Enhancing a Software Testing Ontology

Pablo Becker, Guido Tebes, María Fernanda Papa, and Luis Olsina

GIDIS_Web, Facultad de Ingeniería, UNLPam, General Pico, LP, Argentina
[beckerp, guido_tebes, pmfer, olsinal]@ing.unlpam.edu.ar

Abstract. This paper discusses a software testing ontology called TestTDO (*software Testing Top-Domain Ontology*) v2.0, which is an enhanced version of its predecessor, TestTDO v1.3. This new version considers a broader scope and is aligned with terms, properties, and relationships of ProjectCO (*Project management Core Ontology*) v2.0, a recently developed ontology for project management. In addition, TestTDO v2.0 considers a study conducted on recognized testing glossaries, which identified terms with the highest syntactic and semantic frequencies. After performing a comparative analysis between these glossaries and the terms in TestTDO v1.3, most of the definitions and labels did not undergo significant changes. Finally, the practical impact of TestTDO v2.0 concepts and relationships on software testing process specifications is analyzed.

Keywords: software testing, ontology, ontological architecture, glossary.

Actualizando una Ontología de Pruebas de Software

Resumen. Este artículo presenta una ontología de pruebas de software llamada TestTDO v2.0, que es una versión actualizada de su predecesora, TestTDO v1.3. Esta nueva versión considera un alcance más amplio y está alineada con los términos, propiedades y relaciones de ProjectCO v2.0, una ontología recientemente desarrollada para la gestión de proyectos. Además, TestTDO v2.0 considera un estudio realizado en glosarios de pruebas reconocidos, que identificó los términos con las frecuencias sintácticas y semánticas más altas. Luego de realizar un análisis comparativo entre los términos de estos glosarios y los de TestTDO v1.3, la mayoría de las definiciones y etiquetas no sufrieron cambios significativos. Finalmente, se analiza el impacto práctico de los conceptos y relaciones de TestTDO v2.0 en las especificaciones de proceso de pruebas de software.

Palabras clave: prueba de software, ontología, arquitectura ontólogica, glosario.

1 Introduction

In the field of Software Engineering (SE), quality assurance plays a crucial role in ensuring that products and systems meet the required standards and fulfill user expectations. One of the key areas contributing to quality assurance is testing. This domain is complex, as it encompasses a wide variety of methods, processes, and strategies primarily aimed at identifying defects and improving the overall quality of software products. All of them involve many specific domain concepts, making it valuable to have a solid conceptual foundation that defines terms, properties, relationships, and axioms.

To achieve this aim, a software testing ontology can serve as a valuable resource. An ontology is the most structurally robust vocabulary, which unambiguously defines terms, properties, relationships, and axioms. By providing a structured representation of knowledge, an ontology facilitates better communication among stakeholders and enhances the consistency and effectiveness of testing strategies, functioning as a common reference vocabulary in the specifications of activities and methods.

This paper discusses TestTDO (*software Testing Top-Domain Ontology*) v2.0, an enhanced version of its predecessor, TestTDO v1.3 (Tebes et al., 2021a). The development of TestTDO v2.0 was driven by the need to harmonize and align its terminology with that of other established ontologies, particularly the recently developed ProjectCO (*Project management Core Ontology*) v2.0 (Olsina et al., 2024a). ProjectCO provides a comprehensive vocabulary for project management and considers terms from international glossaries such as PMBOK (*Project Management Body of Knowledge*) (PMI, 2021), APM (*Association for Project Management*) (APM, 2021), IAPM (*International Association of Project Managers*) (IAPM, 2021), and PRINCE2 (*PRojects IN Controlled Environments*) (PRINCE2, 2017). As a result, the new version of TestTDO considers an expanded scope.

Furthermore, TestTDO v2.0 incorporates insights from a study performed by Olsina et al. (2024b) on testing glossaries, which included well-known standards such as ISO/IEC 29119 (ISO, 2022), ISTQB (*International Software Testing Qualifications Board*) (ISTQB, 2022), and TMMi (*Test Maturity Model Integration*) (TMMi Foundation, 2018). This study aimed to identify the terms with the highest syntactical and semantic frequency within these glossaries, thereby ensuring that the ontology reflects widely accepted concepts and definitions in the field. As a result, after performing a comparative analysis between these glossaries and the TestTDO v1.3 terms, most of the definitions and labels did not undergo significant changes for TestTDO v2.0. This is because TestTDO v1.3 was already developed considering previous versions of the ISTQB (ISTQB, 2021) and ISO/IEC 29119 (ISO, 2013) glossaries.

The original motivation for this research was to represent a set of interrelated concept systems structured in ontologies that integrate terms, relationships, constraints, and axioms at different levels of generality and specificity. These conceptual systems encompass entities (things) and assertions, processes, goals, situations, events, project management, functional and non-functional requirements, testing, measurement, and evaluation, as well as metrics and indicators. These systems of concepts should be placed within the context of an ontological architecture for representing, reusing, and extending knowledge, which should be useful not only for SE but for any science. As a practical impact, the concepts incorporated into the specifications of strategies, processes, methods, etc., must emerge from these common reference vocabularies.

The main contribution of this paper is the presentation of the concepts, relationships, and axioms of the new TestTDO, which align with terms and relationships of the ontologies at the core and foundational levels. Also, to illustrate the applicability of the TestTDO conceptualization, the reuse of its terms and relationships in the specifications of activities, methods, and work products included in a testing strategy is showcased.

In short, TestTDO v2.0 represents a significant advancement in the field of software testing ontology. By broadening its scope, harmonizing its terms and relationships with established frameworks, and incorporating insights from industry standards, it lays a solid foundation for future research and practice in the field of quality assurance.

The remaining sections are organized as follows: Section 2 provides an overview of the architecture in which TestTDO and other ontologies are placed. Section 3 analyzes the goal, scope, and requirements of the updated testing ontology and its conceptualization. Section 4 illustrates the applicability of TestTDO conceptualization by reusing its terms in specifications of activities and methods included in testing strategies. Section 5 discusses related work and Section 6 outlines the conclusions and future work.

2 TestTDO in a Multi-level Ontological Architecture

As mentioned, the original motivation for this research was to represent a set of interrelated concept systems structured in ontologies that integrate terms, relationships, constraints, and axioms at different levels of generality and specificity. To achieve this, we developed an ontological architecture known as FCD-OntoArch (*Foundational, Core, and Domain Ontological Architecture for sciences*) (Olsina, 2023), which encompasses

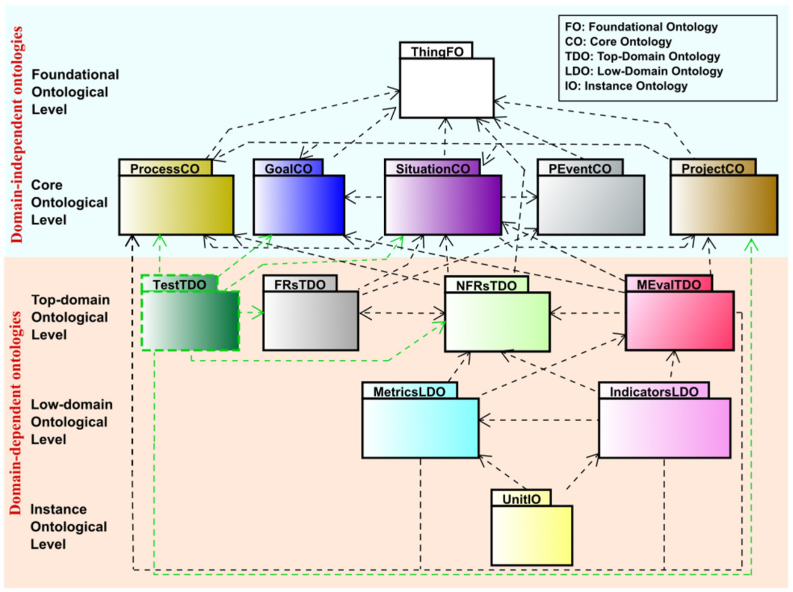

Fig. 1. TestTDO in the context of FCD-OntoArch. Note: FRs stands for Functional Requirements, NFRs for Non-Functional Requirements, and MEval for Measurement and Evaluation.

foundational, core, domain, and instance levels. In turn, as depicted in Fig. 1, the domain ontological level is further divided into Top-domain and Low-domain.

To identify the ontological components within FCD-OntoArch, ontologies associated with a specific level of generality or specificity are designated based on their position within the architecture. Therefore, an ontology at the foundational level is referred to as Foundational Ontology (FO for short); ontologies at the core level are termed Core Ontologies (CO); those at the top-domain level are called Top-Domain Ontologies (TDO); and ontologies at the low-domain level are designated as Low-Domain Ontologies (LDO). There is also a layer for instances (IO).

Fig. 1 illustrates, on the left side, the five levels of generality/specificity, as well as the domain-independent and domain-dependent layers. The components on the right side correspond to the ontologies already built. Furthermore, Fig. 1 shows that ontologies at the same level can be related to each other, except for the foundational level, where only one ontology exists. Additionally, the terms, properties, and relationships of ontologies at lower levels can be semantically enriched by those from higher levels.

At the foundational level, ThingFO (Olsina, 2023) is the only necessary ontology. It represents the world through particular Things, universal Things, and Assertions. These unique concepts (along with their related terms and relationships) are specialized in the lower components. In other words, terms included in lower-level ontologies must be semantically enriched with those represented in higher-level ontologies. At the lower levels of the FCD-OntoArch, the following ontologies have been located: i) at the core level: ProcessCO (Becker et al., 2022), GoalCO, SituationCO, PEventCO (Particular Event), and ProjectCO; ii) at the top-domain level: TestTDO, FRsTDO, NFRsTDO (Olsina et al., 2025), and MEvalTDO; and iii) at the low-domain level: MetricsLDO and IndicatorsLDO.

D'Aquin et al. (2011) proposed a set of characteristics or principles that should be considered when designing ‘beautiful ontologies’. The authors argue that the concepts and relationships of the designed domain ontology must be grounded on a foundational ontology. Moreover, they state that the built ontology should be modular, extensible, and capable of reusing already developed related ontologies, aligning with the principle of being integrated into a modular architecture, framework, or network.

In line with the first characteristic, TestTDO is founded upon the foundational ontology ThingFO. Regarding the second characteristic, TestTDO was developed as an ontological component at the top-domain level that has domain-dependent concepts and relationships for the field of testing activities and methods. It extends and creates more specialized terms and relationships derived from core ontologies such as ProcessCO, GoalCO, SituationCO, and ProjectCO, while also reusing elements from FRsTDO (Functional Requirements) and NFRsTDO (Non-Functional Requirements).

Fig. 1 also shows with green dashed lines the links from TestTDO to other ontological components. The dependency relationships of TestTDO with other higher components indicate that some of its terms are semantically enriched with terms from ProcessCO, GoalCO, SituationCO, and ProjectCO. Ultimately, the presented multilayer ontological architecture fosters a clear separation of concerns by delineating ontological levels and assigning conceptual components in the right place. This also favors the modularity, extensibility, and reuse of ontological elements across all levels.

3 Analysis of TestTDO Scope, Conceptualization, and Validity

Next, in subsection 3.1, we discuss the goal of TestTDO, as well as its scope, defining a set of competency questions (CQs). Then, in subsection 3.2, we depict the resulting TestTDO conceptualization and analyze a subset of concepts and relationships that will be used in Section 4. Finally, in subsection 3.3, we show an extract of the verification matrix of terms, relationships, properties, and axioms of TestTDO considering CQs.

3.1 Definition of Competency Questions

One of the aims of this work is the discussion of the latest updated and harmonized version 2024 (v2.0) of TestTDO, taking into account the core ontologies already developed. It should be noted that we formalized a previous version of it in 2019 (Tebes et al., 2021a), but it was not harmonized with ProjectCO v2.0, which is a recently developed project management ontology (Olsina et al., 2024a). To harmonize TestTDO with the latest ProjectCO, it was necessary to expand the scope of TestTDO. To achieve this, new competency questions (CQ) were defined, and some existing ones were redefined. As a result, TestTDO v2.0 includes 36 CQs, compared to 25 in version 1.3.

In this stage, we considered the work done to define the scope of ProjectCO, which established semantic categories to which concepts can belong (Olsina et al., 2024a). These terminological-semantic categories were also useful in identifying conceptual blocks or patterns and specifying the scope of the testing ontology at the top-domain level in the context of the FCD-OntoArch framework.

The new CQs are categorized and distributed as follows: 4 CQs belong to the Test Project Entity category, 3 to the Test Goal, 5 to the Test Situation, 4 to the Test Requirements, 5 to the Testing Work Process, 7 to the Testing Work Product, 3 to the Testing Method, 3 to the Testing Agent, and 2 CQs are included in the Testing Strategy category. As an example, the following CQs belong to the Test Project Entity category:

CQ1. *For a test project that operationalizes a test goal, does the test project have a process with an associated testing strategy that helps achieve the purpose of the test goal?*

CQ2. *For a given test project, is there a work process to manage it?*

CQ3. *For a given test project, are there management and engineering work processes related to it?*

CQ4. *For a given test project, what are the test entities necessary as inputs in a testing process?*

Note that in TestTDO v1.3, there was only one CQ for this category. The increase from 1 CQ to 4 in this category is a result of TestTDO v2.0's expanded scope and its alignment with ProjectCO. The reader can find all the CQs in Tebes et al. (2024).

As a final remark, Monfardini et al. (2023) reported on a survey that showed that CQs have proven useful in helping to define the scope of a given ontology and verify its conceptualization. We fully subscribe to these insights of usefulness, both in helping to define the scope and in verifying and validating the conceptualization.

3.2 TestTDO's Conceptualization

In addition to the principles proposed by D'Aquin et al. (2011) mentioned above, other quality criteria that should be considered for the designed ontology are namely formal simplicity and transparency promoting also the use of graphical representations for the conceptualization; coverage completeness but, at the same time, conciseness and self-intuitiveness of the elements included; and balanced representation of both taxonomic (generic and partitive hierarchical) and non-taxonomic (associative) relationships.

Considering these criteria, the TestTDO conceptualization meets the criterion of

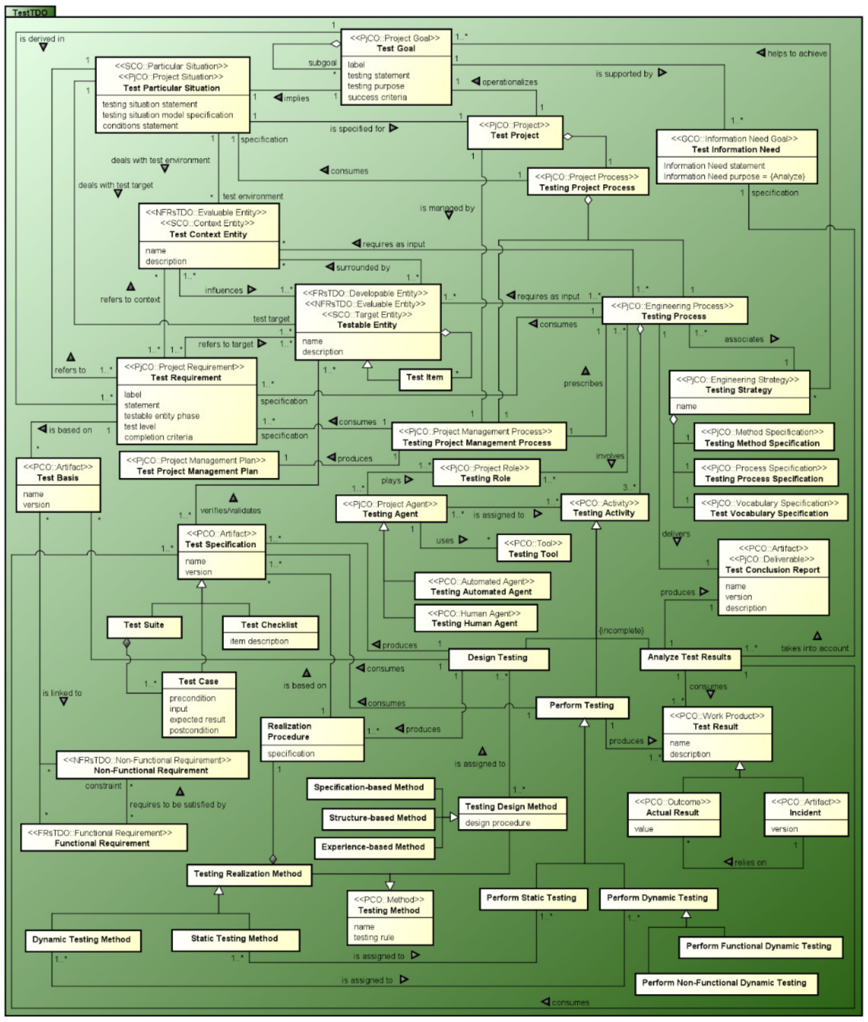

Fig. 2. Software Testing ontology. Note: PCO stands for Process Core Ontology, SCO for Situation Core Ontology, PjCO for Project Core Ontology, GCO for Goal Core Ontology, FRs for Functional Requirements, and NFRs for Non-Functional Requirements.

having a balanced representation of types of relationships, i.e., taxonomic and non-taxonomic ones. As a result, the TestTDO conceptualization includes 49 main terms –each semantically enriched with terms reused mainly from core ontologies–, 39 properties, 32 taxonomic and 46 defined non-taxonomic relationships, as well as UML (*Unified Modeling Language*) constraints and 17 axioms. Fig. 2 depicts the resulting conceptualization of TestTDO v2.0 specified in UML.

It is important to note that the ontologies in this work utilize stereotypes as a mechanism for reusing, enriching, and aligning terms semantically. Becker et al. (2022) argue that stereotypes can simplify model complexity, enhancing both understandability and communicability. For instance, in Fig. 2, we observe that the term 'Test Specification' has the semantics of <<Artifact>>, which is a term defined in ProcessCO. In addition, 'Artifact' inherits the semantics from the term <<Thing>> in ThingFO.

Below we describe the conceptualization of TestTDO using the following text convention: ontology top-domain terms begin with capital letters, properties are in italics, and relationships are underlined. Please note that some terms used in the sequel are defined in Table 1. For definitions of the remaining terms, refer to Tebes et al. (2024).

As can be seen in the ontology representation of Fig. 2, a Test Project (see definition in Table 1) <u>operationalizes</u> a Test Goal, which encompasses four properties: *label*, *testing statement*, *testing purpose*, and *success criteria*. In turn, a Test Goal <u>is derived in</u> Test Requirements and <u>implies</u> a Test Particular Situation, <u>dealing with</u> Testable and Test Context Entities (see definitions in Table 1). Furthermore, a Test Requirement

Table 1. Definition of some terms of the TestTDO ontology

TestDO Term	Definition
Test Context Entity	It is an Evaluable and Context Entity in a given Test Particular Situation that represents a concrete entity that influences the Testable Entity.
Test Goal (synonym **Test Objective**)	It is a Project Goal (Olsina et al., 2024a) for testing that the organization intends to achieve through a specific Test Project.
Test Particular Situation	It is a Project Situation (Olsina et al., 2024a) for a particular goal-oriented test endeavor.
Test Project	It is a Project (Olsina et al., 2024a) representing a goal-oriented, temporary endeavor for testing with definite start and end dates, which considers a managed set of interrelated Testing Activities, tasks, and test project resources aimed at producing, modifying, and delivering unique work products (e.g., Test Specifications, Test Results, etc.) to meet the needs of certain stakeholders.
Testable Entity (synonym **Test Object**)	It is a Target Entity, Developable or Evaluable, in a given Test Particular Situation that represents a concrete entity to be tested. <u>Note</u>: Depending on the Test Particular Situation implied by the Test Goal, Testable Entity has the semantics of Developable Entity (from FRsTDO) or Evaluable Entity (from NFRsTDO).
Testing Project Management Process	It is a Project Management Process (Olsina et al., 2024a) that includes a set of managerial activities intended to achieve the Test Goal operationalized by a Test Project.
Testing Process (synonym **Testing**)	It is an Engineering Process (Olsina et al., 2024a) that includes Testing Activities conducted to facilitate the discovery of defects and/or the assessment of characteristics and attributes of Testable Entities in a given Test Particular Situation
Testing Strategy	It is an Engineering Strategy (Olsina et al., 2024a) that helps to achieve the testing purpose of a Test Goal.

refers to these entities and is based on the Test Basis, which includes requirements documentation, source code, and system design, among other artifacts. A Test Basis is an <<Artifact>>, which is linked to Functional and Non-Functional Requirements (terms taken from FRsTDO and NFRsTDO, respectively).

Continuing with the description, a Test Project mandatorily has a Testing Project Process, which in turn has two work processes that are labeled Testing Project Management Process and Testing Process (see definitions in Table 1). Note that in TestTDO v1.3, these terms and relationships were different. For example, Testing Project Management Process was labeled as Testing Management, and Testing Process was simply called Testing. Additionally, the stereotypes for these terms had to be updated according to the latest version of ProjectCO. In the case of Testing Project Management Process, it is now semantically enriched by the stereotype <<Project Management Process>>, while Testing Process is enriched by <<Engineering Process>>; both stereotypes are defined as terms in ProjectCO.

On the one hand, a Testing Project Management Process consumes Test Requirements and produces a Test Project Management Plan (called Test Plan in v1.3), which is defined as "*a Project Management Plan describing how the Test Project will be executed, monitored, controlled, and closed*". On the other hand, a Testing Process requires as input the Testable and Test Context Entities, and consumes the specification of Test Requirements. As output, it delivers the Test Conclusion Report. Note that many of these latter terms and relationships have been added or updated from the previous v1.3., to satisfy, for example, CQ3 and CQ4.

Considering engineering activities related to a Testing Process, there are at least three types of Testing Activities represented in Fig. 2, namely: Design Testing, Perform Testing, and Analyze Test Results. It is important to say that these three activities are the minimum and necessary set for any Testing Process. Other activities and sub-activities can be considered (as we will show in Section 4), but we only made the generic activities explicit since the ontology is at the top-domain level. It's worth mentioning that these activity terms were previously referred to as Testing Design, Testing Realization, and Testing Analysis, respectively. This change was made to align them with the MEvalTDO activity nomenclature.

Design Testing aims to design (and hence produce) a set of Test Specifications as well as Realization Procedures. Test Specification has the semantics of <<Artifact>> and there are three types, namely: Test Checklist, Test Case, and Test Suite. Test Cases contain the necessary information (e.g. *preconditions*, *inputs*, *expected results*, and *postconditions*) to perform mainly dynamic testing. Note that Test Cases with common constraints on their realization can be grouped into Test Suites. Finally, Test Checklists contain a list of *item descriptions* to be checked to perform primarily static testing.

Perform Testing consumes one or more Test Specifications to produce one or more Test Results. This term has the semantics of <<Work Product>> and there are two types of it, namely: Actual Result and Incident. The former is an <<Outcome>> that represents a numerical or categorical *value* –expected or unexpected. Instead, an Incident is an <<Artifact>> or document, which reports deviations (e.g., between the Test Case's *expected result* and the Actual Result), anomalies (e.g., an error or a failure) or other arisen issues during the testing realization (i.e. the Perform Testing activity). When an

Incident occurs since there is a mismatch between the Test Case's *expected result* and the Actual Result, then this Incident relies on the corresponding Actual Result.

Additionally, Perform Testing is aimed at enacting a static or dynamic testing activity. So, Perform Static Testing aims at checking a Testable Entity against one or more Test Specifications without the execution of its software code, if any. Instead, Perform Dynamic Testing aims at verifying/validating a Testable Entity against one or more Test Specifications with the execution of its software code. Finally, Analyze Test Results takes into account the specific Test Information Need to produce a Test Conclusion Report. The Test Conclusion Report is an <<Artifact>> that documents the analysis of all Test Results. For example, this document could contain details about the degree to which the Test Goals were achieved by analyzing Test Information Need goals and the coverage level achieved by the executed Test Cases, among other analyses.

TestTDO also has terms related to Testing Method such as Static/Dynamic Testing Method, Testing Realization Method, and Testing Design Method. As shown in Fig. 2, a Static/Dynamic Testing Method is assigned to Perform Static/Dynamic Testing. Examples quoted in the literature on Static Testing Methods are walkthrough, technical review, inspection, among others.

TestTDO defines the term Testing Method as a specific way to perform the specified steps for a task included in a Testing Activity. The specific and particular way of a Testing Method –i.e. how the specified steps in a testing task should be made– is represented by a *procedure* (e.g., *design procedure* or Realization Procedure) and *rules*. At this point, it is important to emphasize that TestTDO, from its inception, has adopted the principle from ProcessCO of maintaining a clear separation of concerns between 'the what' (activities and tasks) and 'the how' (methods and procedures).

Like the Testing Realization Method, the Testing Design Method is another kind of Testing Method, but it is assigned to the Design Testing activity. There are three kinds of Testing Design Methods, namely: Specification-based Method (also known as black-box), Structure-based Method (also known as white-box), and Experience-based Method. These types of Testing Realization Methods can, in turn, be broken down into more specific methods, such as Classification Tree Method, Scenario Testing, Statement Testing, Decision Testing, Modified Condition/Decision Coverage Testing, Error Guessing, and Exploratory Testing, among many others. However, these terms are not included in TestTDO, as it operates at the top-domain ontological level.

Finally, it is important to highlight the term Testing Strategy (see its definition in Table 1), which is related to the term Testing Process, taking into account that a Testing Process associates a Testing Strategy. Any Testing Strategy can be seen as a resource of a Test Project that includes principles and integrated properties such as a Testing Process Specification ("*It is a Process Specification that represents a model which relates a set of Testing Process elements such as Testing Activities, tasks, inputs and outputs, pre- and post-conditions, artifacts, and roles, amongst others*"), a Testing Method Specification ("*It is a Method Specification that represents the specific and particular way to perform the specified steps in the description of a Testing Activity, particularly, in its sub-activities and tasks, which is part of a Testing Process*"), and a Testing Vocabulary Specification ("*It is a Vocabulary Specification that represents a concept system for the Test domain*"), to help achieve the *testing purpose* of a Test Goal.

3.3 TestTDO's Verification

In Tebes et al. (2021a), five verification and validation methods were carried out for the conceptualization and implementation of TestTDO. Due to space constraints, this paper presents only an excerpt of the static verification of the TestTDO conceptualization against the CQs, as suggested in Monfardini et al. (2023). The primary aim is to ensure that all CQs are addressed by the terms, properties, relationships, and axioms.

Table 2 shows a fragment of the verification matrix for TestTDO. The complete matrix can be seen in Tebes et al. (2024). To design the matrix, we used the 36 CQs and the ontology elements (terms, relationships, properties, and constraints/axioms) as the test basis. The matrix features one row per each CQ, in which we record the elements of the conceptualization that contribute to answering it. For example, in Table 2, the second column records three non-taxonomic relationships, two taxonomic relationships, and one axiom for CQ1. Note that axioms are specified in first-order logic.

After several iterations in the TestTDO conceptualization process, we ensured that each CQ was successfully addressed.

Table 2. Test Checklist (fragment) for inspecting how CQs are covered by TestTDO terms, properties, relationships, and axioms.

Competency Question (CQx)	What are the TestTDO Terms, relationships, *properties*, Axioms, and their definitions/specifications that answer the corresponding CQ?	Actual Result (pass/fail)
CQ1. For a test project that operationalizes a test goal, does the test project have a process with an associated testing strategy that helps achieve the purpose of the test goal?	A Test Project operationalizes a Test Goal	pass
	A Testing Project Process is part of a Test Project	
	A Testing Process is part of a Testing Project Process	
	A Testing Process associates a Testing Strategy	
	A Testing Strategy helps to achieve a Test Goal	
	Related axiom: A8 (*)	
CQ2. For a given test project, is there a work process to manage it?	A Testing Project Process is part of a Test Project	pass
	A Testing Project Management Process is part of a Testing Project Process	
	A Test Project is managed by a Testing Project Management Process	

(*) *A8 description/specification: All Test Project operationalizes a Test Goal and has a Testing Process that associates a Testing Strategy iff this Testing Strategy helps to achieve the operationalized Test Goal.*

$$\forall tp, \forall tg, \exists t, \exists tpp, \exists ts: TestProject(tp) \land TestGoal(tg) \land operationalizes(tp, tg) \land TestingProjectProcess(tpp) \land TestingProcess(t) \land partOf(tpp, tp) \land partOf(t, tpp) \land TestingStrategy(ts) \land associates(t, ts) \leftrightarrow helpsToAchieve(ts, tg)$$

4 TestTDO's Usefulness

As commented above, any Testing Strategy includes a Testing Process Specification, a Testing Method Specification, and a Testing Vocabulary Specification. Fig. 3 depicts a Testing Process Specification for dynamic testing using the UML activity diagram and the SPEM (*Software & Systems Process Engineering Metamodel*) notation, which represents functional and behavioral process perspectives (views) that include Testing Activities, their sequences, inputs and outputs, and so on.

Looking at Fig. 3, we can see –highlighted in blue– the three main Testing Activities: Design Testing (A1), Perform Dynamic Testing (A2), and Analyze Test Results (A3). These activity names are taken from the TestTDO conceptualization (Fig. 2), which represents the Testing Vocabulary Specification. Note that sub-activity names are not included in TestTDO since the ontology is in the top-domain level and attempts to meet the criteria of completeness of coverage but, at the same time, conciseness. Despite this, the sub-activity names consider TestTDO terms. For example, the sub-activity Define Test Requirements (A1.1) and Specify Test Particular Situation (A1.2) use the terms Test Requirement and Test Particular Situation, respectively.

Additionally, consumed or produced artifacts are also named using TestTDO terms. For instance, in Fig. 3, Analyze Test Results consumes Test Results and produces a Test Conclusion Report. Note also that in TestTDO these corresponding terms are associated using the consumes and produces relationships.

Considering the concept of Testing Method Specification, terms in TestTDO are also employed to define templates that represent methods. For instance, Tebes et al. (2021b) document a Test Case template used in the activity Design Test Cases (A1.3). This template contains the essential information required to produce Test Cases. Specifically, this information is derived from the properties associated with the term Test Case, including *preconditions*, *inputs*, *expected results*, and *postconditions*.

In summary, TestTDO serves as a Testing Vocabulary Specification and is useful as a common reference vocabulary for Testing Process Specification and Testing Method Specification. Consequently, the specifications of Testing Strategies ensure syntactical and terminological uniformity and semantic consistency, thus facilitating the understanding and communication of their processes, methods, and tools.

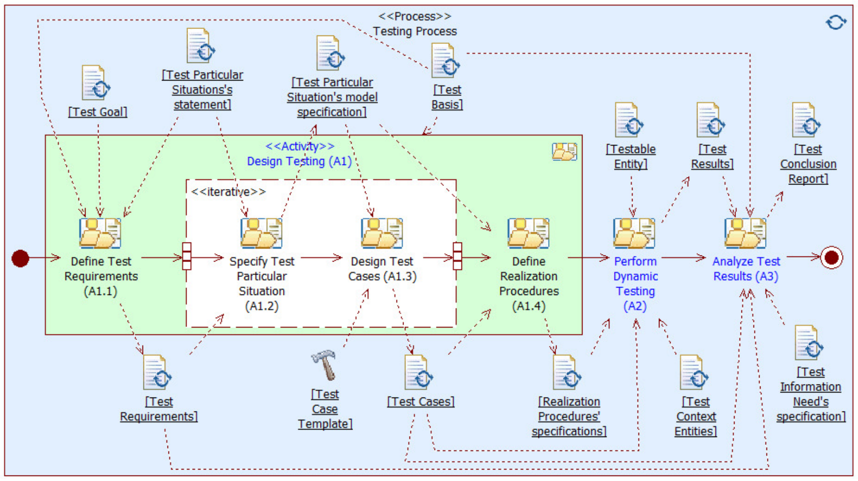
<<Process>>
Testing Process
[Test Goal]
[Test Particular Situations's statement]
[Test Particular Situation's model specification]
[Test Basis]
[Testable Entity]
[Test Results]
[Test Conclusion Report]
<<Activity>>
Design Testing (A1)
<<iterative>>
Define Test Requirements (A1.1)
Specify Test Particular Situation (A1.2)
Design Test Cases (A1.3)
Define Realization Procedures (A1.4)
Perform Dynamic Testing (A2)
Analyze Test Results (A3)
[Test Requirements]
[Test Case Template]
[Test Cases]
[Realization Procedures' specifications]
[Test Context Entities]
[Test Information Need's specification]

Fig. 3. Functional and behavioral view specification of the Testing Process for dynamic testing.

5. Related Work and Discussion

In this section, we discuss related work that addresses the three principles or features outlined by D'Aquin et al. (2011) for ontology construction. These are: i) the use of international terminological standards, such as glossaries; ii) the incorporation of elements from a foundational ontology to specialize and align newly developed elements; and iii) the requirement that the designed ontology be modular, extensible, and integrated into a multi-level ontological architecture or network.

As a result of a systematic literature review carried out by Tebes et al. (2020), we selected, analyzed, and evaluated 12 ontologies related to the software testing domain. Most of these ontologies did not meet all three principles simultaneously. Furthermore, none of the 12 ontologies analyzed by Tebes et al. (2020) were created with the same goal as TestTDO. The main objective of TestTDO is to provide a common vocabulary for specifications of methods and processes within a family of testing strategies. In contrast, most of the selected ontologies were developed for very specific purposes, making their scope narrower than that of TestTDO, which is broader and located at the top domain level in the architecture.

The only selected ontology that met the three principles is the Reference Ontology on Software Testing (ROoST) (De Souza et al., 2017). This ontology was developed to create a common conceptualization of the software testing domain, specifically focusing on the testing process. However, its scope is limited to dynamic and functional testing, leaving aside static and non-functional testing. ROoST is part of a network whose root is the UFO foundational ontology (Guizzardi, 2005), which consists of a set of ontologies: UFO-A (endurants), UFO-B (perdurants or events), and UFO-C (social entities, built on top of UFO-A and B). Instead, TestTDO is part of an ontological architecture that includes just one foundational ontology, i.e., ThingFO, which has a small set of terms that makes it easy to reuse and specialize in lower-level ontologies.

Also, we can mention the ontology documented by Asman (2015). This ontology aims to represent general software testing knowledge and is the only selected ontology that explicitly claims to be a top-domain ontology. However, its scope is restricted to terms related to the formal review process and does not include general terms for the overall testing process. Additionally, it lacks top-domain terms such as Test Basis, Test Result, and Actual Result, as well as terms related to test projects, goals, and situations.

Regarding TestTDO v1.3, it was developed based on the guidelines outlined by D'Aquin et al. (2011). Following the development of v1.3, in 2023, Olsina et al. (2024b) reviewed the most recent ISTQB and ISO 29119 glossaries to determine if some terms or definitions needed to be added or updated in the new TestTDO. For example, we found that among the terms ending in "testing" that have full syntactic and semantic similarity (i.e., the term appears in all three glossaries and their definitions fall into the same semantic category), only three could impact TestTDO terms/definitions, namely: Testing, Static Testing, and Dynamic Testing. However, there were no changes in ISTQB glossary versions, and the definitions in the new ISO 29119 were only slightly updated, which did not affect the corresponding terms of TestTDO.

Considering some differences between TestTDO v1.3 and v2.0, we have noted in Section 3 that the new version has a broader scope in terms of CQs. Also, in v1.3 we used Testing as the main term and Testing Process as a synonym, but in v2.0, we decided to use Testing Process as the main term and Testing as a synonym. Its definition was also revised; the original stated "*It is a process (Work Process) that is composed of at least three interrelated Testing Activities conducted to facilitate the discovery of defects and/or the assessment of Characteristics and Attributes of a Testable Entity*". The new definition (see Table 1) clarifies that this concept refers to an Engineering Process. This change is aligned with ProjectCO v2.0, as it differentiates the concepts of management processes from engineering (technical) processes. This aspect is not made explicit terminologically in previous ontologies or glossaries for software testing.

6. Conclusions and Future Work

This paper has discussed an enhanced version of the TestTDO conceptualization. This domain-specific ontology for testing is located at the top-domain level within a five-tier architecture and is now harmonized and aligned with established ontologies, such as ProjectCO, a core-level project management ontology. While TestTDO v1.3 contained 44 terms and only 5 of them were semantically enriched with ProjectCO terms, TestTDO v2.0 includes 49 terms, of which 15 are stereotyped with project terms.

It is important to note that TestTDO v2.0 incorporates insights from a syntactic and semantic consistency study of testing glossaries, which included recognized standards such as ISO/IEC 29119, ISTQB, and TMMi. This ensures that the current ontology reflects widely accepted terms and, to some extent, definitions of terms in the field.

As a practical impact, we have demonstrated how TestTDO, when used as a Testing Vocabulary Specification, serves as a common reference vocabulary for both Testing Process Specifications and Testing Method Specifications. Consequently, the specifications of Testing Strategies ensure syntactical and terminological uniformity, as well as semantic consistency, facilitating a clearer understanding and communication of their processes, methods, procedures, and tools, among other benefits.

Just as we have developed a Testing Process Specification specifically for dynamic testing, in future work, we will develop specifications for processes and methods for static testing using TestTDO v2.0 as a Testing Vocabulary Specification.

References

Asman, A. and Srikanth, R. M. (2015). A Top Domain Ontology for Software Testing. Master Thesis, Jönköping University.

APM (2021). Association for Project Management. APM Glossary, Available at: https://www.apm.org.uk/resources/glossary/. Last accessed 2023/11/15.

Becker, P., Papa, M.F., Tebes, G., and Olsina, L. (2022). Discussing the Applicability of a Process Core Ontology and Aspects of its Internal Quality. Software Quality Journal, Springer, 30:(4), pp. 1003–1038. https://doi.org/10.1007/s11219-022-09592-3.

D'Aquin, M. and Gangemi, A. (2011). Is there beauty in ontologies?. Applied Ontology, 6:(3), pp. 165–175.

De Souza, É. F., De Almeida Falbo, R., and Vijaykumar, N. L. (2017). ROoST: Reference ontology on software testing. Applied Ontology, Vol. 12, Number 1, pp. 59–90.

Guizzardi, G. (2005). Ontological foundations for structural conceptual models. Ph.D. Thesis, Netherlands, Universal Press.

IAPM (2021). International Association of Project Managers Project Managers' Guide and Agile Project Managers' Guide. Available at: https://www.iapm-cert.net/documents/glossary-en/a.html. Last accessed 2023/11/15.

ISO (2013). International Organization for Standardization ISO/IEC/IEEE 29119-1: Software and systems engineering – Software Testing – Part 1: Concepts and definitions.

ISO (2022). International Organization for Standardization ISO/IEC/IEEE 29119-1: Software and systems engineering – Software Testing – Part 1: General concepts.

ISTQB (2021). International Software Testing Qualifications Board. Standard Glossary of Terms used in Software Testing, version 3.5.

ISTQB (2022). International Software Testing Qualifications Board. Standard Glossary of Terms used in Software Testing, version 3.6.1. Automatically generated on September 26, 2022 from https://glossary.istqb.org/en/reports.

Monfardini, G.K.Q., Salamon, J.S., and Barcellos, M.P. (2023). Use of Competency Questions in Ontology Engineering: A Survey. In: Almeida, J.P.A., et al. (eds) Conceptual Modeling, ER2023, LNCS, vol.14320, Springer, Cham. https://doi.org/10.1007/978-3-031-47262-6_3.

Olsina, L. (2023). The Foundational Ontology ThingFO: Architectural Aspects, Concepts, and Applicability. In: Fred, A., et al. (eds.) Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2021, Communications in Computer and Information Science, Vol. 1718, Springer, Chapter 5, pp. 73–99.

Olsina, L., Becker, P., and Papa, M.F. (2024a). The Project Management Ontology called ProjectCO: Architectural Aspects, Concepts, and Usefulness. In: CIbSE 2024, Curitiba, Brazil, SBC_OpenLib, pp. 61–75, https://doi.org/10.5753/cibse.2024.28439.

Olsina, L., Becker, P., Papa, M.F., and Rossi, G. (2025). Ontological Concepts of Non-Functional Requirements Applied to Particular Situations in Measurement and Evaluation Projects, CLEI Electronic Journal, Vol. 28, Number 3, Paper 7.

Olsina, L., Lew, P., and Becker, P. (2024b). Comparative analysis of the syntactic and semantic consistency of terms in software testing glossaries. Software Quality Journal 32, 27–52.

PRINCE2 (2017). 6th Edition Glossaries of Terms. Available at: https://www.axelos.com/resource-hub/glossary/prince2-6th-edition-glossaries-of-terms. Last accessed 2023/11/15.

PMI (2021). Project Management Institute. PMBOK® Guide, A Guide to the Project Management Body of Knowledge. 7th Edition. Available at: https://www.pmi.org/pmbok-guide-standards/foundational/pmbok?sc_camp=D750AAC10C2F4378CE6D51F8D987F49D. Last accessed 2023/11/15.

Tebes, G., Becker, P., and Olsina, L. (2024). TestTDO v2.0's Concepts, Relationships, and Constraints – A Top-Domain Software Testing Ontology. pp. 1–27, http://dx.doi.org/10.13140/RG.2.2.36663.94883.

Tebes, G., Olsina, L., Peppino, D., and Becker, P. (2021a). Specifying and Analyzing a Software Testing Ontology at the Top-Domain Ontological Level. Journal of Computer Science & Technology, 21:(2), pp. 126–145, https://doi.org/10.24215/16666038.21.e12.

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. Elsevier Journal, Information and Software Technology 123:1-23.

Tebes, G., Peppino, D., Becker, P., and Olsina, L. (2021b). A Situation-aware Scenario-based Testing Strategy. SCCC 2021 - 40th International Conference of the Chilean Computer Science Society, In: IEEE Xplore, La Serena, Chile (held virtually, 15-18 Nov. 2021) pp. 1–8.

TMMi Foundation (2018). Test Maturity Model Integration (TMMi®) – Guidelines for Test Process Improvement, Release 1.2.