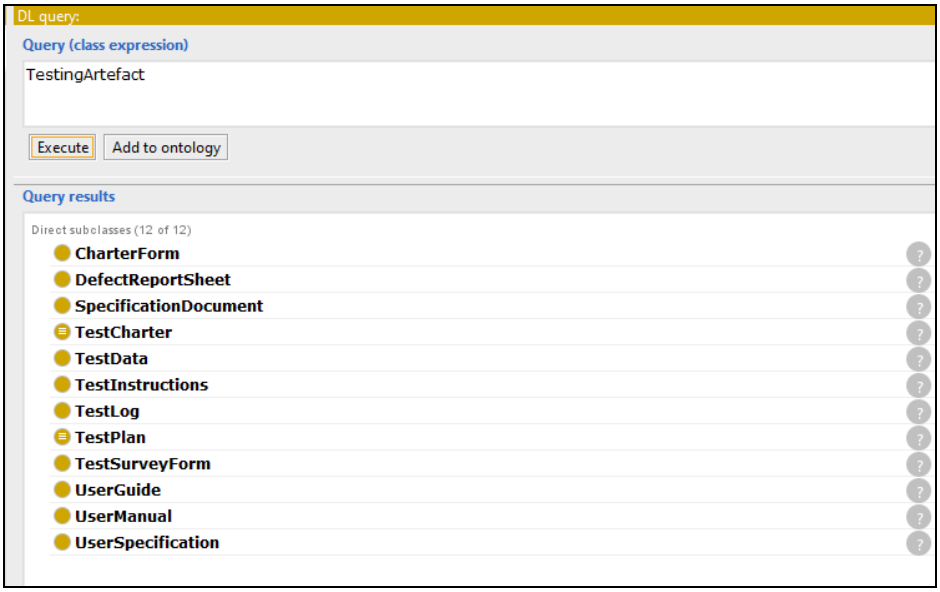

Figure 13: The result containing list of testing artefacts
Remark: Result is accurate.

4.2.2.6 Result for the Query: Who are the specific testers within the research and which bugs (represented with their respective bug codes) were detected by each of them?

SPARQL Query:

```
SELECT ?ExplorativeTester ?Bug
    WHERE {
        ?ExplorativeTester io:detectBug ?Bug. }
```

The result for this query is shown in Figure 14.

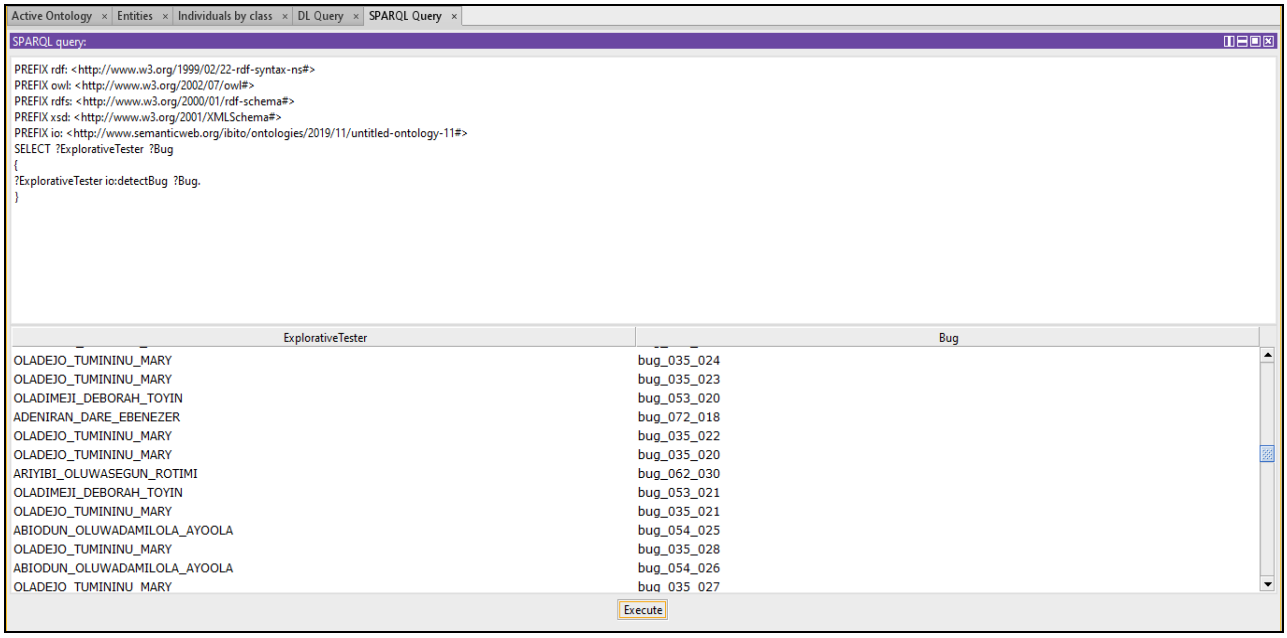

Figure 14: The result containing the list of testers and the respective bugs they detected
Remark: Result is accurate.

4.2.3.7 Result for the Query: Outline the descriptions of the bugs detected by the explorative testers for the bugs during the test procedure.

SPARQL Query:

```
SELECT ?ExplorativeTester ?Bug ?bug_description
    WHERE { ?ExplorativeTester io:detectBug ?Bug.
    ?Bug io:bug_description ?bug_description. }
```

The result for this query is shown in Figure 15.

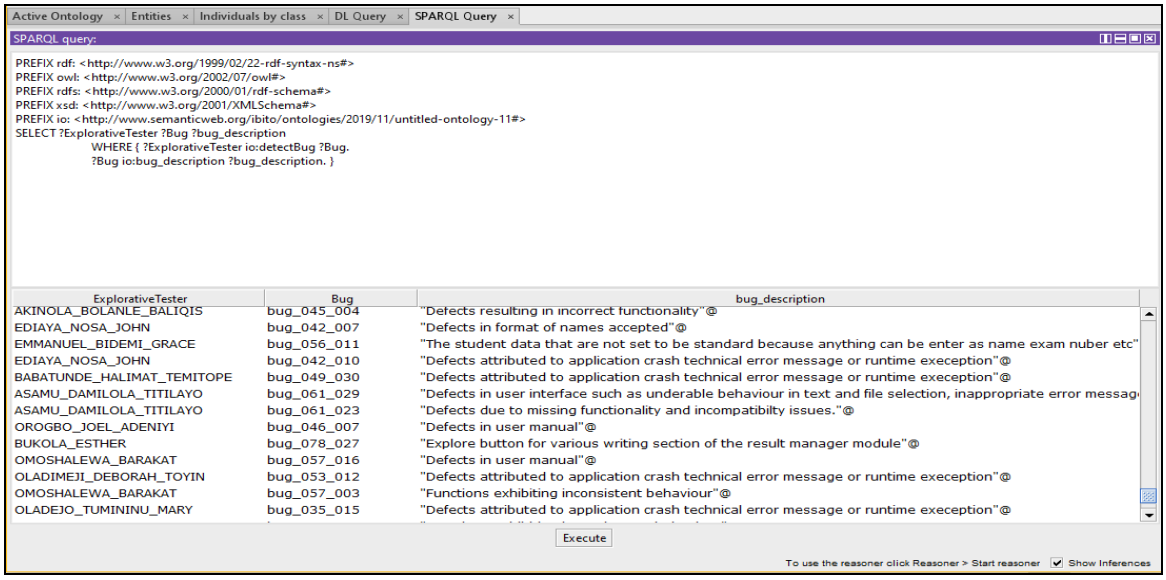

Figure 15: The result containing the descriptions of the bugs detected by the explorative testers for the bugs during the test procedure.
Remark: Result is accurate.

5.0 Conclusion

The activity ontology for Exploratory Software Testing (EST) developed in this study can be shared, applied, reused and improved throughout the organization. It can also be used as a learning guide on the exploratory software testing process as it gives a common understanding of the structure of EST for software testers. The domain knowledge can also be reused by the experts in the domain and by software development organizations. This work has created a framework for artificial intelligence agents in the area of exploratory software testing. Someone who wants to create intelligent agent in the area of exploratory software testing can make use of the ontology developed in this study as a knowledge base.

References

[1]. Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In: Formal Ontology in Conceptual Analysis and Knowledge Representation, Padova, Italy. *Proceedings... Padova*: ACM, 1993.

[2]. O'Leary, D. (1998). Using AI in knowledge management: knowledge bases and ontologies. *IEEE Intelligent Systems*, University of Southern California, 13(3):34–39, 1998b.

[3]. Rios, J. A. (2005). Ontologias: alternativa para a representação do conhecimento explícito organizacional. In: Encontro Nacional de Ciencia de Informacao, Salvador, Bahia. *Proceedings... Salvador: DBPL*, 2005.

[4]. Davenport, T. H.; Prusak, L. (2000). Working knowledge: how organizations manage what they know. 2. ed. Boston, USA: Harvard Business School Press, 2000.

[5]. Quadri S.M.K. and Farooq S. U. (2010). Software Testing – Goals, Principles & Limitations, *International Journal of Computer Applications*, 6(9), Sept. 2010.

[6]. James W. and Aybüke A. (2004). Knowledge Management in Software Engineering – Describing the Process, *Proceedings of the*

2004 Australian Software Engineering Conference (ASWEC'04).

[7]. Itkonen J, and Rautiainen K. (2005). Exploratory Testing: A Multiple Case Study, *Proceedings of ISESE*, 2005, pp. 84-93.

[8]. Itkonen J., Mantyla M. V., and Lassenius C. (2007). Defect detection efficiency: Test Case Based vs. Exploratory Testing, *proceedings of 1st International Symposium on Empirical Software Engineering and Measurement*, 2007, pp. 61-70.

[9]. Olszewska, J. I. (2020). AI-T: software testing ontology for AI-based systems. In Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (Vol. 2, pp. 291-298). SciTePress. https://doi.org/10.5220/0010147902910298

[10]. Hassnain, M., Jeong, S. R., Pasha, M. F. and Ghani, I. (2021). An Ontology Based Test Case Prioritization Approach in Regression Testing. Computers, Materials and Continua 67(1):1051-1068. DOI: 10.32604/cmc.2021.014686

[11]. Chimalakonda S. and Nori K.V. (2020). An ontology based modelling framework for design of educational technologies, *Smart Learning Environments* (2020) 7:28 https://doi.org/10.1186/s40561-020-00135-6

[12]. Ayorinde I. T. (2020). A Formalised Ontology of Musical Instruments. *International Journal of Computer Applications* 176(24):28-32, May 2020. DOI: 10.5120/ijca2020920235. Number 24 (ISBN: 973-93-80901-08-1)

[13]. Tebes, G., Peppino, D., Becker, P., Matturro, G., Solari, M., and Olsina, L. (2020). Analyzing and documenting the systematic review results of software testing ontologies, Information and Software Technology, 123:1–23, 2020.

[14]. Mårtensson T., Martini A., Ståhl D. and Bosch J. (2019). Excellence in Exploratory Testing: Success Factors in Large-Scale Industry Projects, *Proceedings* 20th International Conference, *PROFES* 2019, Barcelona, Spain, November 27–29, 2019, pp.299-314.

[15]. Ayorinde, I. T. and Oyedeji, O. A. (2019). An Ontology for Intra-Campus Transport System (ICTS) (A Case Study of the University of Ibadan Campus). *Journal of Digital Innovations and Contemp. Res. In Sc., Eng., & Tech.* 7(4): 65-78.