COURSE PROJECT

ONTOLOGIES AND SEMANTIC WEB

Testing Process and Activities with closer look at Unit Testing

## Made by: Mert Mustafov Kamberov 90b 371222025

## 08.12.2024

# Ontology purpose

This documentation provides a detailed overview of the Testing Process and Activities Ontology with focus on unit testing, designed to model and represent the various aspects of software testing. The ontology is constructed using OWL (Web Ontology Language) and aims to facilitate knowledge representation, standardization, and interoperability within the software testing domain.

The primary objectives of this ontology include:

* **Standardization**: Establishing a common vocabulary for testing processes and activities.
* **Interoperability**: Enabling seamless communication between different software tools and systems involved in testing.
* **Knowledge Representation**: Providing a formal structure for representing concepts related to software testing, allowing for automated reasoning and analysis.

Questions related to the ontology:

1. When did a testing process start and when did it end?

|  |
| --- |
| SELECT ?testingProcess ?startDate ?endDate  WHERE {  ?testingProcess a untitled:Testing\_Process .  ?testingProcess untitled:hasProcessStartDate ?startDate .  ?testingProcess untitled:hasProcessEndDate ?endDate .  FILTER(?testingProcess = <http://www.semanticweb.org/mert\_kamber/ontologies/2024/11/untitled-ontology-5#Car\_Testing\_Process>)  } |

1. When did a testing activity start and when did it end?

|  |
| --- |
| SELECT ?activity ?startDate ?endDate  WHERE {  ?activity a untitled:Testing\_Activity .  ?activity untitled:hasStartDate ?startDate .  ?activity untitled:hasEndDate ?endDate .  } |

1. What are the test cases?

|  |
| --- |
| SELECT ?testCase ?id ?status ?coverage  WHERE {  ?testCase rdf:type ont:Test\_Case .  ?testCase ont:hasID ?id .  ?testCase ont:hasStatus ?status .  ?testCase ont:hasCoverage ?coverage  } |

1. What are the different types of Unit Testing?

|  |
| --- |
| SELECT ?testingType  WHERE {  ?testingType rdfs:subClassOf ont:Unit\_Testing  } |

1. How many tests do each type of Unit Tests have?

|  |
| --- |
| SELECT ?unitTestType (COUNT(?instance) AS ?count)  WHERE {  ?unitTestType rdfs:subClassOf ont:Unit\_Testing .  OPTIONAL { ?instance rdf:type ?unitTestType }  }  GROUP BY ?unitTestType  ORDER BY DESC(?count)} |

1. Is there are test case with ID “FG-001”

|  |
| --- |
| ASK {  ?testCase rdf:type ont:Test\_Case .  ?testCase ont:hasID "FG-001" .  } |

# Glossary

 **Testing Process**: The overall methodology followed during testing, encompassing planning, execution, and evaluation.

 **Testing Activity**: Any action undertaken to evaluate software performance, reliability, or quality.

 **Testing Framework**: A set of guidelines or rules used to structure and organize testing efforts.

 **Bound**: A classification indicating a specific limit or boundary within testing parameters.

 **Integration Testing**: A type of testing that assesses the interaction between integrated components or systems.

 **Unit Testing**: The practice of testing individual components or modules of a software application.

 **Test Case**: A set of conditions or variables under which a tester assesses whether a system meets requirements.

 **Test Suite**: A collection of test cases intended to be executed together to validate a specific feature or functionality.

# Used ontologies

Souza, Erica & Falbo, Ricardo & Vijaykumar, Nandamudi. (2017). ROoST: Reference Ontology on Software Testing. Applied Ontology. 12. 1-32. 10.3233/AO-170177.

# Basic logic rules of the ontology

## Class Hierarchies

* **Subclass Relationships**: Classes are organized hierarchically, where subclasses inherit properties from their parent classes. For example:
  + Boundary\_Testing is a subclass of Unit\_Testing.
  + Composite\_Testing\_Activity is a subclass of Testing\_Activity.

## Disjoint Classes

* Certain classes are defined as disjoint, meaning they cannot share instances. For instance:
  + Negative\_Testing and Positive\_Testing are disjoint classes, indicating that a testing instance cannot be both.

## Object Properties

* **Domain and Range Restrictions**: Object properties have specific domains (the class from which instances can use the property) and ranges (the class of the values that the property can point to). For example:
  + The property executes has a domain of Testing\_Framework, meaning only instances of this class can execute tests.

## Cardinality Constraints

* **Cardinality Restrictions**: These rules specify how many times a property can be associated with an instance. For example:
  + The property hasTestingActivity must have at least one instance associated with it for a Testing\_Process.

## Inverse Properties

* Some object properties have defined inverse relationships, allowing for bidirectional reasoning. For example:
  + The property executes has an inverse property isExecutedBy, allowing one to infer relationships in both directions.

## Functional Properties

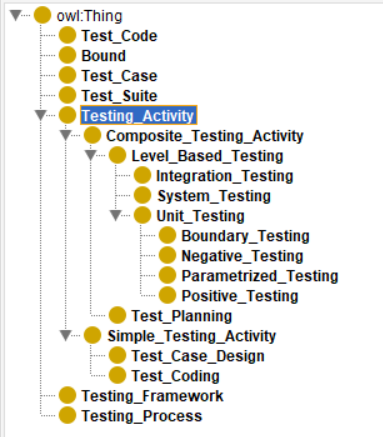
* Certain properties are defined as functional, meaning they can only have one value for a given instance. For example:
  + The property produces is functional, indicating that each instance can produce only one result.

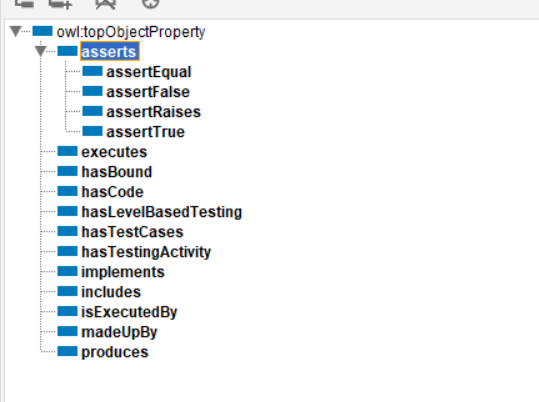
## Assertions

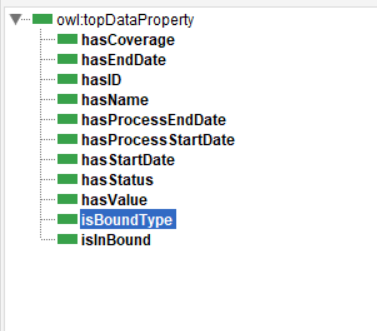
* **Class Assertions**: Instances are asserted to belong to specific classes, such as:
  + The individual Brakes\_Testing\_Activity is asserted to be an instance of Testing\_Activity.

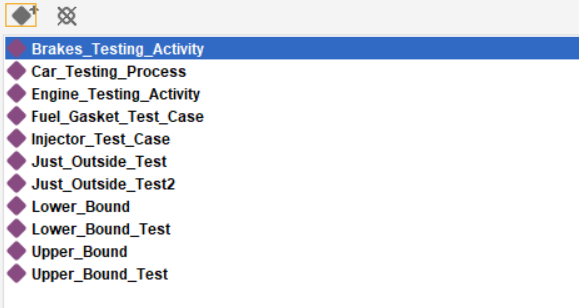
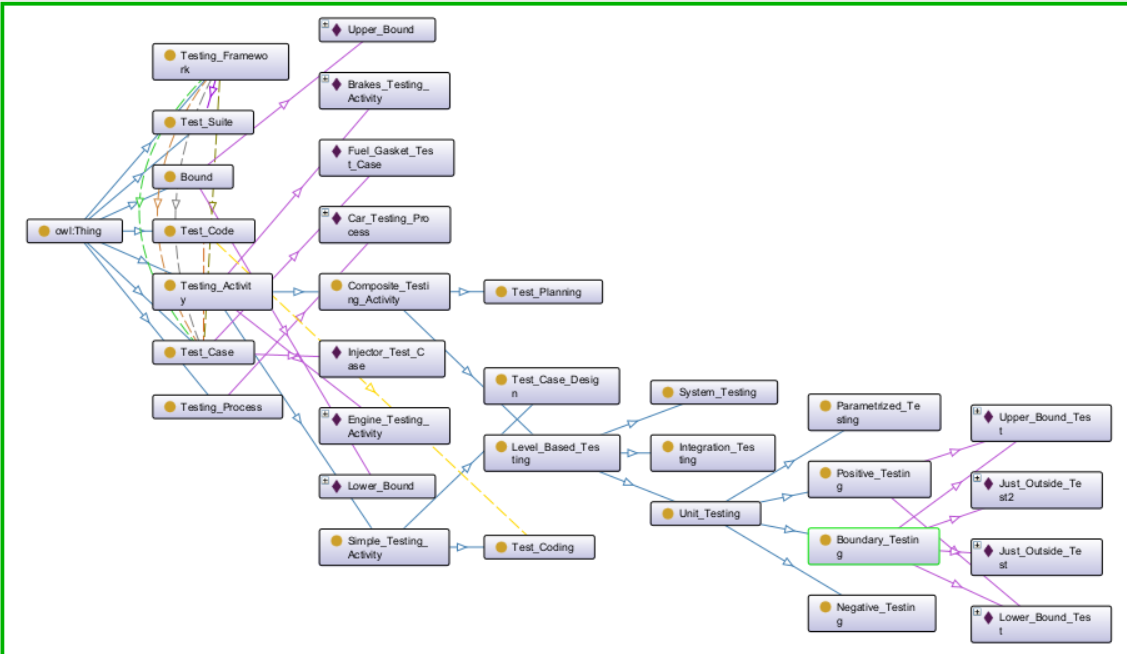
These logical rules create a structured framework that facilitates reasoning about testing methodologies and their interrelations within the ontology.

# Important screenshots from Protégé



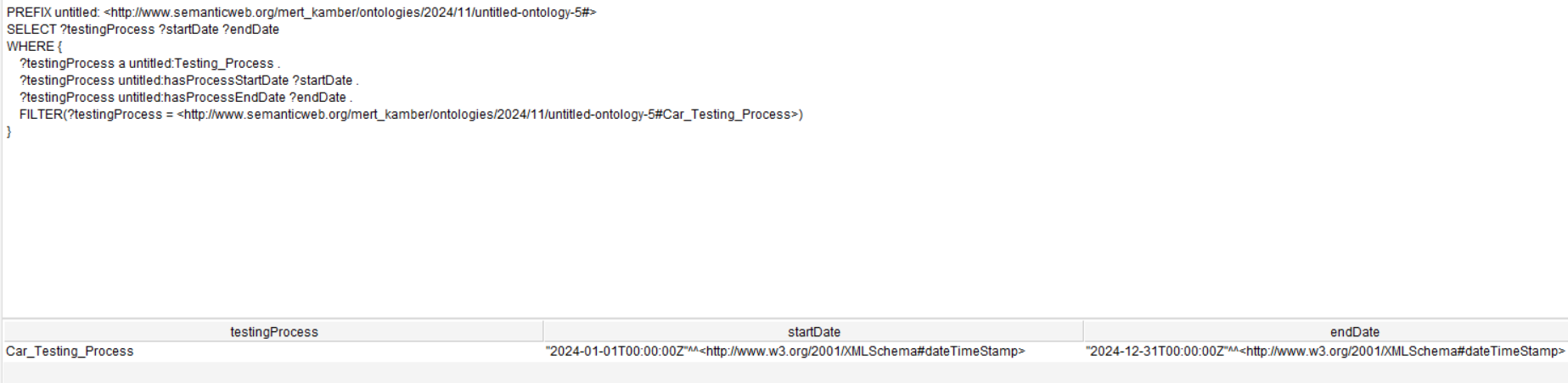




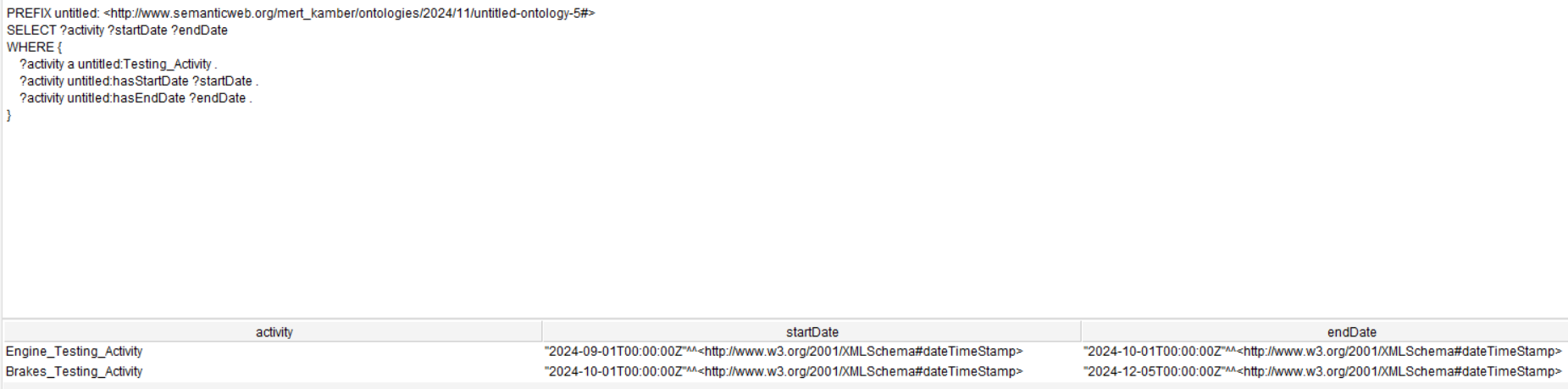
 

### SPARQL queries with results screenshots

1.



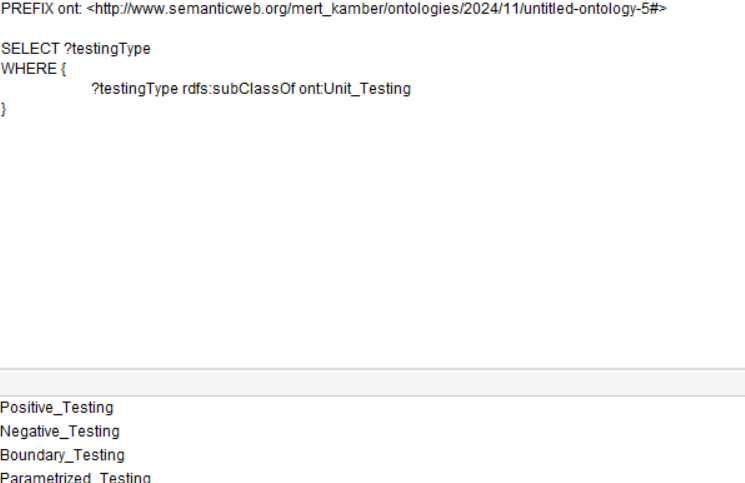
2.



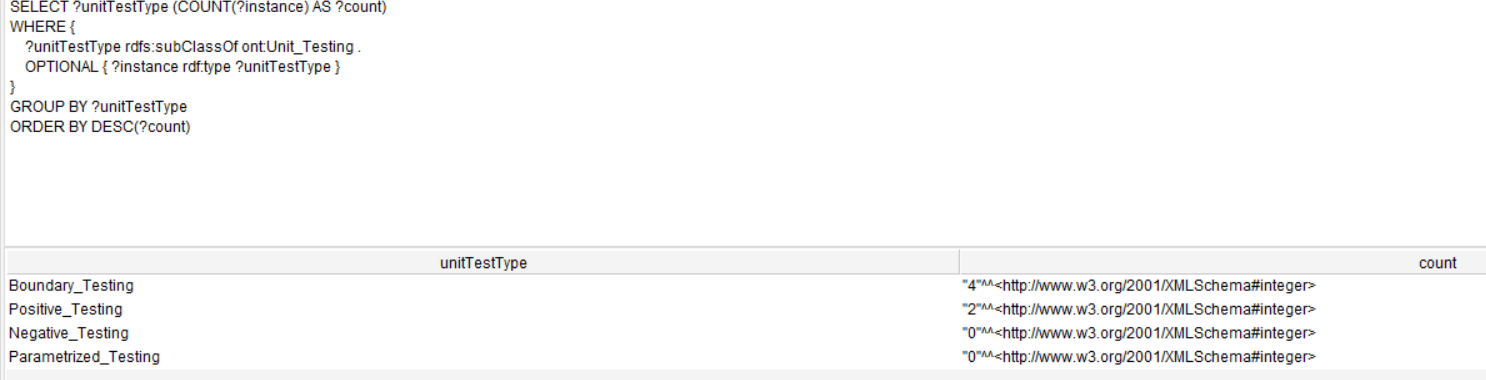
3.



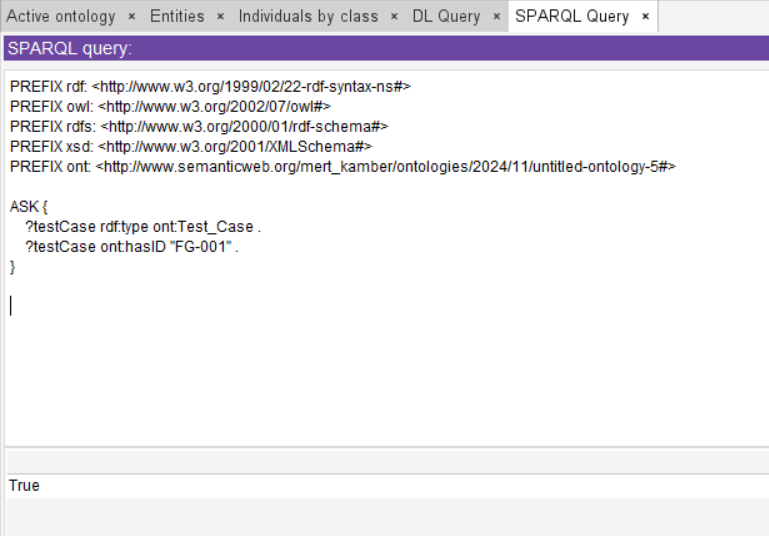
4.



5.



6.



# Resources

Souza, Erica & Falbo, Ricardo & Vijaykumar, Nandamudi. (2017). ROoST: Reference Ontology on Software Testing. Applied Ontology. 12. 1-32. 10.3233/AO-170177.

Software engineering by Ian Sommerville 10th edition