ESTRATEGIA INTEGRADA DE PRUEBAS DE SOFTWARE CONSCIENTE DE LA SITUACIÓN Y BASADA EN ESCENARIOS

Ing. Guido Sebastian TEBES

Grupo de Investigación y Desarrollo en Ingeniería de Software y Web (GIDIS_Web)

Facultad de Ingeniería

Universidad Nacional de La Pampa

Director: **Dr. Luis Antonio OLSINA**

Co-Director: **Dr. Gustavo ROSSI**

Tesis presentada para obtener el grado de *Doctor en Ciencias Informáticas*

Facultad de Informática - Universidad Nacional de La Plata

Octubre 2023

RESUMEN

En la actualidad, las aplicaciones de software se han vuelto muy complejas ya que, en algunas situaciones particulares, dependen de otros sistemas o servicios para realizar correctamente sus funciones. En otras palabras, esto significa que un sistema no está aislado y está influenciado por entidades de contexto. Debido a su complejidad inherente, algunos enfoques o estrategias de pruebas de software existentes no son lo suficientemente efectivos para verificar y validar situaciones particulares en las que es relevante considerar y modelar las entidades de contexto. Además, hay un escaso número de metodologías que son útiles para probar este tipo de situaciones. Con la intención de contribuir en esta área, en esta tesis doctoral se propone una estrategia de pruebas de software basada en escenarios y consciente de la situación. Esta estrategia es consciente de la situación porque la situación debe ser modelada y considerada para producir casos de prueba. La misma fue inicialmente validada ya que se aplicó en dos empresas del ámbito privado por dos estudiantes de grado para sus proyectos finales de ingeniería.

Una estrategia es un recurso fundamental de una organización que define un curso específico de acción a seguir, es decir, especifica qué se debe hacer y cómo hacerlo. A su vez, una estrategia debería integrar tres capacidades fundamentales o pilares, a saber: i) una especificación de proceso, ii) una especificación de métodos, y iii) una especificación de base conceptual. Por lo tanto, la estrategia propuesta en este trabajo también considera estas tres capacidades. El beneficio de integrar estos tres pilares en una estrategia en particular es que la misma especificará qué actividades están involucradas, cómo llevarlas a cabo a través de métodos, y todo esto dentro de un marco semántico de un vocabulario de uso común y compartido.

Dado que una ontología es la representación más rica para modelar bases conceptuales, se considera que una estrategia integrada debería tener entonces una ontología como base conceptual y no meramente un glosario y/o taxonomía. Por ello, en esta tesis doctoral, se decidió desarrollar y utilizar una ontología de pruebas de software para dar soporte a la estrategia integrada. Además, las especificaciones de procesos y métodos deberían utilizar los conceptos que involucra esta base conceptual ontológica para que la estrategia sea consistente.

Por otro lado, es importante contar con procesos bien especificados como parte de una estrategia integrada. Un proceso bien especificado debería describir cuáles son las principales actividades que deben ser realizadas, sus productos de trabajo consumidos y producidos, qué roles intervienen, cuál es el flujo a seguir entre las diferentes actividades, entre otros aspectos. Además, otro aspecto que fortalece las especificaciones de procesos es el modelado de diferentes vistas o perspectivas de proceso. Como beneficio, un proceso bien especificado no solo permite el entendimiento del mismo, sino que también facilita la comunicación entre las partes interesadas. Además, asegura la repetibilidad y la reproducibilidad en la implementación de las actividades y tareas.

PALABRAS CLAVES: ontologías; estrategias integradas; pruebas de software; procesos; métodos; situación; contexto; pruebas de escenarios

AGRADECIMIENTOS

Esta tesis doctoral es el producto de mucho esfuerzo, dedicación y estudio. Me gustaría mencionar y agradecer a las personas que me acompañaron y apoyaron a lo largo de este trabajo.

En primer lugar, quiero agradecer a mi director, Luis Olsina, por su apoyo, dirección, disposición, dedicación y acompañamiento en este largo camino que decidí transitar. Durante el transcurso del mismo me brindó muchas recomendaciones, críticas constructivas, enseñanzas, y también compartió conmigo sus saberes, todo con el objetivo de mejorar este trabajo y mejorarme a mi como profesional. De la misma manera quiero agradecer a Gustavo Rossi, por codirigir esta tesis y brindarme el soporte logístico necesario.

También quiero agradecer a las personas que integran e integraron, durante este trabajo, el grupo de investigación GIDIS_Web. Ellos son Pablo Becker, Fernanda Papa, Belén Rivera y Denis Peppino, que me acompañaron y fueron también parte del trabajo de investigación de esta tesis doctoral. Quiero hacer una mención especial a mi gran amigo Denis Peppino por su apoyo, amistad y el trabajo compartido durante estos últimos años en el grupo de investigación, y también por su acompañamiento y compresión durante este último año en donde tuve que llevar a cabo la escritura de la tesis mientras trabajaba al mismo tiempo en una empresa, en el equipo el cual él lidera.

A su vez quiero agradecer y mencionar a mi pareja, Camila Frias, la cual tuvo que compartirme con este trabajo y muchas veces soportar mis ausencias. Gracias por el acompañamiento, cariño, comprensión y el apoyo en estos últimos años desde que empecé este doctorado. Además, quiero agradecer a mi familia por haber estado presente durante el transcurso de este trabajo y también por todo el apoyo, cariño y el acompañamiento que me dieron.

También quiero mencionar y agradecer a la Facultad de Ingeniería de la UNLPam por darme un espacio de trabajo y contribuir con parte del financiamiento para realizar este doctorado. Otra institución a la cual quiero agradecer que contribuyo económicamente con este trabajo fue el FONCyT a través de una beca otorgada.

Por último, quiero agradecer a todas las personas que, de una u otra manera, también me han ayudado y acompañado a lo largo de estos años.

General Pico, La Pampa, 30 de mayo de 2023

Guido Sebastian Tebes

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

1.1 Introducción y Motivación

Actualmente, los sistemas de software son de mucha utilidad para sus usuarios, permitiéndoles adquirir información de interés o realizar ciertas acciones específicas. Por ejemplo, en una rutina normal de muchas personas, es común levantarse por la mañana y utilizar sus dispositivos móviles (u otros dispositivos) para acceder a portales de noticias online, observar las condiciones climáticas del día, el tráfico, el correo electrónico, entre otras acciones. Generalmente, estas aplicaciones de software no están aisladas y están influenciadas por otros sistemas y/o servicios externos. Por lo tanto, estas aplicaciones deben estar conscientes de su contexto o entidades de contexto que interactúan con ellas para realizar correctamente sus tareas.

Cuando se produce una falla en un sistema de software, la misma puede impactar negativamente al usuario, ya sea causando que pierda confianza en la aplicación o incluso conducir al abandono de la misma. Además, si es una aplicación crítica, puede implicar grandes pérdidas económicas o, peor aún, la pérdida de vidas humanas. Con el objetivo de prevenir estas situaciones, los ingenieros de software deben verificar y validar los sistemas de software de antemano, es decir, antes que lleguen a utilizarlos los usuarios reales.

Verificación y validación involucra un conjunto de actividades, herramientas, métodos y estrategias que dan soporte al aseguramiento de la calidad. En este sentido, las pruebas de software (i.e., *Software Testing* o, simplemente *testing*) son un enfoque útil que da soporte a la verificación y validación de sistemas de software (Bourque & Fairley, 2014; ISO, 2013a). Es importante mencionar que las pruebas de software utilizan procesos, métodos y estrategias los cuáles ayudan a verificar el cumplimiento de requisitos funcionales y no funcionales de aplicaciones de software.

En la actualidad, existen muchos métodos o técnicas de pruebas, tanto para pruebas dinámicas como estáticas. Algunas de ellas son técnicas basadas en la especificación, o también conocidas como técnicas de caja negra, otras basadas en la estructura (caja blanca) (ISO, 2015). Con respecto a pruebas estáticas, existen métodos como *walkthroughs*, inspecciones, revisiones técnicas, entre otros (IEEE, 2008a).

A pesar de que existen muchos métodos o técnicas para las pruebas de software, no todos ellos son lo suficientemente adecuados para probar el software en diferentes situaciones en las cuales es importante modelar las entidades de contexto que influencian el objeto de prueba. Además, como mencionan los autores en (Amalfitano et al., 2019), hay un escaso número de tecnologías (es decir, métodos, herramientas, estrategias) que son útiles para probar situaciones en las que es importante considerar las entidades de contexto que interactúan con el objeto de prueba. Un enfoque que puede ser de utilidad para este propósito son las pruebas de escenarios, ya que al usar esta metodología se consideran modelos de situación para diseñar los casos de prueba, los cuales podrían capturar/modelar información importante de contexto.

Las pruebas de escenarios se mencionan en muchos documentos como en (Bourque & Fairley, 2014; ISO, 2013a, 2015; ISTQB, 2021b; Sommerville, 2011; TMMi Foundation, 2018). Según el glosario estándar de términos usados en pruebas de software (*Standard Glossary of Terms used in Software Testing*) publicado por *The International Software Testing Qualifications Board* (ISTQB) (ISTQB, 2021b), las pruebas de escenarios son un

tipo de técnica de prueba de caja negra en la cual los casos de pruebas se diseñan para ejecutar escenarios de casos de uso. Más ampliamente, ISO 29119-1 (ISO, 2013a) define las pruebas de escenarios como una clase de técnica de diseño de pruebas en la cual las pruebas son diseñadas para ejecutar escenarios individuales. Además, indican que un escenario puede ser una historia de usuario, un caso de uso, una secuencia de eventos, entre otros.

En resumen, considerando las definiciones de pruebas de escenarios presentadas en (ISO, 2013a; ISTQB, 2021b), podemos destacar que son un método de diseño de casos de pruebas basado en especificaciones (caja negra), en el cual estas especificaciones son llamadas escenarios. Además, estos escenarios representan las diferentes situaciones particulares que involucran al objeto bajo prueba (también llamado *Testable Entity* en (Tebes, Olsina, et al., 2021)) y las entidades de contexto de prueba relacionadas. Notar que estos escenarios o situaciones deben estar especificadas o modeladas, ya sea por ejemplo usando casos de uso, o algún otro diagrama como diagramas de comunicación o de secuencia.

Por otra parte, según los autores de (Olsina & Becker, 2017), una estrategia sistemática y bien especificada que ayude a alcanzar metas de negocio debería ser un recurso clave de una organización que defina un curso de acción a seguir. En otras palabras, debería especificar qué hacer y cómo hacerlo. Además, es importante que se considere un vocabulario común a utilizar dentro de una estrategia, el cual de soporte a los distintos conceptos que involucran los procesos y métodos especificados. Por lo tanto, una estrategia debería integrar tres capacidades fundamentales o pilares: una especificación de proceso, una especificación de método y una especificación de base conceptual robusta (por ejemplo, estructurada como una ontología). Como beneficio de integrar estas tres capacidades, para una estrategia en particular se conocería qué actividades están involucradas, cómo llevarlas a cabo a través de métodos, y todo esto dentro de un marco semántico de un vocabulario de uso común y compartido.

Considerando el primer pilar o capacidad, es decir, una especificación de proceso, el mismo debe prescribir el conjunto de actividades principales, las entradas y salidas, roles e interdependencias, entre otros aspectos. Generalmente, un modelo de proceso dice qué se debe hacer, pero no cómo debe hacerse, es decir, no especifica qué métodos, técnicas o herramientas deben ser utilizados para llevar a cabo las actividades. Además, una especificación robusta de proceso debe incluir modelos del proceso desde diferentes vistas o perspectivas. De esta manera, se asegura la repetibilidad y reproducibilidad en la implementación de las actividades, consistencia en los resultados y, a su vez, se facilita el entendimiento y comunicación entre las partes interesadas.

La segunda capacidad, la cual incluye métodos y herramientas, permite llevar a cabo y automatizar las descripciones de las actividades prescriptas/informadas en un modelo de proceso. Los métodos deben ser asignados de una forma flexible para realizar las actividades especificadas y son usualmente automatizados por herramientas.

Por último, pero no menos importante, el tercer pilar se refiere a un marco conceptual que sea flexible y terminológicamente consistente. Un marco bien establecido debería estar construido sobre una base conceptual, como por ejemplo pueden ser glosarios, taxonomías u ontologías. Una base conceptual es robusta si especifica de manera explícita los conceptos, sus propiedades y relaciones, y las restricciones acordadas para un dominio particular. Esta capacidad contribuye a que no se generen ambigüedades y favorece la comunicación, entendimiento y la especificación de los procesos y métodos de una

estrategia. Por ello, es importante que las especificaciones de métodos y procesos usen los términos que están explícitamente definidos en el marco conceptual.

En (Becker et al., 2019; Olsina & Becker, 2017; Tebes, Peppino, Becker, et al., 2018) se ilustra una familia de estrategias, que integran las tres capacidades anteriormente mencionadas, la cual ayuda a alcanzar diferentes propósitos de evaluación. Teniendo en cuenta que existen estrategias para propósitos de evaluación, se podrían confeccionar también otras estrategias que ayuden a alcanzar propósitos de verificación y validación, y que estén dirigidas por actividades de *testing*. Además, teniendo en cuenta los beneficios que proporciona una estrategia integrada, una estrategia de *testing* también debería contar con estas tres capacidades.

Otros trabajos que ilustran enfoques/estrategias para pruebas de software también consideran especificaciones de procesos, especificaciones de métodos y un marco conceptual que brinda soporte semántico. Por ejemplo, ISO 29119 cuenta con un conjunto de partes en las cuales, la parte 2, está enfocada en procesos de pruebas dinámicas (ISO 29119-2 (ISO, 2013b)), la parte 4 en métodos/técnicas de diseño de pruebas (ISO 29119-4 (ISO, 2015)), y la parte 1 contiene la base conceptual estructurada como un glosario (ISO 29119-1 (ISO, 2013a)). Otro ejemplo podría ser la organización ISTQB, la cual dispone de un conjunto de documentos donde se especifican procesos y métodos de prueba (ISTQB, 2018, 2021a). Al igual que ISO 29119, también estructuran la base conceptual como un glosario (ISTQB, 2021b).

Por lo tanto, considerando que existen pocos enfoques de prueba que sean útiles para probar sistemas de software donde sea relevante tener en cuenta la situación con las entidades de contexto involucradas, y que las pruebas de escenarios pueden ser útiles para probar estos tipos de sistemas, en este trabajo se consideró desarrollar una estrategia de *testing* que siga el enfoque de pruebas basadas en escenarios. Además, teniendo en cuenta que una estrategia sistemática y bien especificada debería integrar las tres capacidades anteriormente mencionadas, esta estrategia a desarrollar también debería contar con ellas. Esta estrategia propuesta recibe el nombre de Estrategia de Prueba basada en Escenarios y consciente de la Situación (*Situation-aware Scenario-based Testing Strategy: SaST Strategy*).

Es importante mencionar que la estrategia a desarrollar va a estar soportada semánticamente por un conjunto de ontologías que pertenecen a una arquitectura ontológica llamada FCD-OntoArch (*Foundational, Core, and Domain Ontological Architecture for Sciences* (Olsina, 2021)). Esta es una arquitectura ontológica de 5 capas, que considera los niveles fundacional, central (*core*), dominio de alto nivel (o dominio superior), dominio de bajo nivel, e instancia. La estrategia SaST utiliza principalmente conceptos que provienen de una ontología de dominio de alto nivel llamada TestTDO (*Top-Domain Ontology for Software Testing*) (Tebes, Olsina, et al., 2021). Dado que una ontología es más rica estructuralmente que un glosario, se consideró utilizar ontologías como base conceptual, a diferencia de otras propuestas como ISO e ISTQB.

1.2 Objetivos

En esta sección se especifican los objetivos generales y específicos a ser alcanzados. Se consideró que un objetivo general puede ser dividido en objetivos específicos, y que, si se alcanzan todos sus objetivos específicos, entonces se alcanzará el objetivo general. Estos objetivos fueron desarrollados teniendo en cuenta la motivación de este trabajo, la cual fue comentada en la sección anterior.

- Objetivo general: Desarrollar una estrategia dirigida por actividades de *testing* que ayude a las organizaciones a alcanzar propósitos de *testing* como verificar, validar, encontrar defectos, vulnerabilidades, entre otros. Esta estrategia debe integrar las tres capacidades anteriormente mencionadas, a saber: una especificación de proceso, una especificación de método, y una base conceptual robusta estructurada como una ontología. Además, la estrategia debe ser útil para probar situaciones en las que es importante considerar entidades de contexto. También, la estrategia debe seguir el enfoque de pruebas basadas en escenarios.

Para alcanzar el objetivo general anterior, se propusieron los siguientes objetivos específicos (OE):

- OE_1: dado que la estrategia a desarrollar debe estar semánticamente soportada por una ontología, se debe investigar cuáles son las ontologías existentes para el dominio de las pruebas de software.
- OE_2: Una vez conocido el estado del arte de las ontologías de pruebas de software (i.e., una vez alcanzado el OE_1), se debe adoptar/adaptar alguna existente si cumple con los requisitos establecidos. En caso de que ninguna ontología existente cumpla con los requisitos establecidos, se deberá desarrollar una nueva.
- OE_3: Haciendo uso de los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de proceso que considere actividades de *testing* de diseño, ejecución y análisis, y además especifique cuáles son sus artefactos de entrada y salida correspondientes. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.
- OE_4: Usando los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de método que este basado en la técnica de diseño de pruebas basada en escenarios. Además, debe dar soporte a situaciones que consideren entidades de contexto. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.
- OE_5: Una vez confeccionada la estrategia integrada, se debe validar la misma para inicialmente evaluar su utilidad. Notar que se deben haber alcanzado primero los objetivos OE_3 y OE_4 para poder lograr este objetivo.

1.3 Contribuciones

Esta sección está destinada a describir las contribuciones de este trabajo de doctorado. A continuación, en la Tabla 1-1, se resume cada una de las contribuciones realizadas relacionadas a cada objetivo específico descrito en la sección anterior. Con la intención de facilitarle la lectura al lector de este trabajo, los objetivos específicos fueron replicados en la Tabla 1-1.

El primer objetivo a alcanzar era conocer el estado del arte de las ontologías de pruebas de software y, para ello, llevamos a cabo una Revisión Sistemática de Literatura (RSL) sobre ontologías de pruebas de software (Tebes et al., 2020). Para la realización de la misma, se tuvo que estudiar cuál era el proceso para llevar a cabo un estudio secundario, el cual se encuentra inicialmente documentado en (Brereton et al., 2007; B. Kitchenham, 2004a). Observando las guías y procedimientos propuestos en los trabajos citados, vimos una oportunidad de mejora en la especificación del proceso debido a las dificultades que se tuvieron a la hora de realizar el estudio secundario sobre ontologías de pruebas de software. Por lo tanto, como una contribución no directamente relacionada con este

trabajo de doctorado, se especificó un proceso en SPEM el cual considera distintas vistas de modelado de proceso y que indica qué hacer a la hora de realizar un estudio secundario. Más detalles del mismo se pueden encontrar en el Capítulo 3 o en (Olsina et al., 2019).

Tabla 1-1: Contribuciones de este trabajado relacionadas a cada objetivo específico.

Objetivos específicos (OE)	Contribuciones (C)
OE_1: dado que la estrategia a desarrollar debe estar semánticamente soportada por una ontología, se debe investigar cuáles son las ontologías existentes para el dominio de las pruebas de software.	C_1: Una especificación de proceso para revisiones/mapeos sistemáticos de literatura. C_2: Una Revisión Sistemática de Literatura sobre ontologías de pruebas de software.
OE_2: Una vez conocido el estado del arte de las ontologías de pruebas de software (i.e., una vez alcanzado el OE_1), se debe adoptar/adaptar alguna existente si cumple con los requisitos establecidos. En caso de que ninguna ontología existente cumpla con los requisitos establecidos, se deberá desarrollar una nueva.	C_3: Una especificación de proceso para la metodología de investigación de *Design Science Research.* C_4: Una nueva ontología de dominio de nivel superior para pruebas de software llamada TestTDO.
OE_3: Haciendo uso de los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de proceso que considere actividades de *testing* de diseño, ejecución y análisis, y además especifique cuáles son sus artefactos de entrada y salida correspondientes. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.	C_5: Una especificación de proceso para la estrategia SaST haciendo uso del lenguaje SPEM (*Software & Systems Process Engineering Metamodel* (OMG, 2008)), el cual considera perspectivas de proceso.
OE_4: Usando los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de método que este basado en la técnica de diseño de pruebas basada en escenarios. Además, debe dar soporte a situaciones que consideren entidades de contexto. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.	C_6: Una especificación de método para la estrategia SaST, la cual es una plantilla que captura información relacionada a la situación/escenario de prueba, entre otras cosas.
OE_5: Una vez confeccionada la estrategia integrada, se debe validar la misma para inicialmente evaluar su utilidad. Notar que se deben haber alcanzado primero los objetivos OE_3 y OE_4 para poder lograr este objetivo.	C_7: la ilustración de la aplicación de la estrategia en un contexto real solo para el diseño de los casos de pruebas y escenarios. C_8: la presentación de otra aplicación de la estrategia, también en un contexto real, la cual incluyo el diseño, implementación, automatización y ejecución de las pruebas.

Una vez realizada la RSL sobre ontologías de pruebas de software, llevamos a cabo la construcción de una ontología de *testing* dado que ninguna de las existentes satisfacía en gran medida los requisitos establecidos. La ontología desarrollada recibe el nombre de TestTDO (Tebes, Olsina, et al., 2021). A su vez, se propuso utilizar la metodología *Design Science Research* (DSR) como enfoque general que guie la construcción de TestTDO. Observando los trabajos publicados que documentan el proceso de DSR (A. R. Hevner et al., 2004; Wieringa, 2014), vimos una oportunidad de mejora en la especificación de su proceso, como también sucedió con el proceso para llevar a cabo estudios secundarios. En consecuencia, como otra contribución que no está directamente relacionada con este trabajo de doctorado, se especificó el proceso de DSR. Los detalles del mismo se pueden encontrar en el Capítulo 4 o en (Tebes et al., 2020).

Una vez construida la ontología de pruebas de software se procedió a construir las especificaciones de proceso y método para la estrategia SaST (Tebes, Peppino, et al., 2021). Finalmente, una vez obtenida la estrategia integrada, la misma se aplicó dos veces por estudiantes de grado en el marco del proyecto final de la carrera Ingeniería en Sistemas. En estas dos aplicaciones de la estrategia SaST se realizaron, por un lado, el diseño de pruebas de integración para un sistema que se está desarrollando en una empresa relacionada a la agroindustria (Eleicegui, 2022). Por otro lado, se diseñaron, implementaron, automatizaron y ejecutaron pruebas de sistema para un software de gestión de una empresa inmobiliaria (Fernandez Reale, 2023).

1.4 Organización de la Tesis

El resto de la tesis doctoral se estructura de la siguiente manera. El Capítulo 2 tiene por objetivo dar al lector una base sobre distintos aspectos que serán útiles para comprender mejor los capítulos siguientes. En primer lugar, se resalta la relevancia de llevar a cabo procesos de *testing* en las organizaciones. Luego se exponen los aspectos fundamentales referentes a estrategias integradas, haciendo énfasis en los tres pilares que se consideran que una estrategia de *testing* debe tener de manera conjunta para ser eficaz, repetible y consistente, a saber: una especificación de proceso, una especificación de método y un marco conceptual robusto. Por otro lado, se introduce al lector en el modelado de proceso, mostrando sus beneficios y describiendo las diferentes vistas o perspectivas que se pueden tener presente al momento de modelar un proceso. Por último, se justifica la importancia de contar con una base conceptual ontológica como marco conceptual de una estrategia integrada.

En el Capítulo 3 se presenta el estado del arte relacionado a ontologías de pruebas de software. Para ello, se llevó a cabo una RSL y también se describe en dicho capítulo un proceso propuesto para realizar estudios secundarios. Este proceso fue utilizado para conducir la RSL sobre ontologías de *testing*. Al analizar los resultados obtenidos de la RSL se concluyó que es necesario construir una nueva ontología que de soporte como marco conceptual a una familia de estrategias integradas de *testing*. Por lo tanto, en el Capítulo 4 siguiente, se detalla un proceso propuesto de DSR que se utilizó para construir la ontología de *testing* llamada TestTDO. Además, se describe la conceptualización de TestTDO y las actividades de verificación y validación que se llevaron a cabo para asegurar la calidad de dicha ontología.

Luego, en el Capítulo 5, se especifica el proceso y el método de la estrategia SaST, ambos soportados semánticamente por la ontología TestTDO. A su vez, en este capítulo se muestran dos aplicaciones de la estrategia llevadas a cabo por alumnos en el marco de sus tesinas de grado para la carrera de Ingeniería en Sistemas. Finalmente, en el Capítulo 6, se presentan las conclusiones de esta tesis doctoral y posibles trabajos futuros de investigación.

Es importante también mencionar que como parte de esta tesis se incluyeron seis apéndices los cuales contienen, entre otras cosas, detalles de los artefactos producidos al verificar y validar la ontología TestTDO. Por un lado, el Apéndice A tiene la vista organizacional para el proceso propuesto de RSL. Por su parte, el Apéndice B, tiene trabajos relacionados a DSR. Por otro lado, el Apéndice C muestra una matriz de verificación estática para los requisitos del alcance de TestTDO. El Apéndice D describe una lista de verificación también estática pero aplicada a verificar las relaciones de la ontología. El Apéndice E contiene una prueba de concepto utilizando los términos de

TestTDO. Por último, el Apéndice F documenta casos de prueba que se utilizaron para verificar dinámicamente dicha ontología.

CAPÍTULO 2: FUNDAMENTOS SOBRE ESTRATEGIAS INTEGRADAS DE PRUEBAS DE SOFTWARE

2.1 Introducción

Dado que el objetivo general de este trabajo es desarrollar una estrategia dirigida por actividades de *testing* que integre las tres capacidades ya mencionadas (una especificación de proceso, una especificación de método, y una especificación de una base conceptual robusta), este capítulo fundamenta, en la Sección 2.3, la importancia de contar con estrategias integradas. Además, en dicha sección se ilustra un resumen de estrategias de *testing* existentes en la literatura. Por otro lado, en la Sección 2.2, se justifica la importancia de las pruebas de software en las organizaciones citando referencias relevantes en el área. Notar que la base conceptual de una estrategia integrada es uno de los pilares fundamentales o cimientos dado que la terminología de las especificaciones de procesos y métodos deberían basarse en ella de un modo consistente. Por ello y finalmente, en la Sección 2.4, se justifica la importancia de contar con una base conceptual ontológica.

2.2 Relevancia de las Pruebas de Software en las Organizaciones

Un aporte relevante para la Ingeniería de Software es la Guía del *Software Engineering Body of Knowledge* (SWEBOK Guide) (Bourque & Fairley, 2014). El propósito de esta guía es describir, organizar y proveer acceso a la porción del cuerpo de conocimiento de la Ingeniería de Software que es comúnmente aceptada. Notar que el cuerpo de conocimiento en su totalidad existe en toda la literatura publicada. SWEBOK, en su versión 3.0, describe 15 áreas de conocimiento, entre las cuales se encuentra las pruebas de software. Además, cada una de estas áreas se desglosan en un conjunto de tópicos. Para el área de conocimiento *software testing* se documentan los tópicos y subtópicos que se ilustran en la Figura 2-1. Por ejemplo, se encuentra el tópico *Test Techniques*, y como subtópico tenemos *Model-based Techniques*, las cuales son técnicas de diseño de pruebas de caja negra basadas en modelos, es decir, basadas en representaciones abstractas del objeto a ser probado o de sus requisitos. Es importante mencionar que SWEBOK incluye como técnica basada en modelos a las pruebas basadas en modelos de flujo de trabajo (*workflow models*). Un modelo de flujo de trabajo especifica una secuencia de actividades/acciones realizadas por agentes, y es generalmente representado de forma gráfica. SWEBOK denomina "escenarios" a estos modelos de flujo de trabajo. En conclusión, las pruebas basadas en escenarios son consideradas como un tópico relevante dentro del cuerpo de conocimiento de SWEBOK Guide.

Además, en el estándar *de facto* CMMI (CMMI, 2010) se documentan 5 niveles de madurez para las organizaciones, a saber: 1) Inicial –*Initial*-, 2) Gestionado –*Managed*-, 3) Definido –*Defined*-, 4) Gestionado Cuantitativamente –*Quantitatively Managed*- y 5) En Optimización –*Optimizing*. Dentro del nivel de madurez 3, se tienen las áreas de proceso de validación y verificación, las cuales están directamente relacionadas con las pruebas de software. Si observamos el área de proceso de validación, la misma establece una práctica especifica llamada "establecer criterios y procedimientos de validación", en donde se recomienda utilizar casos de prueba y procedimientos para pruebas de aceptación. A su vez, el área de verificación incluye métodos de pruebas de software

también para alcanzar sus objetivos. En esta área se recomienda utilizar métodos de pruebas estáticas como revisiones de pares y *walkthroughs*, y realizar pruebas no funcionales (por ejemplo, pruebas de carga, de rendimiento, entre otras).

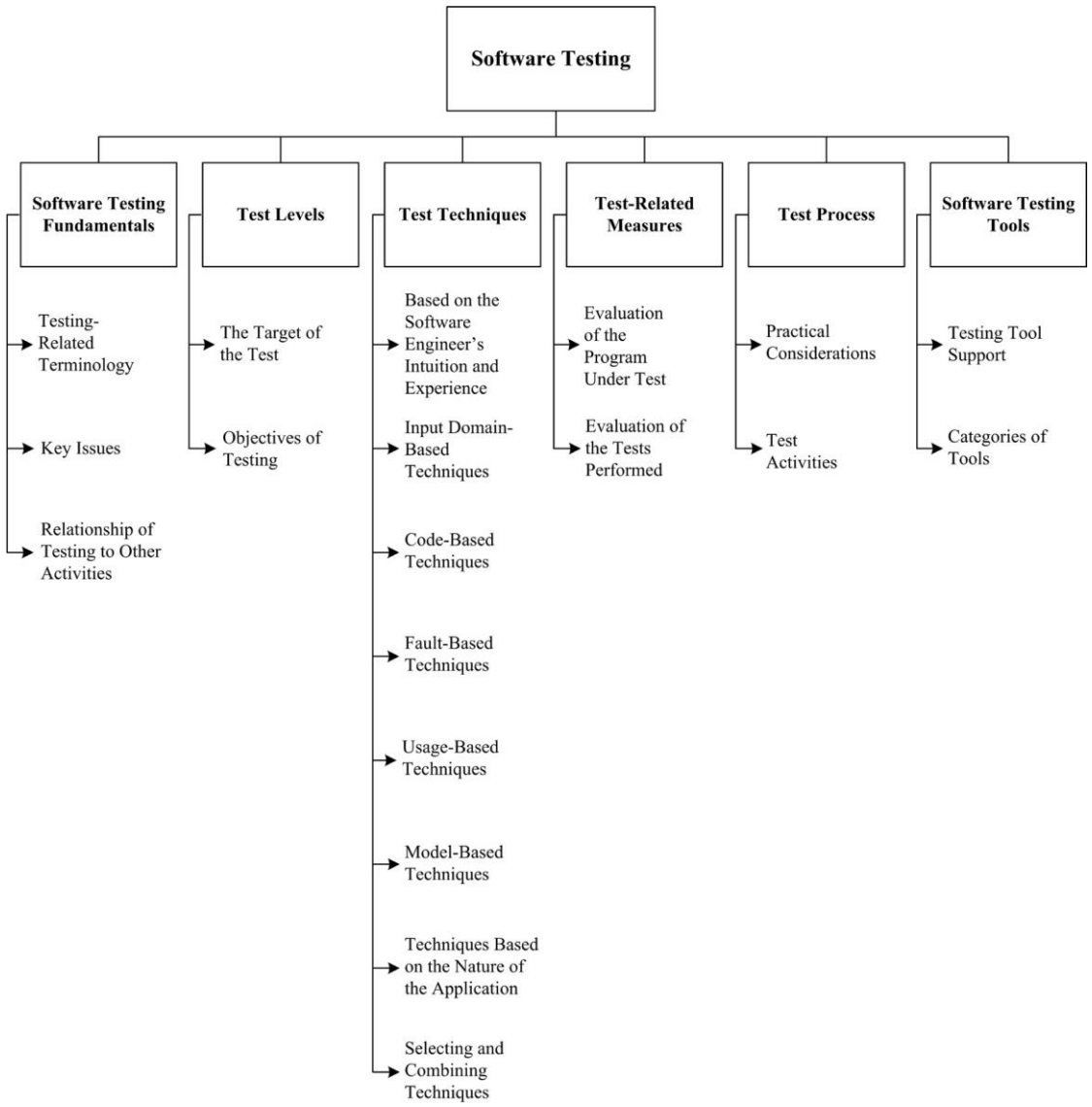


Figura 2-1: Desglose de tópicos para el área de conocimiento de las pruebas de software. Figura tomada de (Bourque & Fairley, 2014).

En este punto es importante mencionar que, para complementar a CMMI, existe TMMi (TMMi Foundation, 2018), el cual contiene guías para la mejora de los procesos organizacionales de pruebas de software. En (TMMi Foundation, 2018) se afirma que las actividades de *testing* se han convertido en un aspecto clave que directamente influencia no solo la calidad del producto de software, sino que también el rendimiento de todo el proceso de desarrollo. TMMi cuenta también con 5 niveles de madurez, a saber: 1) Inicial –*Initial*-, 2) Gestionado –*Managed*-, 3) Definido –*Defined*-, 4) Medido–*Measured*- y 5) en Optimización –*Optimization*. A su vez, cada nivel tiene un conjunto de áreas de procesos que una organización necesita implementar para alcanzar ese nivel de madurez, las cuales se pueden observar en la Figura 2-2.

En noviembre de 2002 se fundó la organización internacional llamada *International Software Testing Qualifications Board* (ISTQB) (ISTQB, 2022), la cual esta soportada por cientos de expertos en pruebas de software. Esta organización promueve el valor de

las pruebas de software como profesión tanta para individuos como para organizaciones. Además, se encarga de certificar y capacitar profesionales de *testing*. Al día de la fecha, se han realizado aproximadamente un total de 1.065.000 de exámenes bajo esta organización, y la misma emitió un total de 774.000 certificados. A su vez, ISTQB brinda gratuitamente un cuerpo de conocimiento sobre pruebas, el cual es constantemente actualizado con las mejores prácticas disponibles en la industria y la investigación más innovadora. Dentro de este cuerpo de conocimiento se encuentra, por ejemplo, el glosario estándar de términos usados en pruebas de software (ISTQB, 2021b).

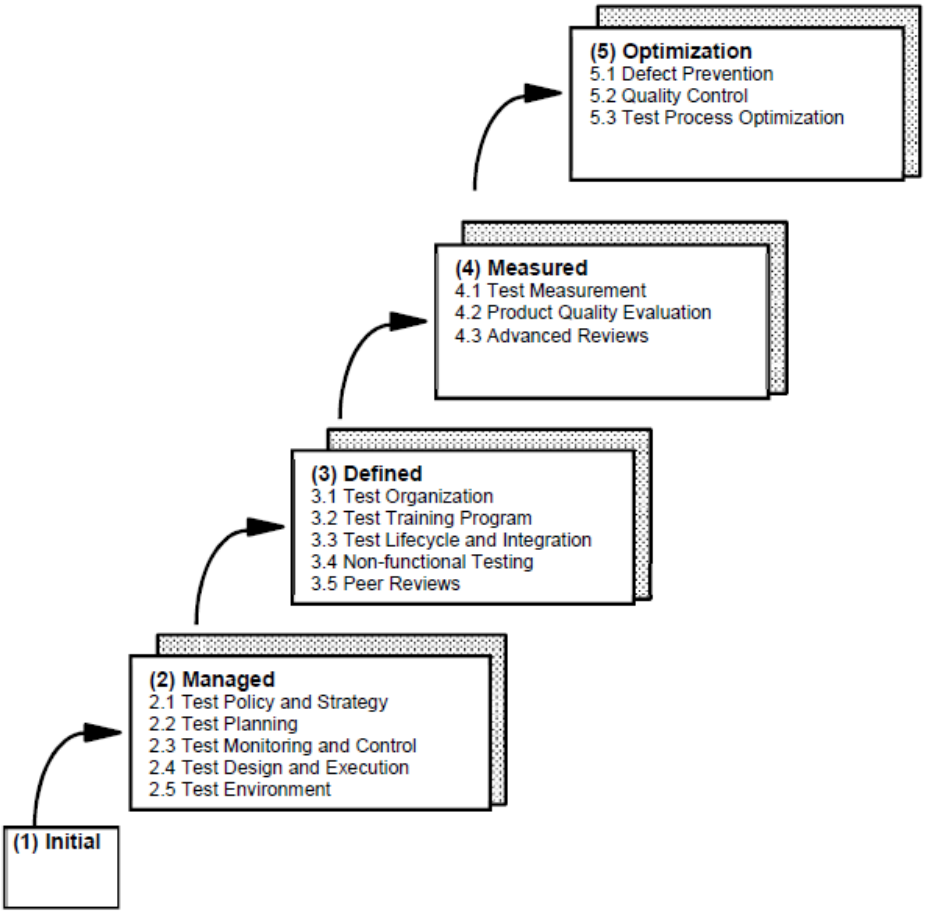


Figura 2-2: Niveles de madurez y sus áreas de proceso en TMMi. Figura tomada de (TMMi Foundation, 2018).

Por otra parte, existen un conjunto de estándares internacionales relacionados con las pruebas de software. Uno muy relevante es la ISO 29119, la cual cuenta al día de la fecha con 5 partes que definen un conjunto de conceptos (ISO, 2013a), procesos (ISO, 2013b), documentos de pruebas (ISO, 2013c), técnicas (ISO, 2015), y un enfoque de especificación de pruebas llamado *keyword-driven testing* (ISO, 2016), el cual da soporte para la automatización de las pruebas. Este estándar cubre solamente procesos de pruebas dinámicas, aunque recomienda utilizar el estándar IEEE 1028-2008 (IEEE, 2008a) que especifica procedimientos para pruebas estáticas y auditorías a ser aplicadas en organizaciones de desarrollo de software.

En conclusión, las pruebas de software cumplen un rol fundamental en las organizaciones de desarrollo de software, aportando a la mejora de la calidad de los productos de software. Al día de la fecha existen mucho estándares y organizaciones de relevancia que tratan las pruebas de software, las cuales fueron mencionadas y citadas en los párrafos anteriores.

2.3 Estrategias Integradas de Pruebas de Software

Como se indicó en la sección anterior, las pruebas de software desempeñan un papel muy importante en las organizaciones de desarrollo de software. Estas permiten obtener información que es útil para la mejora de la calidad de los productos y sistemas. Las organizaciones que establezcan programas de *testing* y/o estrategias de aseguramiento de calidad deberían establecer un conjunto de actividades y procedimientos para llevar adelante procesos de pruebas de software. También es necesario asegurar, con fines de análisis, que los resultados de las pruebas sean reproducibles y comparables entre distintos proyectos de *testing*, por lo tanto, es importante contar con procesos y métodos a seguir que estén especificados y bien documentados. Teniendo en cuenta lo anterior, para diseñar e implementar programas y/o proyectos de *testing* que sean robustos y consistentes, es deseable contar con una estrategia –o enfoque- de *testing* que integre los siguientes tres pilares o capacidades (Olsina & Becker, 2017):

- Una **especificación de proceso** que describa cuáles son las principales actividades de *testing* que deben ser realizadas, sus productos de trabajo consumidos/producidos, qué roles intervienen, entre otros aspectos. Es importante que se consideren diferentes vistas o perspectivas de proceso a la hora de modelar los procesos.
- Una **especificación de métodos y herramientas** que permitan llevar a cabo las tareas del proceso de *testing*.
- Una **especificación de base conceptual** para el dominio de las pruebas de software que sea flexible y consistente. Es deseable que esté basada en una ontología dado que estructuralmente es más rica desde un punto de vista de representación semántica.

Al momento de realizar una actividad de *testing* (por ejemplo, diseñar las pruebas, ejecutar las pruebas, entre otras) puede suceder que los principales agentes involucrados no conozcan qué insumos se necesitan para llevar a cabo la actividad, o qué se espera producir en ella. Incluso, podría suceder que no se conozca puntualmente qué se debe hacer. Una solución a estos problemas es contar con una **especificación de proceso** que considere diferentes vistas o perspectivas de proceso. Modelar el proceso considerando diferentes vistas permite identificar qué se debe hacer, en qué secuencia, quiénes son los responsables o roles involucrados, y cuáles son los productos de trabajo consumidos/producidos. A su vez, un proceso bien especificado no solo permite el entendimiento del mismo, sino que también facilita la comunicación entre las partes interesadas. Además, asegura la repetibilidad y la reproducibilidad en la implementación de las actividades/tareas.

Teniendo en cuenta que aún no existe un consenso generalizado en la terminología utilizada para el dominio de procesos de trabajo, se presenta el significado de algunos términos los cuales son utilizados en este trabajo. Notar que la terminología utilizada en esta tesis para los conceptos relacionados a procesos es la provista por ProcessCO (Becker et al., 2022), una ontología independiente del dominio (a nivel *core*) para procesos de trabajo. Además, para un acceso rápido a los conceptos definidos de ProcessCO, el lector puede acceder a https://arxiv.org/abs/2108.02816. Estos términos serán resaltados la primera vez que se mencionen en el texto en *cursiva*.

En este trabajo, un *proceso* se compone de *actividades*. A su vez, una actividad puede descomponerse en *tareas* y/o en actividades de menor nivel de granularidad denominadas *subactividades*. Una tarea se considera un elemento atómico (es decir, no se puede

descomponer). Además, el proceso, la actividad y la tarea se consideran *entidades de trabajo*, las cuales indican "qué" hacer. Cada entidad de trabajo (proceso/actividad/tarea) consume, modifica y/o produce *productos de trabajo*. Un tipo particular de producto de trabajo es un *artefacto* (por ejemplo, diagramas, documentos, entre otros). Adicionalmente, los *métodos* son *recursos de trabajo* que indican "cómo" llevar a cabo la *descripción* de una entidad de trabajo. Según ProcessCO, muchos métodos pueden ser aplicables a una misma descripción de un trabajo. Por último, un *agente* es quién realiza una tarea en cumplimiento de un *rol*. A su vez, el término rol se define como un conjunto de habilidades, competencias y responsabilidades que un agente debe tener para poder realizar una determinada tarea.

Como se mencionó anteriormente, es importante que se consideren diferentes vistas o perspectivas de proceso a la hora de modelar los procesos. Curtis *et al*. (Curtis et al., 1992) propone cuatro vistas o perspectivas de modelado de procesos. A continuación se explican brevemente cada una de ellas:

- **Vista funcional**: se describen qué actividades se deben llevar a cabo y qué entidades de productos (por ejemplo, productos de trabajo como artefactos y resultados) son necesarios para realizar las actividades, y cuáles son los productos de trabajo producidos por ellas.
- **Vista de comportamiento**: se especifica cuándo y cómo deben ejecutarse las actividades, lo cual incluye identificar secuencias, paralelismos, sincronización, iteraciones, etc., y bajo qué condiciones son realizadas las actividades.
- **Vista organizacional**: tiene como fin mostrar quiénes son los agentes (en cumplimiento de roles) que intervienen en la realización de las actividades.
- **Vista informacional**: se centra en la estructura de los productos de trabajo producidos o requeridos por las actividades, en sus interrelaciones, y en las estrategias de gestión de cambios de los productos de trabajo y seguimiento de los mismos.

En la gran mayoría de los casos, no es computacionalmente viable probar un objeto de prueba con todas las entradas posibles y sus combinaciones. Por ello, es importante contar con un soporte metodológico (conjunto de métodos o técnicas) que permita diseñar un conjunto reducido de casos de prueba para probar un objeto de prueba, y que estos casos tengan buenas posibilidades de detectar defectos. Contar con un conjunto específico de **métodos** permite tener una guía clara de cómo deben realizarse las actividades/tareas de *testing* especificadas. A su vez, utilizar **herramientas** automáticas o semiautomáticas para realizar las tareas puede proveer una mayor confiabilidad en los resultados obtenidos y abaratar costos.

Actualmente, existen muchos **métodos** o técnicas de diseño de pruebas las cuales dan soporte para seleccionar un conjunto de entradas de los casos de prueba y que estas entradas tengan una alta posibilidad de detectar un defecto. ISO 29119-4 (ISO, 2015) clasifica las técnicas en tres grupos (ver Figura 2-3): técnicas basadas en especificaciones (técnicas de caja negra), técnicas basadas en la estructura (técnicas de caja blanca), y técnicas basadas en la experiencia. Las técnicas de caja negra utilizan especificaciones como requisitos, diseños arquitecturales, entre otros, para diseñar las pruebas, pero no la estructura interna del objeto de prueba, como puede ser el código fuente. Por ejemplo, particiones de equivalencias (*Equivalence Partitioning* en Figura 2-3) es una técnica de caja negra la cual considera particionar el dominio de las entradas/salidas del objeto de prueba (probablemente especificado en un requisito) en clases de equivalencias, donde se espera que cada valor de cada partición sea tratado razonablemente similar por el objeto

de prueba. Luego, la idea es probar con al menos un caso de prueba cada partición. Esta técnica se puede complementar con análisis de valores de borde (*Boundary Value Analysis*), la cual considera también probar los límites de las particiones. En cambio, las técnicas de caja blanca, sí utilizan la estructura interna del objeto de prueba para derivar casos de prueba. Las técnicas basadas en la experiencia utilizan la experiencia e intuición del diseñador de pruebas para decidir con qué entradas probar el objeto de prueba.

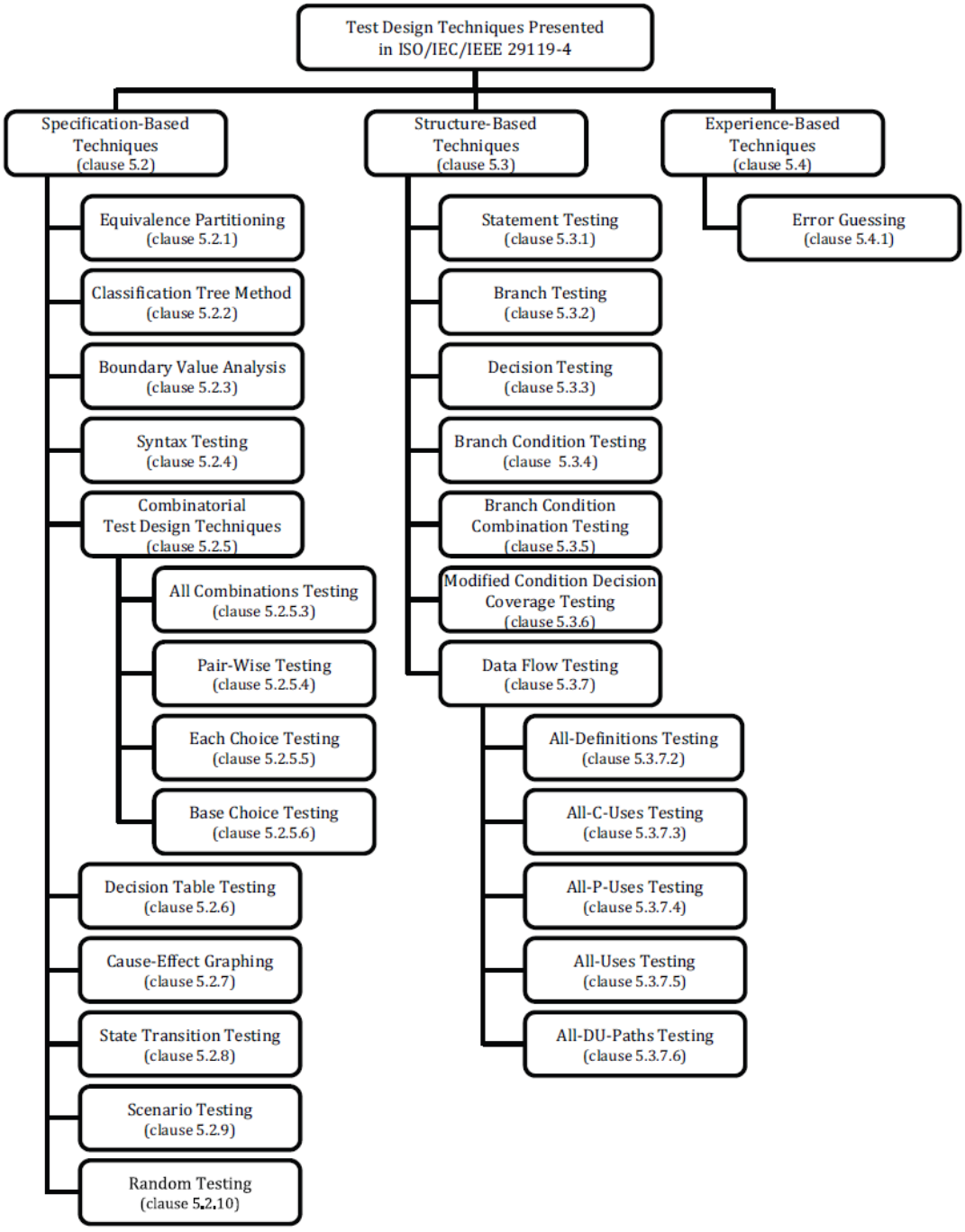


Figura 2-3: Clasificación de técnicas de diseño de pruebas presentes en ISO 29119-4. Figura tomada de (ISO, 2015).

Finalmente, pero no menos importante, la utilización de una terminología técnica del dominio por varios agentes sin la misma base conceptual puede llevar a desacuerdos, confusiones y problemas de ambigüedades dado que la semántica de un mismo concepto puede ser diferente para distintos agentes. Sin una terminología de referencia común,

concisa y consistente, es muy difícil lograr una comunicación efectiva entre las diferentes partes interesadas que participan del proceso de *testing*. El uso de una **base conceptual** robusta (como una ontología) en el que explícitamente se definen todos los conceptos necesarios de un dominio, sus propiedades y relaciones, y las restricciones existentes, promueve la facilidad de comprender y replicar el uso de la estrategia a utilizar. Por ello, es importante que los procesos y métodos de testing utilicen los conceptos de la base conceptual de testing en sus especificaciones, ya que asegura la uniformidad entre las tres capacidades y proporciona consistencia de resultados.

Anteriormente se dieron detalles de los tres pilares o capacidades que debería integrar una estrategia, los cuales son representados alegóricamente en la Figura 2-4. Si una estrategia de *testing* cuenta con estas tres capacidades y, además, las especificaciones de proceso y método utilizan la terminología de la base conceptual, entonces estamos hablando de una **estrategia integrada** de *testing*. Como beneficio de utilizar estrategias integradas se espera poder ejecutar proyectos de *testing* en los cuales se sepa claramente qué hacer (especificación de proceso), cómo hacerlo (especificación de métodos y herramientas), y que exista un entendimiento común de los conceptos claves (base conceptual). Notar que esta noción de estrategia integrada se puede extender a otros dominios y no solo aplica para las pruebas de software. Por ejemplo, en (Olsina & Becker, 2017) se ilustra una familia de estrategias integradas que ayudan a alcanzar propósitos de evaluación tales como comprender, monitorear, seleccionar una alternativa, entre otros.

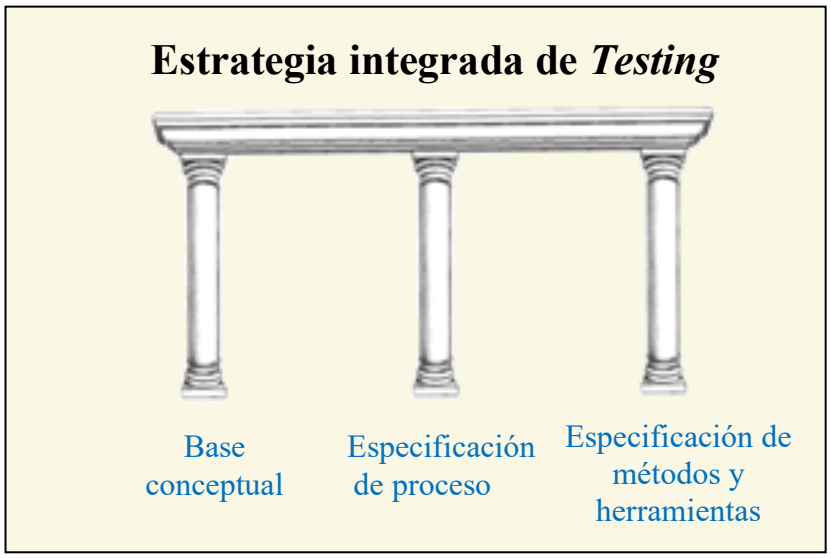


Figura 2-4: Alegoría de los tres pilares que deberían sostener una estrategia integrada de testing.

Actualmente existen estándares relevantes que documentan enfoques que integran las tres capacidades. Un claro ejemplo es ISO 29119, el cual está compuesto por 5 partes. Como se mencionó en el Capítulo 1, la parte 2 (ISO, 2013b) y parte 4 (ISO, 2015) documentan especificaciones de procesos y métodos de pruebas, respectivamente. Además, la parte 3 (ISO, 2013c) contiene información sobre artefactos de *testing*, la cual enriquece las especificaciones de procesos ya que estos artefactos son producidos/consumidos por las distintas actividades/tareas del proceso de *testing*. La parte 1 (ISO, 2013a) documenta la base conceptual estructurada como un glosario. Por último, la parte 5 (ISO, 2016) describe el enfoque llamado *keyword-driven testing*, el cual da soporte para la automatización de las pruebas.

Es importante mencionar que ISO 29119 se enfoca principalmente en pruebas dinámicas, es decir, pruebas que requieren la ejecución del objeto de prueba (ISO, 2013a).

No obstante, hace mención a las pruebas estáticas, pruebas en las cuales el objeto a ser probado es examinado/inspeccionado/revisado sin ejecutar su código. ISO 29119 referencia al estándar IEEE 1028-2008 (IEEE, 2008a) para conocer más detalles sobre pruebas estáticas. Este último documenta aspectos relacionados al proceso de pruebas estáticas para revisiones de gestión y técnicas, inspecciones, *walkthroughs* y auditorias. Entre los aspectos de proceso ilustrados se encuentran roles involucrados, artefactos de entrada y salida, criterios/condiciones de entrada y salida del proceso, y las actividades principales a ejecutar. También contiene un glosario de términos como base conceptual. La desventaja principal a la hora de comprender este estándar es que los aspectos de proceso fueron especificados usando solo texto plano y no un lenguaje de especificación de proceso más formal como lo son por ejemplo SPEM o BPMN (*Business Process Model and Notation*). Además, si bien considera roles, artefactos o productos de trabajo, y actividades, no siempre se menciona explícitamente qué rol está involucrada en cada actividad, y tampoco qué artefacto es consumido/producido en cada una de ellas.

Otro ejemplo de documentación que contiene especificaciones de procesos y métodos de *testing* es la brindada por el estándar *de facto* ISTQB (ISTQB, 2018, 2021a). Además, como se indicó anteriormente, cuenta con un glosario de términos como base conceptual (ISTQB, 2021b). A su vez, TMMi (TMMi Foundation, 2018) contiene un conjunto de áreas de procesos en donde se definen un conjunto de actividades/tareas a realizar (llamadas prácticas), menciona algunos productos de trabajo y también, al igual que todos los anteriores, utiliza como base conceptual un glosario de términos.

Además de los estándares mencionados anteriormente, existen en la literatura otros enfoques o estrategias de *testing*, aunque no todas pueden considerarse estrategias integradas. Por ejemplo, en (Traoré, 2003) se presenta un marco de trabajo (*framework*) que da soporte para probar programas orientados a objetos. El mismo cuenta con una herramienta (3er pilar en Figura 2-4) para la validación y verificación de especificaciones UML. A su vez, en (Traoré, 2003) se especifica un conjunto de pasos (2do pilar en Figura 2-4) a llevar a cabo para generar y ejecutar casos de prueba. Sin embargo, no hace mención a un soporte terminológico o base conceptual (1er pilar) que respalde semánticamente su enfoque.

En (Nguyen et al., 2008) se ilustra un *framework* de pruebas "orientado a metas" para producir casos de prueba usando modelos de metas basados en *Tropos* (Bresciani et al., 2004). Esta estrategia brinda una especificación de proceso que guía a sus usuarios en la generación de casos de pruebas durante el proceso de desarrollo de *Tropos*. La misma está basada en el modelo de proceso en V para el desarrollo de software. Además, los autores en (Nguyen et al., 2008) mencionan que su enfoque contiene una herramienta que da soporte para la generación y ejecución de casos de pruebas. No obstante, este *framework* no cuenta con una base conceptual (1er pilar en Figura 2-4) que lo enriquezca semánticamente.

En otros trabajos que presentan enfoques o estrategias de pruebas de software (Bhattacharjee et al., 2007; Chen & Jones, 2007; de Souza Doreste & Travassos, 2020; Lei et al., 2007; Meriem & Abdelaziz, 2019; Pei et al., 2017), también se puede observar que solo especifican el 2do y/o el 3er pilar de una estrategia integrada, sin considerar un soporte terminológico/semántico como un glosario, taxonomía u ontología. En otras palabras, estos trabajos solo documentan procesos/actividades/pasos a seguir y/o algoritmos/métodos/herramientas.

Además, haciendo uso de la técnica *forward snowballing*, el autor de esta tesis realizó una búsqueda en *Scopus* de estudios primarios que documenten estrategias de *testing* y

citen a alguno de los glosarios de *testing* de ISTQB, TMMi o ISO 29119-1. Entre los pocos artículos encontrados, en (Ramler et al., 2017) solo referenciaban al glosario de ISTQB en la Sección 2 de "*background*", pero sin hacer mención explícita si toda la semántica involucrada de los conceptos principales del enfoque estaba basada en dicho glosario.

Como se comentó en el Capítulo 1, para este trabajo de doctorado se llevó a cabo una RSL sobre ontologías de pruebas de software (Tebes et al., 2020). En dicho estudio secundario se seleccionaron un total de 12 ontologías y, de la misma manera, se realizó una búsqueda en *Scopus* de estudios primarios que documenten estrategias de *testing* y citen a alguna de las 12 ontologías seleccionadas, haciendo uso también de la técnica *forward snowballing*. Como resultado, no se encontró ningún estudio que documente una estrategia/enfoque/metodología o algún proceso/método de pruebas de software y que cite a alguna de estas 12 ontologías.

En conclusión, existen muchas estrategias o enfoques (*approaches*) de *testing*, aunque la gran mayoría no son integradas dado que se observa que no se le da mucha importancia al 1er pilar o capacidad, es decir, a la base conceptual que brinda el soporte terminológico/semántico común. La gran mayoría de los trabajos sobre pruebas de software solo se centran en especificar algoritmos/procedimientos, herramientas, una técnica particular o un proceso, generalmente como un conjunto de pasos y sin una especificación que considere vistas o perspectivas de modelado de proceso. Esto no quiere decir que estos trabajos no tengan valor, sino que se podrían beneficiar de las ventajas que conlleva enriquecer semánticamente las especificaciones de procesos y métodos con una base conceptual terminológica común. Además, para aquellos trabajos que consideraron utilizar una base conceptual, todos los encontrados hasta el presente utilizan glosarios y por lo tanto no aprovechan la riqueza de estructuración semántica de las ontologías.

2.4 ¿Por qué es Importante una Ontología como Marco Conceptual de una Estrategia Integrada?

Desde hace muchos años que las personas comenzaron a generar, representar y almacenar conocimiento. Esto produjo la necesidad de adoptar esquemas que permitan la organización y accesibilidad del conocimiento. Algunos esquemas que se utilizan para organizar el conocimiento son glosarios, taxonomías, ontologías, entre otros. Por ejemplo, las taxonomías son utilizadas desde la mitad de la década de los 90, y se utilizan en Internet para la organización de contenidos. A su vez, las ontologías, surgidas a fines de la década del 90, permiten tanto la organización del conocimiento como su reutilización e inferencia de nuevo conocimiento y facilitan que la información se pueda compartir. Estos esquemas dan soporte para la creación, gestión y visualización de modelos, los cuales muestran una perspectiva explícita de los conceptos que forman determinado dominio y de la estructura semántica subyacente. Los esquemas tienen diferente capacidad sintáctica y semántica dependiendo de sus objetivos, pero todos ellos contienen un vocabulario para definir los conceptos que involucra y las relaciones entre estos conceptos. A su vez, cada forma de representación del conocimiento emplea diferentes métodos, que van desde el uso de solo términos y sus definiciones en los glosarios, hasta el uso de axiomas para realizar inferencias en las ontologías. Esta forma de representar el conocimiento determina la complejidad y riqueza estructural de cada esquema. En orden ascendente en complejidad y riqueza estructural tenemos: glosarios, taxonomías y ontologías.

Un glosario es un conjunto ordenado de términos con una definición concisa y clara (Schneider, 2009). Los glosarios son mecanismos simples para recolectar términos pertenecientes a una misma disciplina o campo de estudio. Cada entrada en un glosario consiste de un término y un texto donde se lo define o explica, así como también se pueden encontrar acrónimos y sinónimos de los términos. Las definiciones son usualmente breves y están escritas en lenguaje natural, sin hacer uso de lenguajes formales.

Por su parte, una taxonomía es una colección de términos de un vocabulario organizados en una estructura jerárquica (Fayen, 2007). Los términos dentro de una taxonomía se relacionan a través de la relación padre-hijo. Como una taxonomía es el resultado de clasificar conforme a características en común, existen diferentes tipos de relaciones "padre-hijo", como por ejemplo todo-parte, género-especie, tipo-subtipo, etc. Como beneficio, la clasificación de los términos ayuda a mejorar su comprensión.

Otra manera de estructurar el conocimiento es usando ontologías. Desde el punto de vista de la inteligencia artificial, Gruber (Gruber, 1995) define una ontología como: "una especificación explícita de una conceptualización". En (Borst, 1997) esta definición fue ligeramente modificada: "Las ontologías se definen como la especificación formal de una conceptualización compartida". Además, los autores en (Studer et al., 1998) dieron más detalles de estas definiciones agregando que: **conceptualización** se refiere a un modelo abstracto de algún fenómeno en el mundo, proveniente de haber identificado los conceptos relevantes de dicho fenómeno; **explícita** significa que los conceptos usados y las restricciones para su uso se definen explícitamente; **formal** hace mención al hecho de que la ontología debería ser legible o interpretable por una computadora; y **compartida** refleja la noción de que una ontología captura conocimiento consensuado, es decir, no es conocimiento privado de un individuo, sino aceptado por un grupo o comunidad.

Una ontología está compuesta de un conjunto no vacío de conceptos o términos principales, los cuales representan las entidades relevantes del dominio a modelar. A su vez, estos conceptos principales pueden contener atributos o propiedades, así como también se pueden encontrar relaciones entre ellos. También pueden existir un conjunto de axiomas que establecen restricciones que involucran a los conceptos de la ontología. Todo esto se especifica explícitamente con el objetivo de limitar las posibles interpretaciones de lo que se declara. En otras palabras, con una ontología se busca evitar ambigüedades en la interpretación de las definiciones, capturando y estructurando conocimiento que puede ser compartido y reusado (Swartout & Tate, 1999).

Es importante que una ontología cumpla con un conjunto de características deseables. Entre ellas, las definiciones de los términos involucrados en la ontología deberían ser lo más objetivas posibles, y siempre que sea posible, deben estar definidos por condiciones necesarias y suficientes (no solamente por condiciones necesarias) (Gruber, 1995). A su vez, una ontología debería tener diversificación de jerarquías, es decir, es conveniente usar tantos criterios de clasificación como sea posible, de manera de representar más conocimiento (Benjamins & Gomez-Perez, 2000). De esta forma se favorece la adición de nuevos términos porque se los puede definir partiendo desde los conceptos preexistentes y criterios de clasificación. También, se debería minimizar la distancia semántica entre conceptos hermanos, esto es, conceptos similares se deben agrupar y representar como subclases de una clase y deberían ser definidos usando las mismas primitivas. Por último, es conveniente el uso de la estandarización de nombres definiendo y respetando reglas para su formación siempre que sea posible, siendo esto una característica deseable para ayudar en el mantenimiento y consistencia de una ontología.

En cierta forma, todas estas características favorecen la comunicación efectiva del significado propuesto de los términos definidos en la ontología.

También es importante mencionar que la inferencia automática es una característica importante en una ontología, ya que el conocimiento capturado en el modelo de dominio es formalizado e implementado de modo que agentes de software pueden entenderlo y actuar en consecuencia. Esta característica es aprovechada en el campo de la Web Semántica para la recuperación inteligente de información, es decir, la obtención de recursos que están relacionados conceptualmente sin que exista una relación explícita entre ambos.

Hasta este punto se presentaron tres representaciones de conocimiento con distintos niveles de semántica, estructuración y características. A modo de resumen, un glosario es un conjunto de términos con una definición breve sobre un tema particular. Por su parte, una taxonomía es esencialmente una estructura de árbol jerárquica que modela un dominio a partir de conceptos que van de lo más abstracto a lo más específico. Finalmente, las ontologías son las representaciones más ricas y formales, ya que definen el significado de los conceptos modelando condiciones que restringen el número de posibles interpretaciones.

La ontología supera al resto de las formas de organización en su capacidad de representar e interconectar mundos a través de descripciones, relaciones, atributos, restricciones y valores para inferir. Es por ello que, en este trabajo, se considera que una estrategia integrada debería tener una ontología como base conceptual y no meramente un glosario y/o taxonomía. Además, las especificaciones de procesos y métodos deberían utilizar los conceptos que involucra esta base conceptual ontológica.

Figura 2-5: Alegoría de los cimientos como base conceptual ontológica que apoya dos pilares, todos ellos sosteniendo una estrategia integrada de *testing*.

En efecto, la Figura 2-5 representa los pilares de una estrategia integrada considerando una ontología como los cimientos o la base conceptual principal la cual apoya las especificaciones de procesos y métodos. Notar que la Figura 2-5 representa mejor la idea que se quiere trasmitir en este trabajo de una estrategia integrada de *testing* en comparación a la Figura 2-4.

Otros trabajos (Bürger & Simperl, 2008; Daraio et al., 2016; Merrell et al., 2021; Milton et al., 2010; Oberle, 2014) también ilustran algunos beneficios de utilizar

ontologías. Entre ellos, se mencionan la interoperabilidad permitiendo el enlace de diferentes sistemas con datos heterogéneos, la estandarización de un vocabulario que mejora el entendimiento dentro de un dominio particular y facilita la comunicación entre agentes (tanto humanos como de software), y el soporte al chequeo de la calidad de los datos. También señalan la utilidad para la reutilización y organización del conocimiento.

Al momento de definir una base conceptual se debería tener en cuenta aquella representación que permita una mayor expresividad sin dejar lugar a las ambigüedades. También es deseable revisar y analizar los estándares relacionados existentes ya que es importante, de ser posible, reutilizar aquellos términos y definiciones presentes en los mismos. Los sinónimos también deberían tenerse en cuenta y ser reflejados en la base conceptual. Estas características son deseables ya que hacen que la base conceptual sea más fácil de asimilar y su aprendizaje sea más rápido. Otra característica esperada es que la base conceptual sea lo más completa posible, es decir, que cubra la mayoría de los términos presentes en el dominio.

En (d'Aquin & Gangemi, 2011) se presenta un conjunto de características o "dimensiones de evaluación" deseables que debería tener una ontología para ser considerada de calidad. Entre ellas, tenemos:

- Dimensión relacionada a la estructura de la ontología: reutilizar ontologías fundacionales, ser formalmente rigurosa, implementar relaciones no taxonómicas, ser modular o estar embebida en un *framework* modular.
- Dimensión relacionada a la cobertura conceptual: reutilizar otras bases conceptuales, tener buena cobertura del dominio, implementar un estándar internacional.
- Dimensión relacionada a tareas conceptuales: estar orientada a una tarea explícita, basarse en preguntas de competencia.
- Dimensión relacionada a la sostenibilidad social: ser el resultado de una evolución (muchas revisiones), tener un amplio uso o aceptación, tener impacto comercial, ser recomendado por la industria, implementar conocimiento científico.
- Dimensión relacionada a la sostenibilidad pragmática: tener aplicaciones construidas sobre ella, estar diseñada para responder consultas de manera eficiente, mantener la expresividad original de los datos y mejorarlos o enriquecerlos, estar bien documentada.

Recordemos que el objetivo principal que se persigue en este trabajo con la utilización de una ontología como parte integrada de una estrategia de testing es el enriquecimiento semántico de las especificaciones de procesos y métodos. Además, otro propósito importante es que la terminología ayude en el entendimiento general de los procesos y métodos que integran la estrategia y a su vez facilite la comunicación entre los agentes que la utilizan. En (Asman & Srikanth, 2015) se describen diferentes tipos de ontologías con diferentes propósitos adicionales a los anteriores. Por ejemplo, se mencionan ontologías las cuales contienen conceptos muy generales que son independientes del dominio y tienen como objetivos facilitar la integración semántica de ontologías de dominio y guiar en el desarrollo de nuevas ontologías. Ejemplos de ontologías *top-level* son ThingFO (Thing Foundational Ontology (Olsina, 2021)) y UFO (Unified Foundational Ontology (Guizzardi, 2005)). También se mencionan ontologías *top-level* de dominio, cuyo propósito es integrar ontologías *top-level* con ontologías de dominio de bajo nivel.

CAPÍTULO 3: ESTADO DEL ARTE DE ONTOLOGÍAS DE PRUEBAS DE SOFTWARE

3.1 Introducción

En el capítulo anterior, específicamente en la Sección 2.4, se fundamentó la importancia de contar con una base conceptual ontológica que de soporte a las especificaciones de procesos y métodos que integran una estrategia. Por lo tanto, dado que dicha base conceptual conforma los cimientos de una estrategia integrada de *testing* (recordar la Figura 2-5), se llevó a cabo una RSL (Revisión Sistemática de Literatura) sobre ontologías de pruebas de software (Tebes et al., 2020) con el objetivo de encontrar estudios primarios que documenten ontologías de *testing*. Como resultado, 12 estudios primarios fueron seleccionados y evaluados con el fin de decidir si alguna de estas ontologías era lo suficientemente adecuada para adoptarla/adaptarla e integrarla como parte de una estrategia de *testing* a desarrollar. Como conclusión, se notó que ninguna de las ontologías existentes para el dominio de las pruebas de software era adecuada para el objetivo perseguido. Todo este estudio se documenta en la Sección 3.3. Recordar que esta RSL realizada está asociada al objetivo específico 1 y contribución 2 (OE_1 y C_2, respectivamente, en Tabla 1-1).

Además, al momento de ejecutar el estudio, inicialmente se tuvo que comprender el proceso para llevar a cabo RSLs para poder realizar esta revisión en particular sobre ontologías de *testing*. Durante este proceso de aprendizaje se observó que el proceso de RSLs estaba débilmente especificado. En consecuencia, y como un trabajo adicional relacionado, se realizó la mejora en la especificación del proceso para llevar a cabo tanto revisiones sistemáticas de literatura como mapeos sistemáticos (Olsina et al., 2019), considerando vistas de modelado de proceso. Este proceso se encuentra detallado en la Sección 3.2. También es importante recordar que este trabajo realizado se encuentra relacionado al OE_1 y fue considerado como la contribución 1 (C_1 en Tabla 1-1) de esta tesis doctoral.

3.2 Proceso para Revisiones Sistemáticas de Literatura y Mapeos Sistemáticos

3.2.1 Introducción

Una RSL tiene como objetivo proporcionar una evidencia exhaustiva de la literatura relevante para un conjunto de preguntas de investigación. Inicialmente, las RSLs se realizaron en el ámbito de la medicina clínica (Sackett et al., 1996). Desde que Kitchenham publicó un informe técnico en el 2004 sobre RSLs (B. Kitchenham, 2004b), su uso en las diferentes comunidades científicas de la Ingeniería del Software se ha vuelto cada vez más frecuente para recopilar evidencia principalmente de estudios primarios y, en menor medida, de estudios secundarios. Se denomina estudio secundario al resultado que se obtiene de aplicar el proceso de RSL sobre estudios primarios, mientras que al aplicarlo sobre estudios secundarios se obtiene un estudio terciario como resultado. Solo por citar algunos ejemplos, en (Sepúlveda et al., 2016; Tahir et al., 2016; Torrecilla-Salinas et al., 2016) se documentan estudios secundarios sobre diferentes temas de la Ingeniería del Software, mientras que en (Garousi & Mäntylä, 2016; B. Kitchenham et al., 2010) se reportan estudios terciarios.

Generalmente los investigadores han utilizado los procedimientos y guías propuestas en (B. Kitchenham, 2004b), que fueron revisados por primera vez en (Biolchini et al., 2005) y luego actualizadas por Kitchenham y otros autores en el 2007 (Brereton et al., 2007; B. Kitchenham & Charters, 2007). Más recientemente, Kitchenham and Brereton llevaron a cabo una RSL (B. Kitchenham & Brereton, 2013) en la cual evaluaron y sintetizaron estudios publicados por investigadores de la Ingeniería del Software (incluyendo diferentes tipos de estudios, no solo primarios) que discuten sobre sus experiencias al realizar RSLs y sus propuestas para mejorar el proceso de una revisión sistemática.

A pesar de que las RSLs se han convertido en una metodología establecida en las investigaciones relacionadas a la Ingeniería del Software y existen guías que ayudan a los investigadores a realizarlas, el proceso de RSL aún no estaba especificado de manera completa y rigurosa. La Figura 3-1 muestra la especificación del proceso ilustrada en (Brereton et al., 2007), que fue totalmente adoptada o ligeramente adaptada por el resto de la comunidad de la Ingeniería del Software hasta el momento actual. Esta especificación de proceso muestra qué hacer a través de sus fases y pasos y en qué orden, o en otras palabras, a través de sus procesos, actividades y tareas (recordar que esta tesis de doctorado sigue la terminología relacionada a procesos de trabajo brindada por ProcessCO (Becker et al., 2022)).

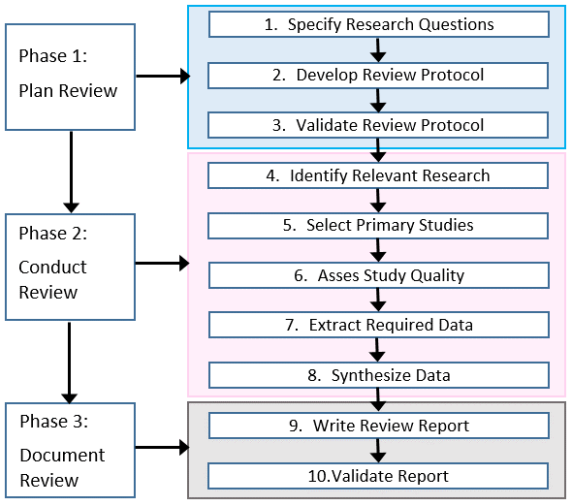


Figura 3-1: Proceso de RSL propuesto por Kitchenham. Figura ligeramente adaptada de (Brereton et al., 2007).

Sin embargo, el proceso de la Figura 3-1 puede mejorarse si tenemos en cuenta los principios de modelado de procesos propuestos por Curtis *et al.* (Curtis et al., 1992) (utilizados por ejemplo en (Becker et al., 2012)). Curtis *et al.* describen cuatro perspectivas/vistas para modelar un proceso llamadas funcional, de comportamiento, organizacional, e informacional, las cuales fueron descriptas en la Sección 2.3. Notar que, una especificación completa de proceso que considere diferentes perspectivas contribuye a una identificación más clara de qué tarea/actividades deben realizarse, en qué orden, por quién y cuáles son los artefactos consumidos y producidos, así como también conocer su estructura interna.

Algunos beneficios de usar el modelado de procesos para fortalecer las especificaciones del proceso en general, y para fortalecer el proceso de RSL en particular, son:

- **Facilitar la comprensión y la comunicación**, lo cual implica que el modelo de proceso (con la riqueza que brindan las representaciones gráficas) debe ser comprensible para la comunidad interesada.
- **Dar soporte a la mejora del proceso**, ya que se identifican todas las perspectivas fundamentales del modelo de proceso, lo que beneficia la reutilización y la evaluación del impacto frente a potenciales cambios en el mismo.
- **Dar soporte a la gestión del proceso**, es decir, dar soporte a las actividades de planificación, programación, y seguimiento y control.
- **Permitir la automatización del proceso**, lo que puede ayudar a proporcionar herramientas de soporte y mejorar el rendimiento.
- **Favorecer la verificación y validación del proceso**, fomentando así la consistencia, repetibilidad y auditabilidad en los proyectos.

Por lo tanto, el principal objetivo perseguido fue mejorar las especificaciones de proceso de RSL existentes, considerando los principios y beneficios del modelado de proceso descriptos anteriormente. Para lograr este objetivo se utilizaron las cuatro perspectivas propuestas por Curtis *et al.*, a saber, funcional, de comportamiento, organizacional e informacional. Además, se especificaron actividades relacionadas con la prueba piloto de una RSL, que en general son ignoradas en otras especificaciones de procesos de RSLs. Como resultado, las RSLs pueden ser más sistemáticas, repetibles y auditables para investigadores y profesionales. Es importante mencionar que, con respecto a los beneficios citados del modelado de procesos, las especificaciones de procesos para las RSLs aquí propuestas tienen como objetivo principal facilitar la comprensión y la comunicación, así como brindar apoyo a la mejora y gestión de procesos. Sin embargo, una discusión exhaustiva y una ilustración detallada del modelado de procesos para admitir completamente la automatización del proceso de RSL quedó fuera del alcance.

El resto de la Sección 3.2 está estructurado de la siguiente manera: la Sección 3.2.2 comenta brevemente la motivación y el trabajo relacionado. La Sección 3.2.3 especifica el proceso propuesto para llevar a cabo RSLs (o mapeos sistemáticos) considerando diferentes vistas de modelado de procesos. Notar que, más detalles sobre este trabajo en particular el cual especifica un proceso para RSLs se pueden encontrar en (Olsina et al., 2019).

3.2.2 Motivación y Trabajo Relacionado

Una motivación para modelar el proceso de RSL surgió al enfrentar dificultades para realizar un estudio piloto de RSL sobre ontologías de pruebas de software (Tebes, Peppino, Dameno, et al., 2018). El objetivo general de esta prueba piloto fue poder afinar y mejorar aspectos del diseño del protocolo como las preguntas de investigación, el protocolo de búsqueda, los criterios de selección y calidad además de los formularios de extracción de datos. Se considera que, en estudios de gran escala como una RSL, una prueba piloto de pequeña escala suele preceder al estudio principal para analizar la validez del diseño. Por lo tanto, es muy útil para los investigadores realizar esta prueba para verificar si los aspectos del diseño son adecuados.

Analizando varios trabajos sobre RSLs, se observan al menos tres problemas principales. Primero, las actividades relacionadas con la prueba piloto en general se ignoran o no se especifican explícitamente. En segundo lugar, algunos aspectos de los procesos de RSLs existentes están débilmente especificados desde el punto de vista de las perspectivas de modelado de procesos. En tercer lugar, en general falta una clara

separación entre qué hacer (procesos) y cómo hacerlo (métodos). A continuación, se mencionan trabajos relacionados con especificaciones del proceso de RSLs donde se detectaron estos problemas.

La primera representación gráfica del proceso de RSL propuesta por Kitchenham (Brereton et al., 2007) se esbozó en 2007, teniendo en cuenta trabajos previos de los mismos autores (B. Kitchenham, 2004b) y otras contribuciones como (Biolchini et al., 2005). Este proceso fue totalmente adoptado o ligeramente adaptado por el resto de la comunidad de la Ingeniería del Software hasta el presente. La mayoría de los trabajos dividen el proceso en tres fases o etapas: Planificar la Revisión, Conducir la Revisión y Documentar la Revisión. Mientras que a nivel de fase/etapa generalmente se conservan las mismas tres actividades principales (por ejemplo, en (Sepúlveda et al., 2016; Tahir et al., 2016; Torrecilla-Salinas et al., 2016), por citar solo algunos trabajos), a nivel de paso (subactividades y tareas) difieren en cierta medida unos de otros. Por ejemplo, en (Tahir et al., 2016) se modelan tres pasos para la fase 1: 1) Necesidad de la RSL; 2) Formular preguntas de investigación; y 3) Revisar el protocolo. Tenga en cuenta que estos pasos difieren de los que se muestran en la fase 1 de la Figura 3-1. Además, en (Sepúlveda et al., 2016) se presentan cinco pasos para la fase 1: 1) Objetivo y necesidad de la RSL; 2) Definir las preguntas de investigación; 3) Definir la cadena de búsqueda; 4) Definir los criterios de inclusión y exclusión; y 5) Validar el Protocolo. La misma falta de consenso en nombrar e incluir pasos se observa en los trabajos mencionados anteriormente para la fase 2. Además, estos trabajos solo modelan el proceso desde la perspectiva de comportamiento, por lo que las entradas, salidas y roles no se especifican en estos modelos de proceso.

Aunque la actividad de realizar una prueba piloto en los procesos de RSLs no suele ser incluida, en (Sepúlveda et al., 2016) se considera el paso "Selección y extracción piloto" en la fase 2. Sin embargo, el resultado de este paso no retroalimenta a la fase 1, el cual puede ayudar a mejorar aspectos del diseño de la RSL como se modela en la especificación de proceso propuesta para este trabajo (ver Figura 3-2).

En la Tabla 3-1 se resumen las características analizadas de algunos procesos de RSLs. Por ejemplo, (Biolchini et al., 2005) utiliza UML para modelar el proceso de RSL, y además considera las vistas de comportamiento e informacional. Además, la especificación de la perspectiva informacional está representada por una estructura de desglose de productos de trabajo basada en texto (*). Notar que no considera especificar actividades relacionadas a una prueba piloto. Por su parte, en (Irshad et al., 2018) no se encuentra ninguna representación gráfica del modelo de proceso para una RSL (**). Es importante mencionar que los autores de (Irshad et al., 2018) adoptaron el proceso de (B. Kitchenham & Charters, 2007).

Por otro lado, es importante señalar que, mientras las RSLs se enfocan en obtener y resumir la evidencia empírica de estudios primarios o secundarios, los estudios de mapeos sistemáticos se utilizan para estructurar/categorizar un área de investigación. Según (Marshall & Brereton, 2013), un mapeo sistemático es una forma más "general" de RSL, que frecuentemente se usa para proporcionar una visión global de un área de investigación al evaluar la cantidad de evidencia que existe sobre un tema en particular.

En (Petersen et al., 2015), los autores realizaron un estudio de mapeo sistemático para identificar cómo se lleva a cabo su proceso y para identificar potenciales mejoras en su realización. Aunque existen diferencias entre mapeos y RSLs en cuanto al objetivo de las preguntas de investigación, el proceso de búsqueda, los requisitos de la estrategia de búsqueda, la evaluación de la calidad y los resultados (B. A. Kitchenham et al., 2010), el

proceso seguido en (Petersen et al., 2015) es el mismo que el utilizado para las RSLs. Por lo tanto, se considera que el proceso de RSL propuesto como contribución de esta tesis de doctorado puede utilizarse también para llevar a cabo mapeos sistemáticos. Además, como hipótesis subyacente, la brecha existente en la falta de estandarización del proceso de RSL/mapeo sistemático utilizado actualmente por las comunidades científicas puede minimizarse si consideráramos más apropiadamente los principios y beneficios del modelado de procesos enumerados en la subsección anterior (3.2.1).

Tabla 3-1: Características de algunos artículos analizados que especifican el proceso de RSL. Tabla tomada y ligeramente adaptada de (Olsina et al., 2019).

	Características de las especificaciones de proceso					
Artículo (referencia)	Vista Funcional	Vista de Comportamiento	Vista Organizacional	Vista Informacional	Notación utilizada para graficar el modelo de proceso	¿Considera actividades de prueba piloto?
(Biolchini et al., 2005)		✓		✓ (*)	UML	
(Brereton et al., 2007)		✓			Informal (uso de flechas y cajas)	
(Garousi & Mäntylä, 2016)		✓			Informal (uso de leyendas para los elementos utilizados)	
(Irshad et al., 2018) (**)						
(Sepúlveda et al., 2016)		✓			UML	✓
(Tahir et al., 2016)		✓			Informal (uso de flechas y cajas)	
(Olsina et al., 2019)	✓	✓	✓	✓	SPEM	✓
(Torrecilla-Salinas et al., 2016)		✓			Informal (uso de flechas, círculos y cajas)	

3.2.3 Especificación del Proceso Propuesto para RSLs

La Figura 3-2 ilustra el proceso para RSLs propuesto, modelado con SPEM y considerando únicamente la perspectiva de comportamiento. Existen varios lenguajes de modelado de procesos como BPMN, SPEM y Diagramas de Actividad UML (OMG, 2017), que son los más populares en la academia y la industria. Sus notaciones son muy similares considerando diferentes características deseables como la expresividad (es decir, la cantidad de patrones de flujo de trabajo soportados), la comprensibilidad, entre otras (Portela et al., 2012; Russell et al., 2006; White & others, 2004). SPEM, UML y BPMN son lenguajes de modelado adecuados que se pueden utilizar para modelar las vistas funcional, de comportamiento y organizacional. Sin embargo, para la perspectiva informacional, BPMN no es un lenguaje adecuado. SPEM permite utilizar tanto el Diagrama de Procesos de Negocio BPMN como también el Diagrama de Actividades UML, entre otros diagramas como el Diagrama de Clases UML para especificar todas las perspectivas/vistas de proceso.

Como se ve en la Figura 3-2, el proceso propuesto al igual que el proceso original de Brereton *et al.* (Brereton et al., 2007) tiene tres actividades principales, a saber: Diseñar la Revisión (A1), Implementar la Revisión (A2), y Analizar y Documentar la Revisión (A3). A su vez, estas actividades contienen subactividades y tareas. Notar que, para la subactividad Diseñar el Protocolo de Búsqueda (*Design Search Protocol*) se muestran las tareas incluidas mientras que no para el resto de las subactividades de A1. Esto se realiza

intencionalmente para comunicar que las subactividades tienen tareas y al mismo tiempo no dar todos los detalles con el fin de preservar la legibilidad del diagrama.

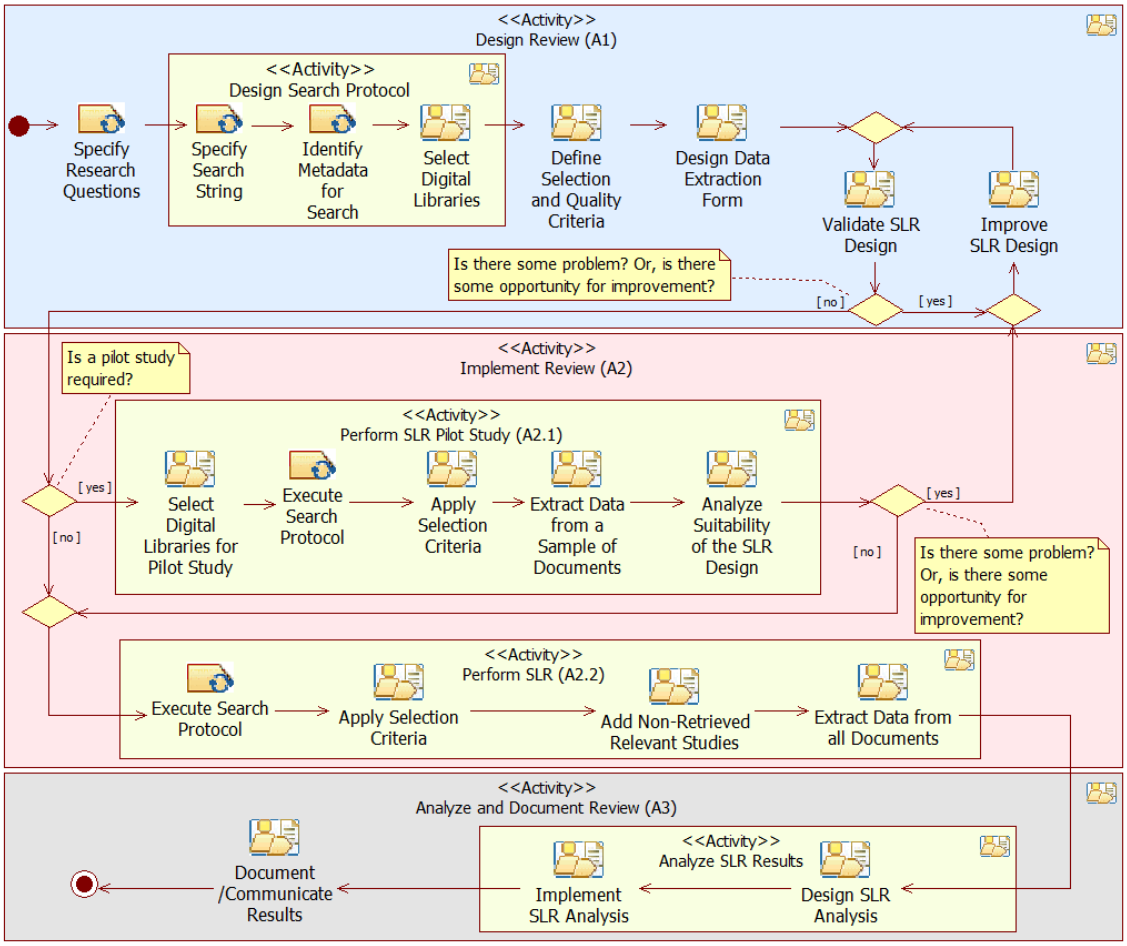


Figura 3-2: Vista de comportamiento del proceso de RSL propuesto. Figura tomada de (Olsina et al., 2019).

Como el lector puede notar, la especificación de proceso aquí propuesta tiene más detalles que otros modelos utilizados actualmente para llevar a cabo RSLs, desde el punto de vista de la perspectiva de comportamiento. Por ejemplo, presentamos nodos de decisión (ver los rombos en la Figura 3-2) para representar iteraciones (ver la iteración en A1 entre *Validate SLR Design* e *Improve SLR Design*) y para indicar que algunas actividades/tareas podrían no realizarse (por ejemplo, la subactividad A2.1 "Realizar el Estudio Piloto de RSL" es opcional). Consecuentemente, este proceso muestra explícitamente a investigadores y profesionales que el proceso de RSL no es totalmente secuencial.

Vale la pena mencionar que la Figura 3-2 muestra un flujo recomendado para el proceso de RSLs. En otras palabras, representa como debería ser "idealmente" el proceso para RSLs aunque a veces, en la realidad, no se ejecute exactamente igual. Se tiene consciencia de que, en la instanciación de un proceso, puede haber algunos puntos de variación, como la paralelización de algunas tareas.

Además, con el fin de enriquecer la especificación del proceso, se considera en la Figura 3-3 la vista funcional junto a la perspectiva de comportamiento. Por lo tanto, a lo largo de todo el proceso, podemos ver el flujo de actividades y los productos de trabajo consumidos y producidos en cada actividad/tarea. La vista funcional es muy importante

ya que indica qué documentos se necesitan para realizar una tarea y qué documentos se deben producir, siendo útil también para fines de verificación. Desafortunadamente, la perspectiva funcional generalmente no se considera en otras propuestas del proceso para RSLs, tal como sintetiza la Tabla 3-1.

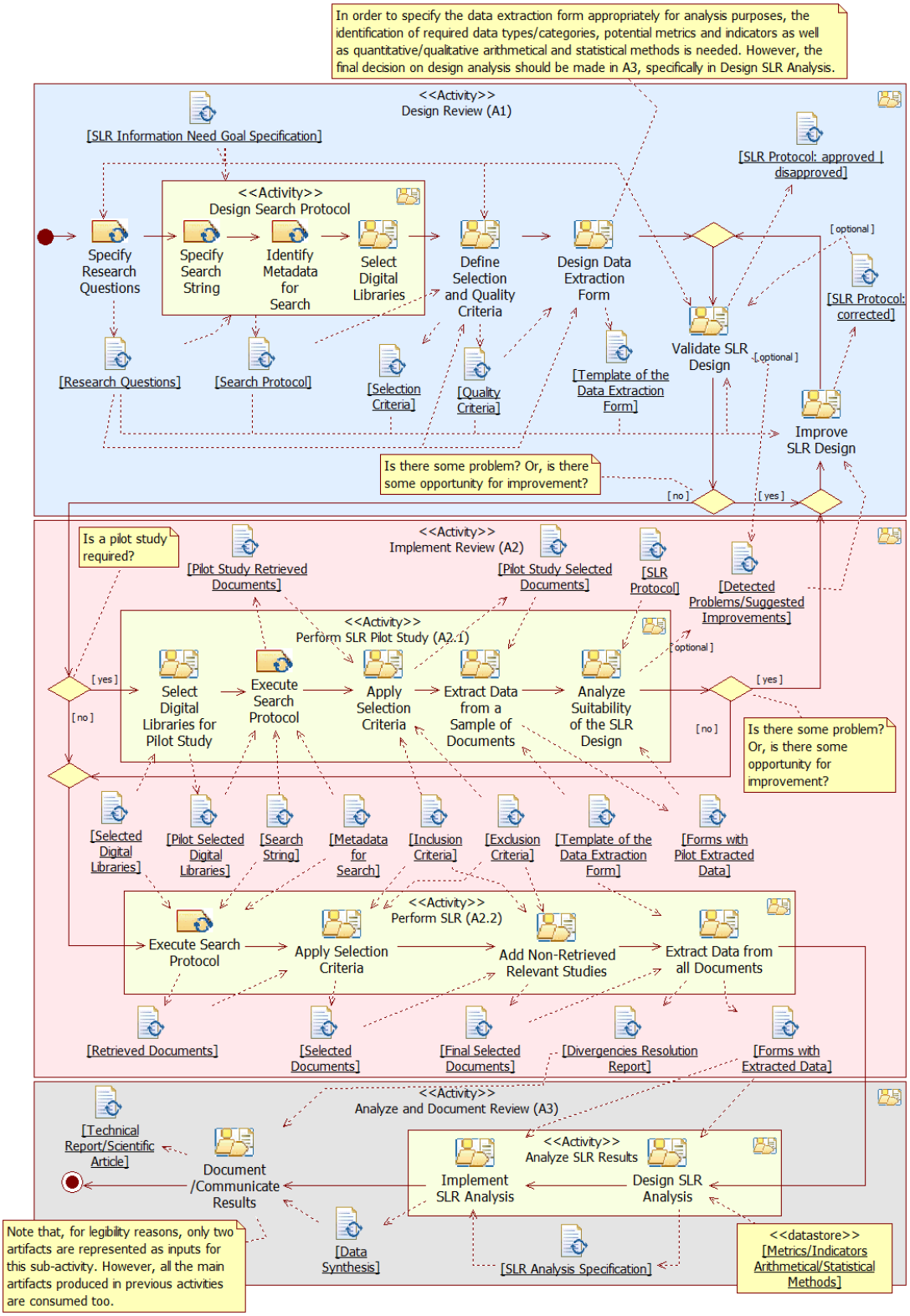


Figura 3-3: Vista funcionales y de comportamiento del proceso de RSL propuesto. Figura tomada de (Olsina et al., 2019).

En las siguientes subsecciones se describen las tres actividades principales considerando sus subactividades y tareas, secuencias, entradas y salidas, roles, teniendo en cuenta las vistas funcional, de comportamiento y organizacional. Adicionalmente, para enriquecer las especificaciones del proceso, en algunos casos se utilizará la perspectiva informacional. También es importante mencionar que el lector puede encontrar la vista organizacional con los diferentes roles involucrados en un proceso de RSL en el Apéndice A.

3.2.3.1 Diseñar la Revisión (A1)

El objetivo principal de la actividad Diseñar la Revisión (A1) es producir el protocolo de la RSL. Para lograr esto, las tareas y actividades representadas en el cuadro azul claro en la Figura 3-3 deben realizarse siguiendo el flujo representado y utilizando los artefactos de entrada y salida.

Como se muestra en la Figura 3-3, la primera tarea es Especificar las Preguntas de Investigación que consume el artefacto "Especificación de Meta de Necesidad de Información de RSL". Este artefacto incluye el propósito de la meta y la declaración establecida por los investigadores, y sirve de guía para el diseño de la revisión. Luego, considerando las "Preguntas de Investigación", se realiza la actividad Diseñar el Protocolo de Búsqueda. Esto incluye las tareas Especificar la Cadena de Búsqueda e Identificar los Metadatos para la Búsqueda, así como también la subactividad Seleccionar las Bibliotecas Digitales. A su vez, esta última incluye las tareas Definir los Criterios de Selección de Bibliotecas Digitales e Identificar Bibliotecas Digitales, como se observa en la Figura 3-4. Ejemplos de criterios de selección de bibliotecas digitales pueden ser el idioma utilizado y el dominio de la biblioteca, entre otros. La selección de las bibliotecas digitales puede determinar el alcance y la validez de las conclusiones de los revisores. Como resultado de la actividad Diseñar el Protocolo de Búsqueda, se obtiene el "Protocolo de Búsqueda" que incluye una cadena de búsqueda compuesta por términos y operadores lógicos, los metadatos sobre los que se aplicará la búsqueda (por ejemplo, título y resumen) y, las bibliotecas digitales seleccionadas (por ejemplo, IEEE, ACM, Springer Link, Google Scholar, entre otras).

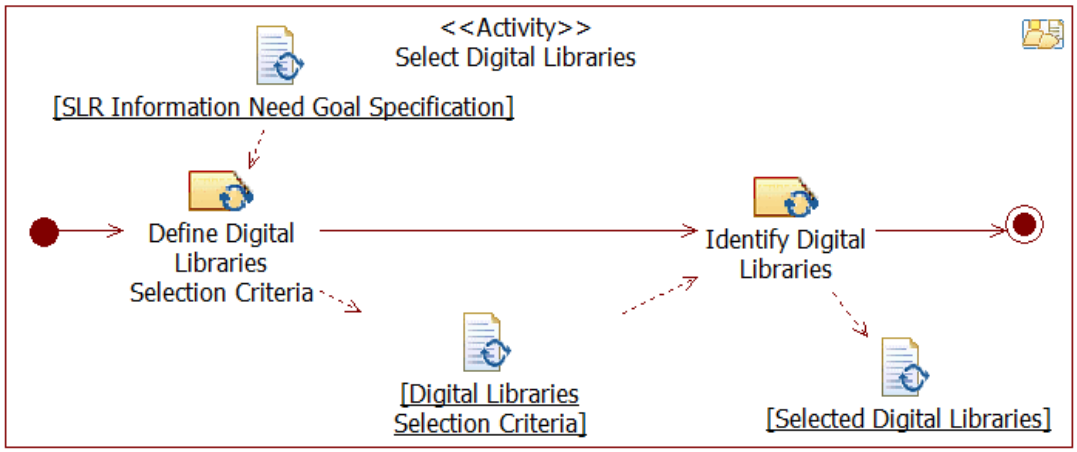


Figura 3-4: Vistas funcional y de comportamiento para la subactividad Seleccionar las Bibliotecas Digitales. Figura tomada de (Olsina et al., 2019).

A partir de los artefactos "Protocolo de Búsqueda" y "Preguntas de Investigación" es posible ejecutar la subactividad Definir Criterios de Selección y Calidad. Esta produce los artefactos "Criterios de Selección" y "Criterios de Calidad". Los "Criterios de

Selección" documentan el conjunto de criterios de inclusión y exclusión (ver Figura 3-5), es decir, las pautas que determinarán si un artículo será considerado en la revisión o no. Los revisores deben preguntarse: ¿Es el estudio relevante para el propósito de la revisión? ¿El estudio es aceptable para ser revisado? Para responder a estas preguntas, los revisores formulan criterios de inclusión y exclusión. Cada revisión sistemática tiene su propio objetivo y preguntas de investigación, por lo que sus criterios de inclusión y exclusión suelen ser únicos (excepto en estudios replicados).

Los "Criterios de Calidad" documentan características que permiten evaluar la calidad de los estudios recuperados en A2, así como identificar aspectos relevantes o deseables para los investigadores. En algunos casos, los criterios de calidad se utilizan como criterios de inclusión/exclusión (o para construirlos) ya que a veces es importante seleccionar estudios de alta calidad para obtener resultados y conclusiones confiables (B. Kitchenham, 2004b; B. Kitchenham & Charters, 2007). Utilizar "Criterios de Calidad" como "Criterios de Selección" es una decisión crítica ya que si los criterios de inclusión son demasiado amplios se pueden incluir estudios de baja calidad, lo que reduce la confianza en el resultado final. En cambio, si los criterios son demasiado estrictos, los resultados se basan en menos estudios y es posible que la evidencia obtenida no sea generalizable (Lam & Kennedy, 2005).

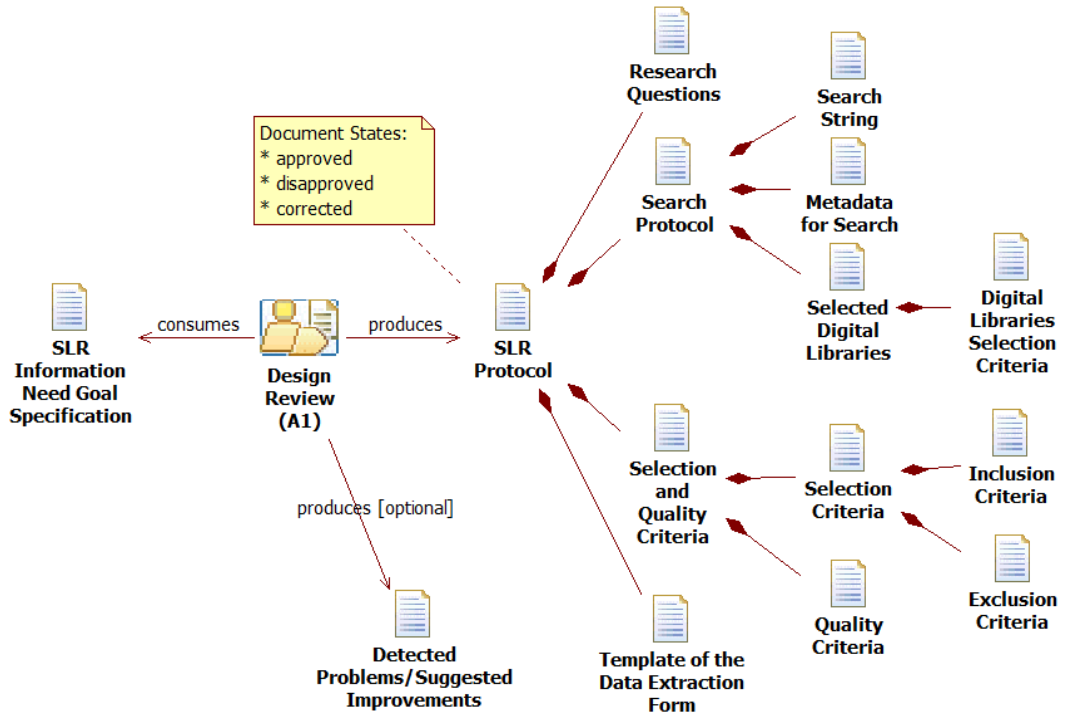


Figura 3-5: Vista informacional para la actividad Diseñar la Revisión (A1). Figura tomada de (Olsina et al., 2019).

Como se muestra en la Figura 3-3, la siguiente actividad es Diseñar el Formulario de Extracción de Datos. Como salida se obtiene la "Plantilla del Formulario de Extracción de Datos", cuyos campos se definen a partir de las "Preguntas de Investigación" y "Criterios de Calidad". Este artefacto producido se utilizará en A2 para recopilar información sobre cada artículo seleccionado. Notar que esta actividad involucra a los roles Diseñador de la RSL y Diseñador Experto en Análisis (según Figura A-1 del Apéndice A). El primero debe tener conocimientos y habilidades para diseñar y especificar el formulario de extracción de datos, mientras que el segundo debe tener experiencia para identificar los tipos de datos necesarios a ser utilizados en el análisis (ver

la nota asociada a la subactividad Diseñar el Formulario de Extracción de Datos en la Figura 3-3). Luego, todos los artefactos producidos hasta el momento deben ser validados. Validar el Diseño de la RSL implica revisar dichos documentos para detectar problemas u oportunidades de mejora. Por lo general, investigadores con experiencia en la realización de RSLs realizan esta actividad (consulte las definiciones de Investigador Experto de la RSL y Validador de la RSL en la Tabla A-1 del Apéndice A). Como resultado se obtiene el documento "Protocolo de la RSL" que contiene todos los artefactos producidos previamente, tal como se representa en la vista informacional de la Figura 3-5.

Por último, es importante mencionar que el documento "Protocolo de la RSL" puede encontrarse en estado *aprobado*, *corregido* o *desaprobado*. En este último caso, también se debe producir una lista de "Problemas Detectados/Mejoras Sugeridas". Este artefacto servirá como insumo para la actividad Mejorar el Diseño de la RSL, que incluye tareas como corregir las preguntas de investigación, corregir la cadena de búsqueda, entre otras tareas, con el fin de introducir cambios en el "Protocolo de la RSL" para mejorarlo. Una vez corregido el protocolo se vuelve a realizar la actividad Validar el Diseño de la RSL con el objetivo de comprobar que el protocolo cumpla con la "Especificación de Meta de Necesidad de Información de RSL". En última instancia, la actividad A1 finaliza cuando se aprueba el "Protocolo de la RSL".

3.2.3.2 Implementar la Revisión (A2)

El objetivo principal de A2 es ejecutar la RSL. El recuadro rosa de la Figura 3-3 muestra las diferentes subactividades y tareas de A2 junto con sus artefactos de entrada y salida. Tenga en cuenta que para aquellos estudios de RSL que se lleven a cabo por primera vez, es decir, que no sean un estudio repetido o replicado, se recomienda realizar primero una prueba piloto (subactividad A2.1) que tiene como objetivo ajustar el "Protocolo de la RSL" producido en la actividad A1. Además, se percibió que esta subactividad A2.1 generalmente se descuida o se especifica de manera deficiente en otros procesos de RSLs existentes.

En la Figura 3-6, se ilustra el flujo de tareas y actividades para un estudio de RSL piloto (A2.1). Cuando se tiene en cuenta en una RSL la realización de dicho estudio piloto, la primera subactividad que se debe realizar es Seleccionar las Bibliotecas Digitales para el Estudio Piloto, la cual es llevada a cabo por el agente que desempeña el rol de Realizador de la RSL. Esta consiste en elegir un subconjunto de bibliotecas (generalmente uno o dos) documentadas en el artefacto "Bibliotecas Digitales Seleccionadas" producido en A1. Luego, se ejecuta la tarea Ejecutar el Protocolo de Búsqueda en las bibliotecas seleccionadas teniendo en cuenta la "Cadena de Búsqueda" y los "Metadatos para la Búsqueda". Como resultado, se produce una lista de "Documentos Recuperados del Estudio Piloto". De esta lista, en la actividad Aplicar los Criterios de Selección, los artículos son descargados y filtrados considerando los "Criterios de Inclusión" y "Criterios de Exclusión". Esto da como resultado el artefacto llamado "Documentos Seleccionados del Estudio Piloto".

A partir de este subconjunto de documentos, los Recolectores de Datos realizan la actividad Extraer Datos de una Muestra de Documentos (ver Figura 3-7). Esta actividad implica la tarea Seleccionar una Muestra de Documentos, la cual puede ser aleatoria (B. Kitchenham, 2004b). Luego, para cada documento, la tarea Extraer los Datos de la Muestra se realiza utilizando la "Plantilla del Formulario de Extracción de Datos". Tenga en cuenta que los datos se extraen de una sola muestra, ya que el objetivo de la prueba

piloto es solo analizar qué tan adecuado es el protocolo que se está siguiendo. Si más de un Recolector de Datos utilizará los formularios en la revisión final, se recomienda entonces que más de uno participe en la extracción de datos del estudio piloto. Probar los formularios por parte de diferentes Recolectores de Datos puede ser útil para encontrar inconsistencias.

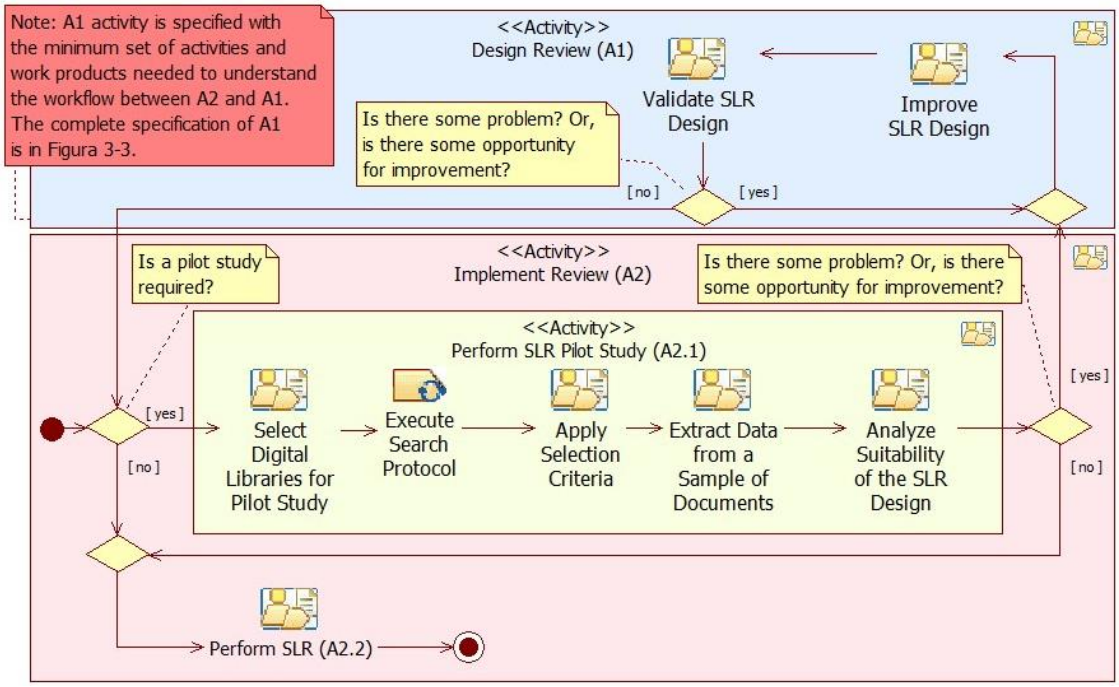


Figura 3-6: Vista de comportamiento para la actividad Realizar el Estudio Piloto de RSL (A2.1). Figura tomada de (Olsina et al., 2019).

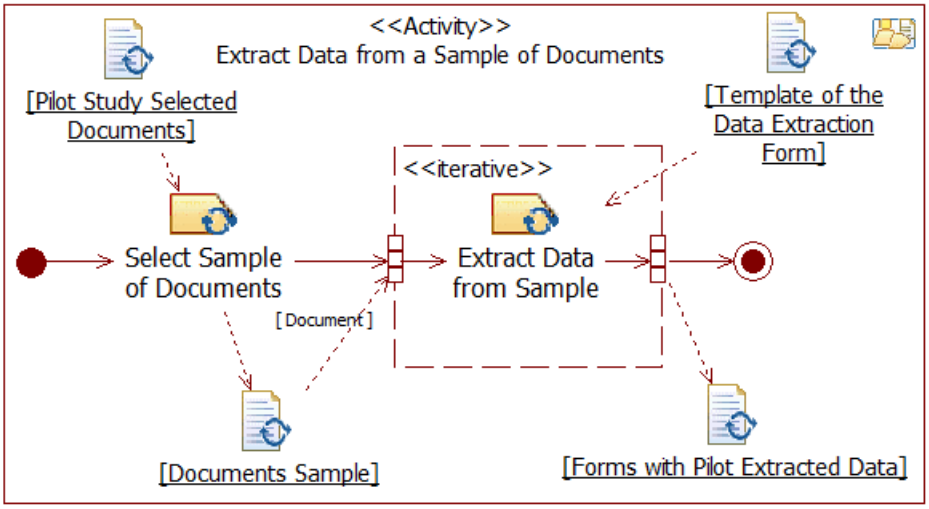


Figura 3-7: Vistas funcional y de comportamiento para la subactividad Extraer Datos de una Muestra de Documentos. Figura tomada de (Olsina et al., 2019).

Finalmente, considerando todas las partes que integran el artefacto "Protocolo de la RSL" (Figura 3-5) así como los "Formularios con Datos Extraídos del estudio Piloto", la actividad Analizar la Adecuación del Diseño de la RSL es realizada por los agentes que desempeñan los roles Validador de la RSL y Experto en el Dominio. Este análisis permite ajustar el formulario de extracción de datos además de otros aspectos del protocolo como las preguntas de investigación, la cadena de búsqueda y/o los criterios de selección. Por ejemplo, un método para validar la cadena de búsqueda es chequear si un conjunto de

documentos ya conocidos se recupera entre los "Documentos Recuperados del Estudio Piloto". Cuando no se detecta ningún problema en el protocolo se lleva a cabo la actividad Realizar la RSL (A2.2). Sin embargo, si se detecta un problema o existe una oportunidad de mejora, las actividades Mejorar el Diseño de la RSL y Validar el Diseño de la RSL deben realizarse nuevamente, como se muestra en la vista de comportamiento especificada en la Figura 3-6. Una vez que se realicen todos los cambios y el "Protocolo de la RSL" ha sido aprobado, se debería ejecutar la subactividad A2.2. Notar que, según la Figura 3-6, se podría realizar un nuevo ciclo de estudio piloto, si fuera necesario.

Realizar la RSL (A2.2) implica la tarea Ejecutar el Protocolo de Búsqueda teniendo ahora en cuenta todas las "Bibliotecas Digitales Seleccionadas". El Realizador de la RSL debe Aplicar los Criterios de Selección en los "Documentos Recuperados" para filtrar aquellos que no cumplan con los criterios definidos en A1. Como artefacto resultante, los "Documentos seleccionados" se utilizan como entrada para la subactividad Agregar los Estudios Relevantes No Recuperados. Esta actividad generalmente se realiza utilizando un método de búsqueda basado en citas, por ejemplo, el método *forward snowballing* (es decir, encontrar los artículos que citaron a los artículos encontrados mediante el proceso de búsqueda) o el método de *backward snowballing* (es decir, encontrar los artículos referenciados en los artículos encontrados mediante el proceso de búsqueda).

Finalmente, la actividad Extraer los Datos de Todos los Documentos se realiza utilizando la "Plantilla del Formulario de Extracción de Datos". Esta actividad es realizada por uno o más Recolectores de Datos. Dependiendo de la experiencia de los agentes Recolectores de Datos, los recursos disponibles, la cantidad de artículos, entre otros factores, un artículo determinado puede ser analizado por uno o dos agentes. En los casos en que el mismo artículo sea leído de forma independiente por varios agentes (como Recolectores de Datos), los datos extraídos deben compararse y los desacuerdos deben resolverse por consenso entre ellos o por un investigador adicional, tal vez por un agente que desempeñe el rol de Investigador Experto de la RSL. Si cada documento es revisado por un solo agente Recolector de Datos ya sea debido a limitaciones de tiempo o recursos, es importante asegurarse de utilizar algún método para verificar la consistencia. Notar que, como se indica la Figura 3-3, las discrepancias deben registrarse en el "Reporte de Resolución de Discrepancias", como así también se sugiere en (Biolchini et al., 2005). Una vez que se completa A2, el artefacto "Formularios con los Datos Extraídos" está disponible para la actividad A3.

3.2.3.3 Analizar y Documentar la Revisión (A3)

A3 es una actividad central del proceso de una RSL (o un mapeo sistemático). El principal objetivo de esta actividad es sintetizar los resultados del análisis en base a la evidencia científica disponible para extraer conclusiones y comunicar los hallazgos. Además, considerando que una RSL debe ser sistemática, reproducible y auditable, la documentación del proceso seguido, los métodos aplicados y los artefactos producidos es un aspecto importante a considerar.

La Figura 3-3 (en el recuadro gris) muestra que Analizar los Resultados de la RSL es la primera subactividad que se realizará en A3. Esta implica a su vez la subactividad Diseñar el Análisis de la RSL, la cual es realizada por un agente que desempeña el rol de Diseñador Experto en Análisis, siendo el responsable de identificar los métodos y técnicas de análisis de datos adecuados a utilizar. Como salida, se produce la "Especificación del Análisis de la RSL". Luego, la subactividad Implementar el Análisis de la RSL debe ser llevada a cabo por un Analizador de Datos, que es responsable de realizar el análisis de

los datos. El análisis se lleva a cabo teniendo en cuenta los artefactos "Formularios con los Datos Extraídos" y "Especificación del Análisis de la RSL". En el análisis se pueden utilizar diversos métodos de medición, evaluación, categorización y agregación de datos, así como medios de visualización (como tablas, gráficos, nubes de palabras, entre otros) para dar respuesta a las preguntas de investigación establecidas (por ejemplo, para abordar los hallazgos de las similitudes y diferencias entre los estudios, entre otros). Como resultado, se produce el artefacto "Síntesis de Datos" la cual, generalmente, suele ser descriptiva. Sin embargo, en ciertas ocasiones, es posible complementar una síntesis descriptiva con resúmenes cuantitativos a través de metaanálisis utilizando técnicas aritméticas y estadísticas de manera adecuada.

Finalmente, un Comunicador Experto lleva a cabo la subactividad Documentar/ Comunicar los Resultados. Para ello, primero se establecen mecanismos de difusión, por ejemplo, informes técnicos, artículos de revistas y congresos, entre otros. Luego, se producen los documentos que transmiten los resultados a la comunidad prevista. De esta manera concluye el proceso para una RSL. Es importante mencionar que toda la evidencia recopilada y los resúmenes deberían estar disponibles públicamente por razones de auditabilidad.

3.3 Revisión Sistemática de Literatura sobre Ontologías de Testing de Software

3.3.1 Introducción

La Sección 3.3 tiene como finalidad documentar los resultados de aplicar el proceso de RSL especificado en la Sección 3.2 con el objetivo de identificar, evaluar y resumir la evidencia disponible sobre ontologías de *testing* de software conceptualizadas (es decir, de aquellas ontologías que no son solo una implementación). Recordar que en el Capítulo 2, Sección 2.4, se fundamentó la importancia de contar con una base conceptual ontológica que de soporte a las especificaciones de procesos y métodos que integran una estrategia. Por lo tanto, una vez finalizada la RSL y utilizando sus resultados (además de otras fuentes de información importantes como por ejemplo (ISO, 2016; ISTQB, 2021b)), se debería poder decidir si es conveniente adoptar, adaptar o crear desde cero una ontología de *testing* que sea adecuada para utilizarla en una familia de estrategias integradas de *testing*.

En esta RSL se establecieron tres preguntas de investigación (RQs, del inglés *Research Questions*), a saber: RQ1 pregunta "¿Cuáles son las ontologías conceptualizadas para el dominio de *testing* de software?"; RQ2: "¿Cuáles son los conceptos incluidos con mayor frecuencia, sus relaciones, propiedades y axiomas necesarios para describir el dominio de *testing* de software?"; y RQ3: "¿Cómo se clasifican las ontologías de pruebas de software existentes?". Como resultado de llevar a cabo este estudio secundario 12 ontologías de *testing* fueron seleccionadas y evaluadas con el fin de decidir si alguna de estas ontologías era lo suficientemente adecuada para adoptarla/adaptarla e integrarla como parte de una estrategia de *testing*. Como conclusión, se notó que ninguna de las ontologías existentes para el dominio de las pruebas de software era adecuada para el objetivo perseguido.

También es importante recordar que, como se mencionó en el Capítulo 1, para llevar a cabo esta RSL primero se realizó una prueba piloto considerando como fuente solamente a Scopus, la cual se documenta en (Tebes, Peppino, Dameno, et al., 2018). En otras palabras, se ejecutaron las actividades A1, A2.1 y Mejorar el Diseño de la RSL del proceso de la Figura 3-2. Luego, utilizando el Protocolo de la RSL mejorado considerando

los resultados de la prueba piloto, se llevó a cabo la RSL en su totalidad (es decir, se ejecutaron las actividades A2.2 y A3) y este trabajo se publicó inicialmente en (Tebes et al., 2019). Finalmente, y como resultado de la extensión de (Tebes et al., 2019), se publicó toda la RSL como artículo de revista en (Tebes et al., 2020).

El resto de la Sección 3.3 está estructurada de la siguiente manera: la Sección 3.3.2 comenta brevemente la motivación de la realización del estudio de RSL. La Sección 3.3.3 contiene el resultado de aplicar todo el proceso de RSL, ilustrando todos los artefactos obtenidos. Finalmente, en la Sección 3.3.4, se discuten los principales hallazgos de este trabajo de RSL, entre otras cuestiones. Notar que, más detalle sobre esta RSL en particular se pueden encontrar en (Tebes et al., 2020).

3.3.2 Motivación para llevar a cabo el Estudio de Revisión Sistemática de Literatura sobre Ontologías de Testing

Recordar que, como se mencionó anteriormente en el Capítulo 2 (Sección 2.3), una estrategia de *testing* bien especificada debería integrar una especificación de proceso, una especificación de métodos y herramientas, y una especificación de base conceptual robusta como una ontología. En (Olsina & Becker, 2017) se discute una familia de estrategias que integran las tres capacidades mencionadas y sirven para alcanzar propósitos de evaluación. Estas estrategias están actualmente soportadas semánticamente por un conjunto de ontologías que pertenecen a una arquitectura ontológica llamada FCD-OntoArch (Olsina, 2021, 2023). Esta es una arquitectura ontológica de 5 capas que considera los niveles fundacional, central (*core*), dominio de alto nivel, dominio de bajo nivel, e instancia (ver Figura 3-8). Notar que la versión anterior a FCD-OntoArch se denomina C-INCAMI v.2 (*Contextual-Information Need, Characteristic Model, Attribute, Metric and Indicator*), la cual se documenta en (Becker et al., 2015; Olsina & Becker, 2018).

FCD-OntoArch está actualmente poblada con un conjunto de vocabularios o terminologías, las cuales se estructuran como ontologías. Por ejemplo, ThingFO se encuentra documentada en (Olsina, 2021), FRsTDO y NFRsTDO están definidas en (Becker et al., 2019; Olsina et al., 2023; Tebes, Olsina, et al., 2020a), y SituationCO en (Olsina et al., 2021). Además, en ese momento, la ontología para *testing* llamada TestTDO todavía no existía ya que se desarrolló luego de llevar a cabo este estudio de RSL.

Teniendo en cuenta que ya existen estrategias integradas que brindan soporte para alcanzar propósitos de evaluación, también se podrían desarrollar estrategias que ayuden a alcanzar propósitos de *testing*. Dado que una estrategia debe integrar una base conceptual robusta, una estrategia de *testing* bien especificada también debería tener esta capacidad. En esta dirección, se decidió llevar a cabo el estudio de RSL para identificar, evaluar y resumir la investigación disponible sobre ontologías de pruebas de software conceptualizadas (es decir, de aquellas ontologías que no son solo una implementación). Una vez finalizada la revisión y utilizando sus resultados se debería poder establecer una ontología de prueba adecuada que se puede reutilizar en una familia de estrategias de prueba. Además, esta ontología debe poder integrarse en la arquitectura FCD-OntoArch, permitiendo su relación con las ontologías de NFRsTDO y FRsTDO (como se muestra en la Figura 3-8).

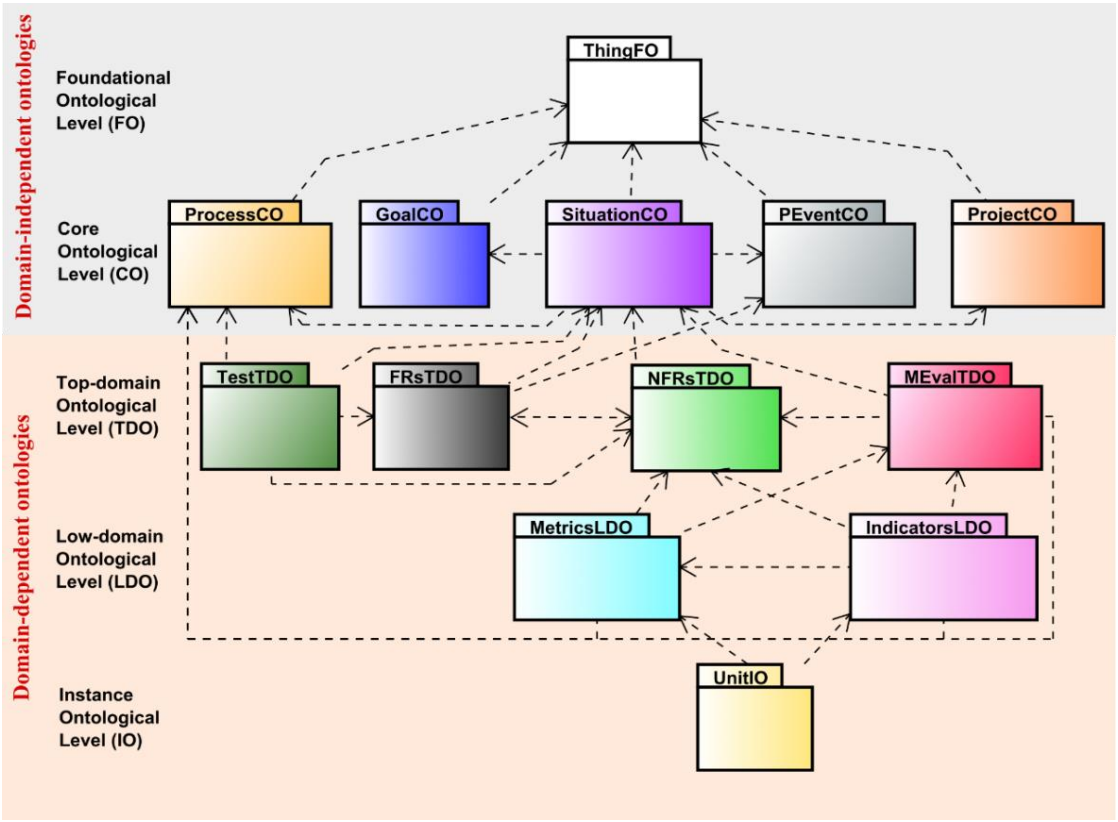


Figura 3-8: La arquitectura ontológica de cinco niveles denominada FCD-OntoArch, poblada con algunas ontologías ya desarrolladas. Notar que, PEvent significa *Particular Event*, FRs es *Functional Requirements*, NFRs es *Non-Functional Requirements*, y MEval es *Measurement and Evaluation*.

3.3.3 Ejecución de la Revisión Sistemática de Literatura sobre Ontologías de Testing

A continuación, se describe la ejecución de las actividades A1-A3 del proceso ilustrado en la Figura 3-3, las cuales fueron realizadas para llevar a cabo una RSL sobre ontologías conceptualizadas en el dominio de las pruebas de software. En la Sección 3.3.3.1 se discute una primera iteración de las actividades A1 (Diseñar la Revisión) y A2.1 (Realizar el Estudio Piloto de RSL), en donde se obtuvo una primera versión del "Protocolo de la RSL" y luego fue mejorado teniendo en cuenta los resultados de la prueba piloto. Finalmente, se ilustra la versión final del protocolo con todos los artefactos que lo componen (recordar Figura 3-5). Luego, en la Sección 3.3.3.2, se muestran los resultados de ejecutar la actividad A2 (Implementar la Revisión). Finalmente, en la Sección 3.3.3.3, se documenta detalladamente el análisis (actividad A3) y los hallazgos de los datos extraídos de los 12 estudios primarios seleccionados.

Es importante mencionar que las actividades A1 y A2.1 se realizaron entre finales de mayo y principios de agosto del 2018 por los miembros del grupo de investigación GIDIS_Web. Además, el protocolo resultante también fue validado por miembros externos a GIDIS_Web pertenecientes a la Universidad ORT de Uruguay. Luego, miembros de ambos grupos de investigación realizaron la actividad A2.2 entre finales de octubre del 2018 y principios de marzo del 2019. Por último, la actividad A3 fue llevada a cabo solamente por los miembros del grupo GIDIS_Web, la cual dio como resultado final a (Tebes et al., 2020).

3.3.3.1 Diseñar la Revisión (A1) y Realizar el Estudio Piloto de RSL (A2.1)

De acuerdo al proceso de RSL ilustrado en la Figura 3-3, la primera tarea a realizar es Especificar las Preguntas de Investigación. El artefacto necesario para realizarla es la "Especificación de Meta de Necesidad de Información de RSL" que contiene, para este caso particular, una declaración establecida por los investigadores como también los requisitos que debe cumplir la ontología de pruebas de software que se adoptará, adaptará, o creará. Entonces, por un lado, la declaración establece: "El grupo de investigación GIDIS_Web necesita un análisis de las ontologías de pruebas de software conceptualizadas para poder establecer la ontología de prueba adecuada que será integrada en la arquitectura ontológica llamada FCD-OntoArch (recordar que su versión anterior era el marco conceptual C-INCAMI v.2). La ontología resultante debería estar vinculada al menos con los componentes conceptuales de requisitos funcionales y requisitos no funcionales (es decir, FRsTDO y NFRsTDO en Figura 3-8)". Por otra parte, los requisitos ontológicos (OR, del inglés *Ontological Requirements*) establecidos para la ontología de pruebas de software a seleccionar son:

- OR#1. Debe estar conceptualizada pero no necesariamente implementada.
- OR#2. Debe tener una amplia cobertura para el dominio de pruebas de software. Por lo tanto, debe contener conceptos relacionados con las pruebas estáticas y dinámicas, así como con las pruebas funcionales y no funcionales.
- OR#3. Debe considerar diferentes tipos de relaciones, es decir, relaciones taxonómicas ("tipo-de/es-un" y "todo-parte/parte-de") y relaciones no taxonómicas (asociaciones).
- OR#4. Debe estar a nivel ontológico de dominio superior (*top-domain ontological level* en Figura 3-8).
- OR#5. Debe desarrollarse siguiendo una metodología adecuada y rigurosa.
- OR#6. Debe desarrollarse considerando términos de estándares internacionales y/u otras ontologías, taxonomías o glosarios para el dominio de pruebas de software.
- OR#7. Debe contener términos del dominio de pruebas que puedan relacionarse con conceptos de requisitos funcionales y no funcionales.
- OR#8. Debe tener una conceptualización representada gráficamente en un lenguaje de modelado apropiado.
- OR#9. Debe tener términos que se basen o estén enriquecidos con términos de otras ontologías en niveles superiores, como ontologías de nivel central y/o fundamental (*core ontological level* y *foundational ontological level*, respectivamente, en Figura 3-8).

A partir de la "Especificación de Meta de Necesidad de Información de RSL" establecida, inicialmente se formularon dos "Preguntas de Investigación" (o "Research Questions" según Tabla 3-2), a saber: (RQ1) ¿Cuáles son las ontologías existentes para el dominio de pruebas de software? Y, (RQ2) ¿Cuáles son los conceptos relevantes, sus relaciones, atributos y restricciones o axiomas necesarios para describir el dominio de pruebas de software? Responder la RQ1 permitirá identificar y analizar las diferentes ontologías de pruebas de software existentes. La RQ2 servirá para conocer los términos (o conceptos), sus relaciones, atributos o propiedades y restricciones necesarias para especificar una ontología para el dominio de *testing*.

Luego, se diseñó el "Protocolo de Búsqueda". Teniendo en cuenta la RQ1, inicialmente se propuso la siguiente cadena de búsqueda: *"Software Testing" AND*

("Ontology" OR "Conceptual Base"). Para este estudio en particular la cadena de búsqueda se aplicó en los tres metadatos seleccionados de un artículo científico, a saber: título, resumen y palabras clave. Finalmente, las bibliotecas digitales seleccionadas incluidas en la revisión fueron Scopus, IEEE Xplore, ACM Digital Library, Springer Link y Science Direct.

Los "Criterios de Selección y Calidad" definidos en esta primera ejecución de la actividad A1 se encuentran en la Tabla 3-2 con el código WP 1.3. Luego, en función de las preguntas de investigación y los criterios de calidad, el Diseñador de la RSL junto al Diseñador Experto en Análisis definieron los campos para el formulario de extracción de datos (ver WP 1.4 en Tabla 3-2). Además, para extraer términos, propiedades, relaciones y axiomas de cada ontología, el campo "conceptos relevantes utilizados para describir el dominio de pruebas de software" contiene un subconjunto de otros campos como se muestra en la Tabla 3-2. Esta "Plantilla del Formulario de Extracción de Datos" permite obtener homogeneidad entre los datos extraídos de cada estudio primario y así facilitar la tarea de analizarlos.

Una vez validados los artefactos producidos se obtuvo el "Protocolo de la RSL" como resultado de A1. Recordar que este protocolo resultante fue validado por miembros de GIDIS_Web y miembros externos pertenecientes a la Universidad ORT de Uruguay. La Tabla 3-2 muestra todos los artefactos que componen este protocolo, los cuales corresponden a los especificados en la vista informacional de la Figura 3-5.

Teniendo en cuenta el flujo entre las actividades que se muestran en la vista de comportamiento de la Figura 3-2, se llevó a cabo un estudio piloto (A2.1) para analizar si el "Protocolo de la RSL" era adecuado. Como parte de la ejecución de A2.1, de las "Bibliotecas Digitales Seleccionadas" en A1 (ver WP 1.2.3 en la Tabla 3-2) se seleccionó Scopus para esta prueba piloto porque contiene recursos digitales de varias fuentes como Elsevier, IEEE Xplore y ACM Digital Library. Como resultado de Ejecutar el Protocolo de Búsqueda se obtuvieron un total de 163 estudios primarios a seleccionar usando los "Criterios de Selección" (ver WP 1.3.1 y 1.3.2 en Tabla 3-2). Luego, al aplicar estos criterios en la actividad Aplicar los Criterios de Selección, quedaron solamente 19 estudios primarios a analizar, los cuales fueron revisados por tres Recolectores de Datos del grupo de investigación GIDIS_Web para Extraer Datos de una Muestra de Documentos (recordar Figura 3-7).

Por último, en la actividad Analizar la Adecuación del Diseño de la RSL, se elaboró una lista de "Problemas Detectados/Mejoras Sugeridas" con el objetivo de mejorar la primera versión del "Protocolo de la RSL" que se muestra en la Tabla 3-2. Luego, una vez que se completó A2.1 y debido a que se detectaron oportunidades de mejora en el protocolo, se ejecutó la actividad Mejorar el diseño de la RSL (recordar Figura 3-3). En la Tabla 3-3 se muestran las actualizaciones (resaltadas en azul y subrayadas) del "Protocolo de la RSL" mejorado después del estudio piloto. A continuación, se describen algunos cambios considerando el documento "Problemas Detectados/Mejoras Sugeridas".

En la pregunta de investigación RQ1, el término "existentes" (ver RQ1 en Tabla 3-2) fue reemplazado por el término "conceptualizadas". El primer término es más general que el segundo e incluye tanto conceptualizaciones como implementaciones de ontologías de *testing*. Sin embargo, el objetivo principal es recuperar ontologías conceptualizadas independientemente de si están implementadas o no. Notar que el nuevo criterio de inclusión 3 considera este aspecto.

Por otro lado, el término "relevante" en la RQ2 de la primera versión del protocolo influyó negativamente en la cantidad de términos extraídos por cada Recolector de Datos. Por lo tanto, se reformuló esta pregunta de investigación tal como se observa en el WP 1.1 de la Tabla 3-3. Además, se cambió el campo "Conceptos relevantes utilizados..." en el formulario de extracción de datos (ver WP 1.4 de la Tabla 3-2) por "Conceptos especificados...". Este cambio hizo que la extracción de datos fuera más objetiva y fácil de interpretar que con el diseño inicial.

Tabla 3-2: Artefacto "Protocolo de la RSL" producido en la primera ejecución de A1 para la RSL sobre ontologías de pruebas de software. Tabla tomada de (Olsina et al., 2019).

Research Questions (WP 1.1) –Note that WP stands for *Work Product*	
RQ1: What are the existing ontologies for the software testing domain? RQ2: What are the relevant concepts, their relationships, attributes and constraints or axioms needed to describe the software testing domain?	
Search Protocol (WP 1.2)	
Search String (WP 1.2.1)	"Software Testing" AND ("Ontology" OR "Conceptual Base")
Metadata for Search (WP 1.2.2)	Title; Abstract; Keywords
Selected Digital Libraries (WP 1.2.3)	Scopus, IEEE Xplore, ACM Digital Library, Springer Link and Science Direct
Selection and Quality Criteria (WP 1.3)	
Inclusion Criteria (WP 1.3.1)	1) That the work be published in the last 15 years; 2) That the work belongs to the Computer Science area; 3) That the work documents a software testing ontology; 4) That the document is based on research (i.e., it is not simply a "lesson learned" or an expert opinion).
Exclusion Criteria (WP 1.3.2)	1) That the work be a prologue, article summary or review, interview, news, discussion, reader letter, or poster; 2) That the work is not a primary study; 3) That the work is not written in English.
Quality Criteria (WP 1.3.3)	1) Is/Are the research objective/s clearly identified? 2) Is the description of the context in which the research was carried out explicit? 3) Was the proposed ontology developed following a rigorous and/or formal methodology? 4) Was the proposed ontology developed considering also its linking with Functional and Non-Functional Requirements concepts?
Template of the Data Extraction Form (WP 1.4)	
Researcher name; Article title; Author/s of the article; Journal/Congress; Publication year; Digital library; Name of the proposed ontology. Relevant concepts used to describe software testing domain: Terms: TermN: definition Attributes: TermN (AttributeN.1=definition, AttributeN.2=definition,…) Relationships: is_a (TermX_type, TermY_subtype) part_of (TermX_whole, TermY_part) RelationM_name (TermX, TermY): definition RelationZ_without_name (TermX, TermY): definition Axioms or restrictions: Axiom: definition/specification Methodology used to develop the ontology; Research context; Research objective/s; Does the proposed ontology consider its linking with Functional and Non-Functional Requirements concepts?; Additional notes.	

Teniendo en cuenta que una conceptualización robusta de una ontología debe incluir no solo los términos, atributos (propiedades), relaciones (tanto taxonómicas –tipo_de y parte_de- como no taxonómicas) y axiomas, sino que también la definición de sus términos, atributos, relaciones no taxonómicas y la especificación de sus axiomas. Con esto en mente, RQ2 trata de obtener evidencia de cuáles son los conceptos incluidos con mayor frecuencia, además de sus relaciones, atributos y axiomas necesarios para describir el dominio de las pruebas de software. Dado que una conceptualización ontológica requiere suficiente espacio para documentarla, el nuevo criterio de exclusión 1 descarta artículos cortos o *short papers* (ver WP 1.3.2).

Además, la lectura completa de los artículos durante el estudio piloto permitió detectar que se presentaban ontologías de varios tipos, como ontologías fundacionales, de dominio superior y de dominio. Dado que el objetivo final después de ejecutar la RSL era adoptar, adaptar o construir una nueva ontología de dominio superior, esta información resultó relevante. En consecuencia, se agregó una nueva pregunta de investigación (RQ3 en el WP 1.1 en la Tabla 3-3) y el campo "Clasificación de la ontología propuesta" en la "Plantilla del Formulario de Extracción de Datos" (ver WP 1.4 en la Tabla 3-3). En resumen, RQ3 trata de obtener evidencia de la clasificación ontológica que los autores dan a una conceptualización.

Vale la pena mencionar que se había analizado una RSL de ontologías de pruebas de software realizada en (Ferreira de Souza et al., 2013). En la misma se incluyeron tanto ontologías conceptualizadas como implementadas. Por lo tanto, la selección de sus estudios primarios fue menos restrictiva que en la RSL aquí propuesta. Además, en (Ferreira de Souza et al., 2013) no se encontraron ontologías de pruebas de software disponibles antes del año 2003. Por lo tanto, utilizando este conocimiento, se reformuló el criterio de inclusión 1, el cual delimita que los estudios primarios sean buscados en el único rango de años donde se conoce que hay trabajos que documentan ontologías de pruebas de software.

También se modificó ligeramente la cadena de búsqueda (comparar WP 1.2.1 en las Tablas 3-3 y 3-4) porque no todos los motores de búsqueda toman en cuenta variaciones o sinónimos de las palabras utilizadas, como sí lo hace Scopus que fue el utilizado en el estudio piloto. La lectura completa de los artículos también permitió detectar que algunos de ellos eran versiones (o fragmentos) diferentes de una misma ontología. En consecuencia, se agregaron los criterios de exclusión 5 y 7 (ver WP 1.3.2 en la Tabla 3-3). Por otro lado, dado que las búsquedas en Scopus recuperan documentos que pertenecen a otras bibliotecas digitales, se agregó el criterio de exclusión 6 del WP 1.3.2 para eliminar estudios primarios recuperados que estén duplicados.

Tabla 3-3: La nueva versión del "Protocolo de la RSL" después de que se realizó la actividad del estudio piloto (A2.1). Notar que los cambios se indican en azul y subrayados con respecto a la información que se muestra en la Tabla 3-2. Tabla tomada de (Olsina et al., 2019).

Research Questions (WP 1.1) –Note that WP stands for *Work Product*	
RQ1: What are the conceptualized ontologies for the software testing domain? RQ2: What are the most frequently included concepts, their relationships, attributes and axioms needed to describe the software-testing domain? RQ3: How are existing software testing ontologies classified?	
Search Protocol (WP 1.2)	
Search String (WP 1.2.1)	("Software Testing" OR "Software Test") AND ("Ontology" OR "Ontologies")
Metadata for Search (WP 1.2.2)	Title; Abstract; Keywords
Selected Digital	Scopus, IEEE Xplore, ACM Digital Library, Springer Link and Science Direct

Libraries (WP 1.2.3)	
Selection and Quality Criteria (WP 1.3)	
Inclusion Criteria (WP 1.3.1)	1) That the work be published in the last 15 years (from the beginning of 2003 until November 12, 2018); 2) That the work belongs to the Computer Science area or to the Software/System/Information Engineering areas; 3) That the document has the ontological conceptualization of the testing domain (i.e., it is not simply a "lesson learned or expert opinion" or just an implementation).
Exclusion Criteria (WP 1.3.2)	1) That the work be a prologue, article summary or review, interview, news, discussion, reader letter, poster, table of contents or short paper (a short paper is considered to that having up to 4 pages size); 2) That the work is not a primary study; 3) That the work is not written in English; 4) That the work does not document a software testing ontology; 5) That the ontology presented in the document be an earlier version than the most recent and complete one published in another retrieved document; 6) That a same document be the result of more than one bibliographic source (i.e., it is duplicated); 7) That the conceptualized ontology in the current document be a fragment of a conceptualized ontology in another retrieved document.
Quality Criteria (WP 1.3.3)	1) Is/Are the research objective/s clearly identified? 2) Is the description of the context in which the research was carried out explicit? 3) Was the proposed ontology developed following a rigorous and/or formal methodology? 4) Was the proposed ontology developed considering also its linking with Functional and Non-Functional Requirements concepts? 5) What other terminologies of the software testing domain were taken into account to develop the proposed ontology?
Template of the Data Extraction Form (WP 1.4)	
Researcher name; Article title; Author/s of the article; Journal/Congress; Publication year; Digital library; Name of the proposed ontology. Specified concepts used to describe software testing domain: Terms: TermN: definition Attributes: TermN (AttributeN.1=definition, AttributeN.2=definition,…) Relationships: is_a (TermX_type, TermY_subtype) part_of (TermX_whole, TermY_part) RelationM_name (TermX, TermY): definition RelationZ_without_name (TermX, TermY): definition Axioms or restrictions: Axiom: definition/specification Methodology used to develop the ontology; Terminologies or Vocabularies taken into account to develop the proposed ontology; Classification of the proposed ontology; Research context; Research objective/s related to software testing ontologies; Does the proposed ontology consider its linking with Functional and Non-Functional Requirements concepts?; Additional notes.	

Finalmente, también se observó que algunas ontologías fueron construidas teniendo en cuenta otras terminologías (por ejemplo, glosarios de estándares internacionales, ontologías y/o taxonomías), lo que puede agregar un grado de calidad a la nueva propuesta. Por esta razón, se agregó el criterio de calidad 5 en el WP 1.3.3 de la Tabla 3-3, lo que implica un nuevo campo en la "Plantilla del Formulario de Extracción de Datos" (ver "Terminologías o Vocabularios tomados en cuenta…" en WP 1.4). Este nuevo criterio de calidad puede proporcionar información útil del proceso de construcción de

cualquier ontología. Además, notar que este criterio está directamente relacionado con el requisito ontológico OR#6.

Tabla 3-4: Versión final del artefacto "Plantilla del Formulario de Extracción de Datos". Nota: el PID representa el número único que identifica cada documento recuperado; y FF significa campo de formulario (*Form Field*). Tabla tomada de (Tebes et al., 2020).

Template of the Data Extraction Form	
Researcher name (FF1)	Last Name (Surname), First Name
PID - Article title (FF2)	
Author/s of the article (FF3)	Author1_Surname, Initials_of_the_Name; Author2_Surname, Initials_of_the_Name; ...
Journal/Congress/other (FF4)	Name
Publication year (FF5)	
Digital library (FF6)	Name (Note: *If an article was repeated, indicate the name of all the libraries in which was found*)
Name of the proposed ontology (FF7)	Name and/or Acronym
Specified concepts used to describe software testing domain (FF8)	Terms: TermN: definition Attributes (Properties): TermN (AttributeN.1 = definition, AttributeN.2 = definition,…) Relationships: is_a (TermX_type, TermY_subtype) part_of (TermX_whole, TermY_part) RelationM_name (TermX, TermY): definition RelationZ_without_name (TermX, TermY): definition Axioms or restrictions: Axiom: definition/specification
Methodology used to develop the ontology (FF9)	Name and/or Acronym
Terminologies or Vocabularies taken into account to develop the proposed ontology (FF10)	Ontologies: name/s and/or reference/s Taxonomies: name/s and/or reference/s Glossaries/Dictionaries: name/s and/or reference/s
Classification of the proposed ontology (FF11)	Select one category: 1- Foundational Ontology (top level) 2- Top-Domain Ontology 3- Domain Ontology 4- Instance/Application Ontology 5- Not determined by authors
Research context (FF12)	Description
Research objective/s related to software testing ontologies (FF13)	Description
Does the proposed ontology consider its linking with Functional and Non-Functional Requirements concepts? (FF14)	1- Yes 0- No
Additional notes (FF15)	(*This field is used for the researcher to make his/her observations or comments about something that is important to highlight*)

Además, también quedaron definidos otros cuatro criterios de calidad, los cuales fueron producidos teniendo en cuenta las RQs establecidas y algunos de los requisitos ontológicos mencionados anteriormente. Es importante remarcar que estos criterios fueron utilizados para obtener información que ayude a evaluar la calidad de los documentos seleccionados y no para excluirlos o incluirlos (como sí lo hacen Garousi *et al.*, por ejemplo, en (Garousi & Mäntylä, 2016)). Los criterios de calidad 1 y 2 tienen como objetivo identificar el objetivo y el contexto de la investigación en la que se

desarrolló la ontología de *testing*. Esta información es valiosa ya que se requiere una ontología que se integre en una arquitectura ontológica que sustente una familia de estrategias. En el caso de encontrar una ontología que tenga un objetivo y/o contexto similar a este podría ser de gran utilidad.

El criterio de calidad 3 tiene como intención indagar sobre la metodología o enfoque específico utilizado para desarrollar la ontología. Por otro lado, el criterio 4 tiene como finalidad establecer si la ontología está directamente relacionada con conceptos de requisitos funcionales y no funcionales. Esta información es importante para el objetivo perseguido ya que se requiere una ontología que debe integrarse y relacionarse con los componentes de FRsTDO y NFRsTDO, como muestra la Figura 3-8. Además, notar que los criterios de calidad 3 y 4 están directamente relacionados con los requisitos ontológicos OR#5 y OR#7, respectivamente.

Esta última versión del "Protocolo de la RSL" considera un total de 15 campos en la "Plantilla del Formulario de Extracción de Datos", tal como muestra el WP 1.4 de la Tabla 3-3. Además, se ilustra en la Tabla 3-4 ésta plantilla con algunas indicaciones útiles para los Recolectores de Datos a la hora de llevar a cabo la tarea de recolectar la información de los estudios primarios. Es importante mencionar que el campo 8 del formulario resultó muy útil para recolectar los datos utilizados luego para calcular las métricas durante la actividad A3, como se mostrará más adelante.

3.3.3.2 Implementar la RSL (A2)

Seis agentes, cuatro investigadores de GIDIS_Web (incluyendo al autor de esta tesis) y dos investigadores de ORT Uruguay realizaron la actividad A2.2 entre finales de octubre del 2018 y principios de marzo del 2019. La Figura 3-9 muestra la tarea Ejecutar el Protocolo de Búsqueda y la actividad Aplicar los Criterios de Selección instanciadas para esta RSL sobre ontologías de *testing*. La carga de la tarea Ejecutar el Protocolo de Búsqueda fue dividida entre ambos grupos de investigación ya que, por un lado, los agentes de GIDIS_Web (sombra verde en la figura) recuperaron documentos de las bibliotecas digitales Scopus, ACM e IEEE Xplore y, por otro lado y en paralelo, los agentes de ORT (sombra rosa) recuperaron documentos de las bibliotecas digitales Springer Link y Science Direct. Además, la carga de trabajo también fue balanceada, es decir, dos integrantes de cada grupo participaron en esta tarea en el cumplimiento del rol Realizador de la RSL. Los dos miembros restantes de GIDIS_Web (no incluyendo al autor de esta tesis) participaron como coordinadores y verificadores de consistencia. Además, la Figura 3-9 muestra las tareas instanciadas para la actividad Aplicar los Criterios de Selección. Notar que estas tareas se programan teniendo en cuenta los artefactos que contienen los criterios de inclusión y exclusión ilustrados en la Tabla 3-3 (ver WP 1.3.1 y WP 1.3.2).

Utilizando la especificación del artefacto "Protocolo de búsqueda" (WP 1.2 en la Tabla 3-3) se recuperaron 731 documentos. La Tabla 3-5 muestra los "Documentos Recuperados" por biblioteca digital.

Tabla 3-5: Cantidad de "Documentos Recuperados" por biblioteca digital después de realizar la tarea Ejecutar el Protocolo de Búsqueda. Tabla tomada de (Tebes et al., 2020).

	Scopus	**IEEE Xplore**	**Springer Link**	**ACM DL**	**Science Direct**	**Total**
Amount of Retrieved Documents	181	88	443	18	1	**731**

La Tabla 3-6 contiene el número de "Documentos seleccionados" obtenido después de realizar la subactividad Aplicar los Criterios de Selección. Como resultado final se obtuvieron 10 estudios primarios seleccionados al aplicar los diferentes criterios de inclusión/exclusión sobre el contenido analizado como se presenta en la primera y segunda columna de la Tabla 3-6, respectivamente. Considerando el estado inicial y el final, la reducción alcanzó un 98,6%. La siguiente actividad que se llevó a cabo fue Agregar los Estudios Relevantes No Recuperados (recordar Figura 3-3). Para esto se utilizó el método *backward snowballing*, además que se tenía conocimiento previo de otros trabajos de investigación relacionados. La Tabla 3-7 muestra los resultados del uso del método *backward snowballing* con dos iteraciones. Tenga en cuenta que los 65 "Documentos recuperados" de las referencias se verificaron en busca de duplicados con los 731 estudios iniciales. Luego, la Tabla 3-8 muestra el número de "Documentos seleccionados finales" después de finalizar la subactividad Agregar los Estudios Relevantes No Recuperados, que totalizó 12 estudios primarios de investigación. Notar que se incluyó una tesis de maestría (Asman & Srikanth, 2015) después de superar todos los criterios de inclusión y exclusión, la cual fue recuperada por conocimiento previo de otros trabajos relacionados (ver "Other Method" en Tabla 3-8).

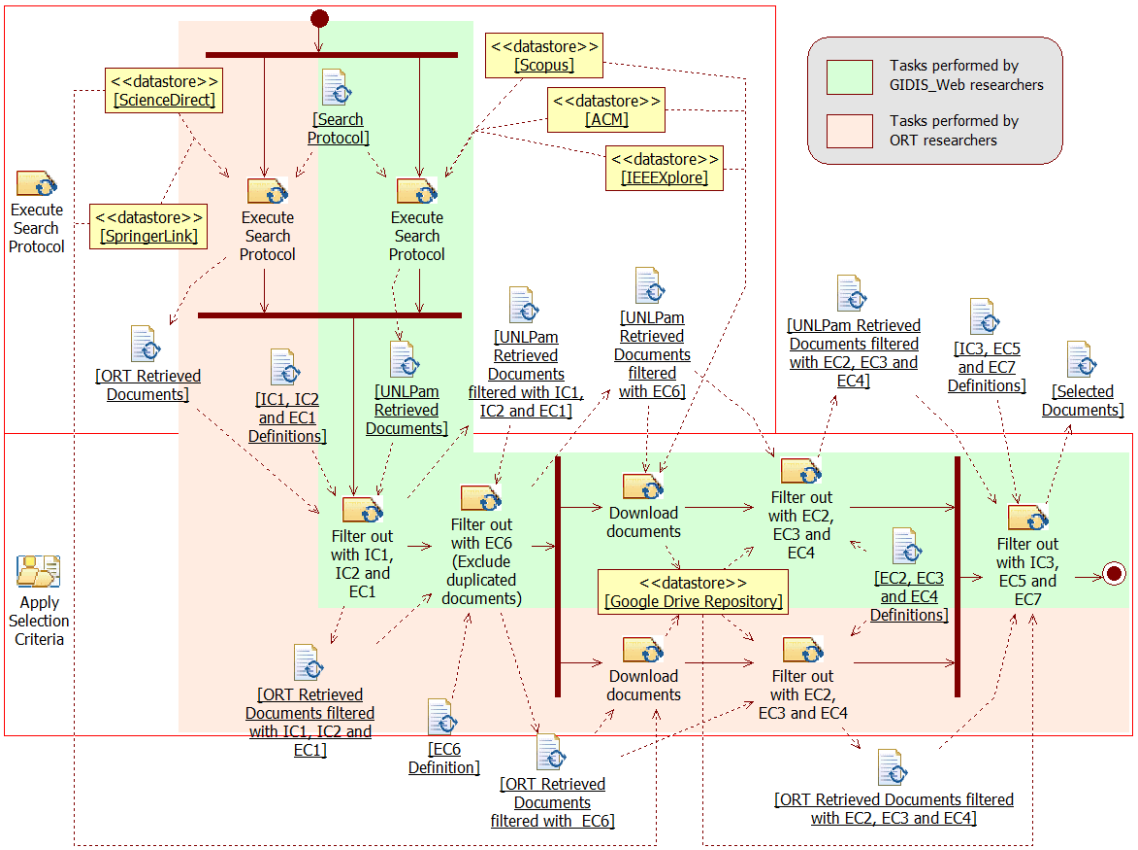


Figura 3-9: Instanciación de las tareas Ejecutar el Protocolo de Búsqueda y Aplicar los Criterios de Selección. Tenga en cuenta que IC y EC significa Criterio de Inclusión y Criterio de Exclusión, respectivamente. Figura tomada de (Olsina et al., 2019).

Finalmente, se debe realizar la subactividad Extraer los Datos de Todos los Documentos, la cual tiene como entrada la "Plantilla del Formulario de Extracción de Datos" (Tabla 3-4) y los "Documentos seleccionados finales" (Tabla 3-8). Como resultado de esta actividad se obtiene el artefacto "Formularios con los datos extraídos" (cargado en https://bit.ly/SLR-TestingOntologies). Esta subactividad requiere mucho tiempo y debe realizarse de manera muy disciplinada y rigurosa. La distribución del

trabajo para la subactividad Extraer los Datos de Todos los Documentos fue la siguiente, dos Recolectores de Datos de GIDIS_Web recolectaron todos los datos requeridos de los 12 documentos. Al mismo tiempo y al azar, se seleccionaron 4 de los 12 documentos que se pusieron a disposición (por Google Drive) para la recolección de datos por parte de los dos Recolectores de Datos de la Universidad ORT Uruguay. La recolección de datos de los dos grupos se realizó de forma aislada y no se compartió hasta el momento en que ambos grupos terminaron la tarea para permitir una verificación más objetiva de la consistencia. Como resultado de esta verificación de los formularios instanciados de ambos grupos, se plantearon algunos problemas menores y se resolvieron las discrepancias.

Tabla 3-6: "Documentos seleccionados" después de realizar la subactividad Aplicar los Criterios de Selección. Tabla tomada de (Tebes et al., 2020).

Criteria	Analyzed Content	Initial Studies	Eliminated	Selected Documents	Reduction (%) w.r.t. the 731 initial papers
IC1, IC2, EC1	Title+Abstract	731	209	522	28.5%
EC6 (Duplicates)	Title+Abstract	522	47	475	6.4%
EC2, EC3, EC4	Full Text	475	456	19	62.3%
IC3, EC5, EC7	Full Text	19	9	10	1.2%
Total			721		**98.6%**

Tabla 3-7: Resultados del uso del método *backward snowballing* en la subactividad Agregar los Estudios Relevantes No Recuperados. Tabla tomada de (Tebes et al., 2020).

Backward Snowballing		1st Iteration	2nd Iteration	Total
Retrieved Documents		52	13	65
Eliminated by Criteria	EC6 (Duplicates)	49	13	62
	IC1, IC2, EC1	0	0	0
	EC2, EC3, EC4	1	0	1
	IC3, EC5, EC7	1	0	1
Final Studies		1	0	1

Tabla 3-8: "Documentos seleccionados finales" después de realizar Agregar los Estudios Relevantes No Recuperados. Tabla tomada de (Tebes et al., 2020).

Selected Documents	Backward Snowballing	Other Method	Final Selected Documents
10	1	1	**12**

Vale la pena mencionar que, debido a la búsqueda de inconsistencias, se detectó que en los conceptos recopilados (es decir, términos, propiedades, etc.) incluidos en el formulario para la ontología ROoST (Ferreira de Souza et al., 2017) no solo había conceptos específicos del dominio de *testing*, sino que también se encontraron conceptos relacionados con la ontología fundacional denominada UFO (Guizzardi, 2005). Por lo tanto, se decidió documentar ambos de forma diferenciada (por colores en el formulario) para que, al momento del análisis, cuenten solo los conceptos pertenecientes al dominio de las pruebas de software. Por ejemplo, hay en ROoST 77 términos en total, pero solo 45 son términos específicos del dominio de *testing*. Todos los problemas planteados durante la subactividad Extraer los Datos de Todos los Documentos se documentaron en el artefacto "Reporte de Resolución de Discrepancias". La Tabla 3-9 resume aspectos de los datos extraídos para los 12 "Documentos seleccionados finales" ordenados en orden descendente por año de publicación. El acrónimo PID (*paper identification*) representa la identificación del artículo original en la lista completa de artículos recuperados en A2.2.

Tabla 3-9: Resumen de los datos extraídos para los 12 "Documentos seleccionados finales". PID es la identificación del artículo original en los "Formularios con los datos extraídos" disponibles en https://bit.ly/SLR-TestingOntologies. Tabla tomada de (Tebes et al., 2020).

PID	Conceptualized Ontology	Ontological Classification	Publ. Year	Page Size	Development Methodology	Used Terminology [Glossary, Taxonomy, Ontology]	FR/NFR Linking?
347	*RTE-Ontology*, Campos *et al.* (de S. Campos Junior et al., 2017)	Domain	2017	6	Not specified	PROV-O Ontology	No
383	Vasanthapriyan *et al.* (Vasanthapriyan, Tian, Zhao, et al., 2017)	Domain	2017	8	Grüninger& Fox Method	Glossary by ISTQB; IEEE 829-2008	No
457	Vasanthapriyan *et al.* (Vasanthapriyan, Tian, & Xiang, 2017)	Domain	2017	15	Grüninger& Fox Method	Glossary by ISTQB; IEEE 829-2008	No
468	*ROoST*, Ferreira de Souza *et al.* (Ferreira de Souza et al., 2017)	Domain	2017	30	SABiO	Glossary by IEEE 610-1990 & 829-1998; Myers G. J., 2004: "The art of software testing"; SWEBOK; Pressman R., 2006: "Software engineering: a practitioner's approach"; ISTQB; Marthur A. P., 2012: "Foundations of Software Testing"; ISO 29119; SP-OPL; E-OPL; UFO Foundat. Ontology by authors	No
1003	Asman *et al.* (Asman & Srikanth, 2015)	Top Domain	2016	74	Methontology; Ontol.Dev.101	Glossary by ISTQB SWTOI; OntoTest;	No
346	*PTOntology*, Freitas *et al.* (Freitas & Vieira, 2014)	Domain	2014	8	Ontology Dev.101	Glossary by SWEBOK; IEEE Glossary of SE Terminology; IEEE Standard for Software Test Documentation	No
424	Arnicans *et al.* (Arnicans et al., 2013)	Not Specified	2013	14	ONTO6	Glossary by ISTQB	No
118	Sapna *et al.* (Sapna & Mohanty, 2011)	Not Specified	2011	10	Methontology; Ontol.Dev.101	Glossary by SWEBOK	No
366	Cai *et al.* (Cai et al., 2009)	Not Specified	2009	6	Skeletal	Glossary by SWEBOK, ISO 9126	No
359	*TOM*, Bai *et al.* (Bai et al., 2008)	Domain	2008	8	Not specified	UML 2.0 Testing Profile (U2TP)	No
397	*OntoTest*, Barbosa *et al.* (Barbosa et al., 2008)	Not Specified	2008	6	Not specified	Glossary by IEEE 829; ISO 12207; The Software Process Ontology by Falbo & Bertollo; STOWS;	No
1000	*STOWS*, Zhu *et al.* (Zhu & Huo, 2005)	Not Specified	2005	33	Not specified	Not specified	No

3.3.3.3 Analizar y Documentar la Revisión (A3)

La actividad A3 fue realizada por todos los miembros del grupo de investigación GIDIS_Web. A3 implica dos subactividades: Analizar los Resultados de la RSL y Documentar/Comunicar los Resultados como se representa en la Figura 3-3. A su vez, la primera tiene otras dos subactividades, a saber: Diseñar el Análisis de la RSL e Implementar el Análisis de la RSL. Diseñar el Análisis de la RSL consume el artefacto "Formularios con los datos extraídos" y utiliza el repositorio de "Métricas/indicadores y métodos aritméticos/estadísticos" para producir la "Especificación del Análisis de la RSL". A continuación, se presentan algunos requisitos, métricas e indicadores de calidad ontológica utilizados para el análisis y para responder a las preguntas de investigación.

Para diseñar el análisis se generó un árbol de NFRs (requisitos no funcionales, 1ra columna de la Tabla 3-10) que especifica características y atributos relacionados con la calidad ontológica. Es importante remarcar que, en un modelo de calidad, las características son más genéricas que los atributos que son más específicos. Por lo tanto, las características no se pueden medir, pero se pueden evaluar utilizando indicadores derivados. En cambio, los atributos pueden medirse y evaluarse utilizando métricas e

indicadores elementales correspondientes, como se discute en (Olsina et al., 2008). En la 2da columna de la Tabla 3-10 se muestran las definiciones de las características y atributos utilizados. Tenga en cuenta que el árbol de NFRs presentado es útil para evaluar cualquier ontología en general, pero los atributos asociados con la subcaracterística "1.2 Calidad de la cobertura terminológica específica del dominio" son específicos para el dominio de las pruebas de software.

Tabla 3-10: Árbol de NFR (1ra columna) para evaluar la calidad ontológica. Además, las definiciones de sus atributos y características se encuentran en la 2da columna. Tabla tomada de (Tebes et al., 2020).

Characteristics / *Attributes*	**Definition: Degree to which…**
1. Ontological Quality (root)	…an ontology is well structured, has good terminological coverage and adheres to other vocabularies.
1.1 Ontological Structural Quality	…an ontology is well structured, i.e., has defined terms availability, defined properties availability, specified axioms availability and it is properly balanced with regard to types of relationships.
1.1.1 Defined Terms Availability	…an ontology has defined terms.
1.1.2 Defined Properties Availability	…an ontology has defined properties.
1.1.3 Specified Axioms Availability	…an ontology has specified axioms.
1.1.4 Balanced Relationships Availability	…an ontology has a balance between the amount of non-taxonomic and taxonomic relationships in addition to the former are defined.
1.1.4.1 Balanced Non-Taxonomic Relationships Availability	…an ontology has a balance between the amount of non-taxonomic and taxonomic relationships.
1.1.4.2 Defined Non-Taxonomic Relationships Availability	…an ontology has defined non-taxonomic relationships.
1.2 Domain-specific Terminological Coverage Quality	…an ontology has good terminological coverage of the domain.
1.2.1 Static Testing Terms Availability	…a software testing ontology has terms related to Static Testing.
1.2.2 Dynamic Testing Terms Availability	…a software testing ontology has terms related to Dynamic Testing.
1.2.3 Functional Testing Terms Availability	…a software testing ontology has terms related to Functional Testing.
1.2.4 Non-Functional Testing Terms Availability	…a software testing ontology has terms related to Non-Functional Testing.
1.3 Compliance to other Vocabularies	…an ontology adheres its terminology with other vocabularies.
1.3.1 Terminological Compliance to International Standard Glossaries	…an ontology adheres its terminology to international standard glossaries.
1.3.2 Terminological Compliance to other Domain/Core Ontologies	…an ontology adheres its terminology to other domain or core ontologies.
1.3.3 Terminological Compliance to Foundational Ontologies	…an ontology adheres its terminology to a foundational ontology.

Solo por describir resumidamente algunos aspectos de medición y evaluación (ver (Becker et al., 2015; Olsina et al., 2014) para una descripción más amplia), un indicador es un método (recurso) que utiliza un valor de indicador de un atributo o característica para evaluarlo. Además, un indicador tiene niveles de aceptabilidad (o criterios de decisión) que ayudan a interpretar el valor del indicador. Mediante el uso de los niveles de aceptabilidad podemos determinar si un atributo o característica satisface un cierto nivel de calidad (por ejemplo, los niveles de calidad insatisfactorio, marginal y satisfactorio en la Figura 3-12) y así proponer acciones de cambio cuando sea necesario.

Existen dos tipos de indicadores, a saber: elementales y derivados o globales. El primero se utiliza para evaluar atributos (como NFRs elementales que deben satisfacerse), mientras que el segundo se utiliza para evaluar características y subcaracterísticas de calidad. Es importante señalar que un valor de indicador de una característica se obtiene mediante el uso de un modelo de agregación (por ejemplo, el aditivo lineal o los modelos de lógica de preferencia (Dujmovic, 1996)), que consume los valores indicadores de los atributos y/o subcaracterísticas relacionadas con el fin de calcular la característica. En

cambio, el valor del indicador de un atributo se obtiene utilizando un modelo elemental, que consume su valor medido.

Para construir el árbol de NFRs presentado en la Tabla 3-10 se consideraron principalmente las prácticas de calidad descritas en (d'Aquin & Gangemi, 2011) para el diseño de ontologías. Dos (de tres) dimensiones son la estructura formal y la cobertura conceptual, que se caracterizan por si la ontología está diseñada basándose en principios, es formalmente rigurosa, implementa relaciones no taxonómicas, tiene una amplia cobertura del dominio, implementa un estándar internacional, reutiliza ontologías fundacionales, entre otras.

Tenga en cuenta que en (Ferreira de Souza et al., 2017) se realizó un análisis cualitativo y se discutieron los hallazgos de su RSL considerando las características descritas en (d'Aquin & Gangemi, 2011). En el presente trabajo se incrementa este análisis desde el punto de vista cuantitativo utilizando métricas e indicadores.

El árbol de NFRs diseñado es útil para realizar la evaluación de la calidad de las ontologías. Además, ayuda a diseñar y seleccionar las métricas e indicadores adecuados. Entonces, luego de construir el árbol de NFRs, se diseñaron métricas e indicadores elementales para cada atributo. Asimismo, se seleccionó un modelo de agregación de indicadores derivados considerando las características y subcaracterísticas. A modo de ejemplo, en la Figura 3-10 se muestra la especificación de una métrica indirecta para el atributo "Disponibilidad de relaciones no taxonómicas balanceadas" (1.1.4.1). Además, la Figura 3-11 muestra una métrica directa relacionada con esta métrica indirecta. Tenga en cuenta que una métrica indirecta es una métrica que especifica una fórmula, la cual utiliza otras métricas. Además, en la Figura 3-12 se ilustra un indicador elemental para el atributo en cuestión (1.1.4.1).

<table>
<tr><td colspan="2">INDIRECT METRIC</td></tr>
<tr><td colspan="2">Name: Percentage of Taxonomic-Conceptual-base Level (%TxCptuaLvl)</td></tr>
<tr><td colspan="2">Objective: Determine the percentage of taxonomic relationships with respect to the total amount of relationships of a conceptualized ontology.</td></tr>
<tr><td>Author: Tebes – Peppino</td><td>Version: 1.0</td></tr>
<tr><td colspan="2">Calculation Procedure (specification):
Formula: $\%\text{TxCptuaLvl} = \frac{\#TxRh}{\#TxRh + \#\text{NoTxRh}} * 100$
where: #TxRh stands for Total amount of Taxonomic Relationships; #NoTxRh stands for Total amount of Non-Taxonomic Relationships; and (#TxRh + #NoTxRh) ≠ 0.</td></tr>
<tr><td>Scale: Numerical</td><td>Representation: Continuous</td></tr>
<tr><td>Value Type: Real</td><td>Scale Type: Ratio</td></tr>
<tr><td>Unit: Percentage</td><td>Acronym: %</td></tr>
<tr><td colspan="2">Related Indirect Metrics:
1) Total amount of Taxonomic Relationships (#TxRh): #TxRh = #Tx-is_a + #Tx-part_of
Related Direct Metrics:
1) Total amount of Non-Taxonomic Relationships (#NoTxRh) –see Figura 3-12</td></tr>
</table>

Figura 3-10: Especificación de la métrica indirecta para cuantificar el atributo "Disponibilidad de relaciones no taxonómicas balanceadas" (1.1.4.1). Figura tomada de (Tebes et al., 2020).

DIRECT METRIC	
Name: Total amount of Non-Taxonomic Relationships (#NoTxRh)	
Objective: Determine the number of non-taxonomic relationships of a conceptualized ontology.	
Author: Tebes – Peppino	**Version:** 1.0
Measurement Procedure: **Type:** Objective **Specification:** Given a conceptual model of an ontology, the expert examines it in order to count the number of non-taxonomic relationships present in it.	
Scale: Numerical	**Representation:** Discrete
Value Type: Natural (including 0)	**Scale Type:** Absolute
Unit: Amount of Relationships	**Acronym**: #

Figura 3-11: Especificación de la métrica directa relacionada con la métrica indirecta de la Figura 3-10. Figura tomada de (Tebes et al., 2020).

ELEMENTARY INDICATOR

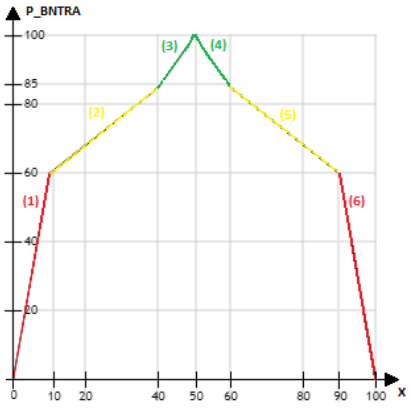
Name: Performance Level of the Balanced Non-Taxonomic Relationships Availability (P_BNTRA)

Author: Tebes – Peppino | **Version:** 1.0

Elementary model:

Specification: the mapping is:

$$\text{P\_BNTRA}(x) = \begin{cases} 6x, & 0 \le x < 10 \ (1) \\ \frac{5}{6}x + \frac{155}{3}, & 10 \le x < 40 \ (2) \\ \frac{3}{2}x + 25, & 40 \le x \le 50 \ (3) \\ -\frac{3}{2}x + 175, & 50 < x \le 60 \ (4) \\ -\frac{5}{6}x + 135, & 60 < x \le 90 \ (5) \\ -6x + 600, & 90 < x \le 100 \ (6) \end{cases}$$



where x is the $\%TxCptuaLvl$ metric

Decision criterion (3 acceptability levels):

Name 1: ■ Unsatisfactory; **Range:** if 0 ≤ P_BNTRA < 60

Description: It indicates that the ontology is unbalanced between the amount of taxonomic and non-taxonomic relationships. Therefore, change actions must be taken into account with high priority.

Name 2: ◆ Marginal; **Range:** if 60 ≤ P_BNTRA < 85

Description: It indicates that the ontology is moderately balanced between the amount of taxonomic and non-taxonomic relationships. Therefore, change actions should be taken into account.

Name 3: ● Satisfactory; **Range:** if 85 ≤ P_BNTRA ≤ 100

Description: It indicates that the ontology is well balanced between the amount of taxonomic and non-taxonomic relationships. Therefore, no change actions are needed.

Scale: Numerical	**Representation:** Continuous
Value Type: Real	**Scale Type:** Ratio
Unit: Percentage	**Acronym**: %

Figura 3-12: Especificación del indicador elemental para el atributo "Disponibilidad de relaciones no taxonómicas balanceadas" (1.1.4.1). Tenga en cuenta que %TxCptuaLvl representa la métrica "Porcentaje de nivel conceptual taxonómico" de la Figura 3-10. Figura tomada de (Tebes et al., 2020).

Siguiendo el procedimiento de medición de la métrica directa (Figura 3-11), se obtiene la medida para cada ontología seleccionada. Cada valor representa el número de relaciones no taxonómicas (#NoTxRh) presentes en cada ontología. Luego, usando la métrica indirecta relacionada, se puede calcular la cantidad total de relaciones taxonómicas (#TxRh). Con estas medidas y usando la fórmula de la Figura 3-10, se calcula el porcentaje de nivel conceptual taxonómico (%TxCptuaLvl). Al interpretar este porcentaje utilizando el indicador especificado en la Figura 3-12, se puede saber si una ontología está bien equilibrada en cuanto a la cantidad de relaciones taxonómicas y no taxonómicas.

Solo se muestran con fines ilustrativos las métricas y el indicador elemental relacionados con un solo atributo. Tenga en cuenta que cada atributo en la Tabla 3-10 tiene una o más métricas relacionadas y un indicador elemental. Por ejemplo, algunas fórmulas para otras métricas indirectas que cuantifican los atributos relacionados con la "Calidad Estructural Ontológica" (1.1) se encuentran a continuación. Notar que las siglas # significa número o cantidad, % porcentaje, Df *Defined*, Tr *Term*, Rh *Relationship*, Tx *Taxonomic*, NoTx *No Taxonomic*, At *Attribute*, y Ax *Axiom*.

%DfTr = (#DfTr / #Tr) * 100 (1)

#Rh = #TxRh + #NoTxRh; donde #TxRh = #Tx-is_a + #Tx-part_of (2)

%DfNoTxRh = (#DfNoTxRh / #NoTxRh) * 100 (3)

%DfAt = (#DfAt / #At) * 100 (Nota: atributo es sinónimo de propiedad) (4)

%DfAx = (#DfAx / #Ax) * 100 (5)

La fórmula (1) se relaciona con la métrica que cuantifica el atributo "Disponibilidad de Términos Definidos" (1.1.1), (2) con "Disponibilidad de relaciones no taxonómicas balanceadas" (1.1.4.1), (3) con "Disponibilidad de relaciones no taxonómicas definidas" (1.1.4.2); (4) con "Disponibilidad de Propiedades Definidas" (1.1.2); y (5) con "Disponibilidad de Axiomas Especificados" (1.1.3). Notar que todos los atributos asociados con la "Calidad Estructural Ontológica" (1.1) tienen una métrica indirecta que da como resultado un porcentaje. Este porcentaje se utiliza por cada uno los indicadores elementales de estos atributos para calcular el valor de indicador de cada uno de ellos, el cual es también un porcentaje y es exactamente el mismo valor que arroja la métrica. Por ejemplo, si la métrica para el atributo 1.1.1 da como resultado 50%, entonces su indicador elemental dará también como resultado 50%, y este último valor se interpretará usando los mismos tres niveles de aceptabilidad que se muestran en la Figura 3-12.

Una vez diseñadas todas las métricas e indicadores, se concluye la subactividad Diseñar el análisis de la RSL. Más detalles sobre las métricas e indicadores elementales asociados a los atributos de las características "Calidad de Cobertura Terminológica específica del Dominio" (1.2) y "Adherencia a otros Vocabularios" (1.3) se pueden encontrar en la Sección 4.3.3.3 para la actividad de medición y evaluación de TestTDO.

La siguiente actividad a realizar es Implementar el análisis de la RSL según la Figura 3-3. Utilizando los artefactos "Formularios con los datos extraídos" y "Especificación del análisis de la RSL", se produce la "Síntesis de datos". Recordar que, en el FF8 del artefacto "Plantilla del formulario de extracción de datos" (denominado conceptos específicos utilizados para describir el dominio de las pruebas de software, Tabla 3-4) se indica el formato en el que se extraen los términos, propiedades, relaciones y axiomas. Esta manera de extraer los datos fue muy útil para calcular las métricas de atributos

relacionados con la característica "Calidad estructural ontológica" (1.1). La Tabla 3-11 contiene los valores obtenidos para las métricas directas e indirectas.

Por otro lado, se utilizaron los datos extraídos del FF10 de la "Plantilla del formulario de extracción de datos" (denominado terminologías o vocabularios tomados en cuenta para desarrollar la ontología propuesta) para medir los atributos relacionados con la característica "Adherencia a otros vocabularios" (1.3) de cada ontología seleccionada. Con respecto a la característica "Calidad de Cobertura Terminológica específica del Dominio" (1.2), se midieron sus atributos analizando los términos extraídos en el FF8. Debido a que no es una tarea sencilla medir los atributos relacionados con la cobertura terminológica de manera objetiva con solo mirar los nombres y definiciones de los términos, también se analizó el texto completo de cada uno de los documentos seleccionados para que la medición sea más precisa. Por ejemplo, la ontología ROoST (Ferreira de Souza et al., 2017) no tiene términos específicos para pruebas dinámicas o pruebas funcionales, pero al analizar el texto completo los autores afirman que "*las pruebas de software consisten en una verificación y validación dinámica del comportamiento de un programa frente al comportamiento esperado…*" y, por lo tanto, usando esta información se consideró que ROoST cubre términos para pruebas dinámicas y funcionales. La Tabla 3-12 muestra todas estas medidas obtenidas para cada ontología seleccionada. Por ejemplo, la ontología con PID 383 (Vasanthapriyan, Tian, Zhao, et al., 2017) no adhiere su vocabulario a otras ontologías fundacionales, de dominio o centrales, pero sí a 2 glosarios de estándares internacionales diferentes. Además, esta ontología tiene términos relacionados con pruebas estáticas, dinámicas y funcionales, pero ninguno se relaciona con pruebas no funcionales.

Tabla 3-11: Medidas para Términos, Atributos (o Propiedades), Relaciones y Axiomas. Tenga en cuenta que Df significa *Defined*; Tr *Term*; Rh *Relationship*; At Attribute; Ax Axiom; Tx *Taxonomic*; NoTx *No Taxonomic*; y CptuaLvl *Conceptual Level*. Tabla tomada de (Tebes et al., 2020).

PID	Terms (Tr)			Attributes (At)			Relationships (Rh)								Axioms (Ax)		
	#Tr	#DfTr	%DfTr	#At	#DfAt	%DfAt	#Rh	#TxRh	#Tx-is a	#Tx-part of	#NoTxRh	#DfNoTxRh	%DfNoTxRh	%TxCptuaLvl	#Ax	#DfAx	%DfAx
347	14	14	100%	0	-	-	16	15	14	1	1	1	100%	93.75%	0	-	-
383	50	7	14%	57	0	0%	65	48	30	18	17	0	0%	73.8%	12	6	50%
457	45	7	15.6%	4	0	0%	46	37	27	10	9	0	0%	80.4%	12	6	50%
468	45	37	82.2%	4	0	0%	51	27	18	9	24	13	54%	52.9%	9	9	100%
1003	54	54	100%	4	0	0%	131	108	94	14	23	23	100%	82.4%	0	-	-
346	14	0	0%	0	-	-	15	7	5	2	8	0	0%	46.7%	0	-	-
424	42	0	0%	0	-	-	41	41	41	0	0	0	-	100%	0	-	-
118	36	0	0%	0	-	-	50	40	18	22	10	0	0%	80%	0	-	-
366	37	0	0%	0	-	-	40	40	40	0	0	0	-	100%	0	-	-
359	15	1	6.7%	0	-	-	20	15	2	13	5	0	0%	75%	0	-	-
397	52	16	30.8%	0	-	-	50	40	37	3	10	0	0%	80%	0	-	-
1000	63	19	30.2%	11	8	72.7%	74	67	44	23	7	7	100%	90.5%	6	6	100%

Una vez que se obtuvieron todas las medidas se utilizaron los indicadores elementales para interpretar y calcular los valores de los indicadores asociados a cada atributo. Luego, utilizando estos valores y el modelo de agregación aditivo lineal, se calcularon los valores de los indicadores para cada característica del árbol de NFR como se muestra en la Tabla 3-13. Este modelo de agregación consiste en asignar un peso a cada atributo (y subcaracterística, si la hubiera) para calcular el valor del indicador de la característica de alto nivel como un promedio ponderado de los valores de los indicadores de cada atributo/ subcaracterística asociada. Tenga en cuenta que los pesos (segunda columna en la Tabla

3-13), que representan la importancia relativa de los elementos en el árbol de NFR, se acordaron en el grupo de investigación GIDIS_Web por consenso de acuerdo con el objetivo declarado. Se asignó un peso de 0,5 a la subcaracterística "Calidad Estructural Ontológica" (1.1); 0,2 a la subcaracterística "Calidad de Cobertura Terminológica específica del Dominio" (1.2); y 0,3 a la subcaracterística "Adherencia a otros vocabularios" (1.3). Se consideró que la "Calidad Estructural Ontológica" (1.1) es la característica de mayor peso ya que sus atributos están relacionados con aquellos elementos que debe tener toda ontología conceptualizada, es decir, términos, relaciones taxonómicas y no taxonómicas además de propiedades y axiomas. Notar también que podemos medir sus atributos de una manera más objetiva. Otra característica relevante es la "Calidad de Cobertura Terminológica específica del Dominio" (1.2). Sin embargo, es difícil realizar la medición de sus atributos de manera objetiva. Por esta razón, se decidió que esta característica tiene el peso más bajo.

Tabla 3-12: Medidas para atributos relacionados con las características "Adherencia a otros Vocabularios" y "Calidad de Cobertura Terminológica específica del Dominio". Tenga en cuenta que "Y" significa Sí, "N" No, y # cantidad de. Tabla tomada de (Tebes et al., 2020).

PID	Compliance to other Vocabularies			Domain Terminological Coverage			
	#International Standard Glossaries	Foundational Ontology	#Domain or Core Ontologies	Static Testing Terms	Dynamic Testing Terms	Functional Testing Terms	Non-Functional Testing Terms
347	0	N	1	N	Y	Y	N
383	2	N	0	Y	Y	Y	N
457	2	N	0	Y	Y	Y	Y
468	5	Y	2	N	Y	Y	N
1003	1	N	2	Y	Y	Y	Y
346	3	N	0	N	Y	N	Y
424	1	N	0	Y	Y	Y	Y
118	1	N	0	N	Y	Y	N
366	2	N	0	N	Y	Y	N
359	1	N	0	N	Y	Y	N
397	2	N	2	N	Y	Y	N
1000	0	N	0	N	Y	Y	N

Una vez concluida la evaluación y obtenidos los resultados, se elaboró la "Síntesis de Datos". Este artefacto cuenta con un resumen de cada ontología y, además, características como el tipo de ontología, calidad ontológica, entre otras. Además, el artefacto "Síntesis de datos" contiene las respuestas a las RQs. A continuación, se discuten las tres RQs establecidas considerando los resultados de este análisis.

Tabla 3-13. Resultados de la evaluación para todas las ontologías seleccionadas. El círculo verde indica el nivel de aceptabilidad " satisfactorio" (●); el rombo naranja indica el nivel " marginal" (◆) y el cuadrado rojo " insatisfactorio" (■). Los valores de los indicadores están en [%]. Tabla tomada de (Tebes et al., 2020).

Continua

Characteristics & *Attributes* / Ontologies (**PIDs**)	**Weig**	**347**	**383**	**457**	**468**	**1003**	**346**
1. Ontological Quality (root)	-	40.05■	46.62■	51.28■	79.54◆	66.71◆	36.41■
1.1 Ontological Structural Quality	**0.50**	55■	33.24■	32.56■	79.08◆	61.92◆	22.81■
1.1.1 Defined Terms Availability	**0.40**	100●	14■	15.6■	82.2◆	100●	0■
1.1.2 Defined Properties Availability	**0.10**	0■	0■	0■	0■	0■	0■
1.1.3 Specified Axioms Availability	**0.20**	0■	50■	50■	100●	0■	0■
1.1.4 Balanced Relationships Availability	**0.30**	50■	59■	54■	87●	73◆	76◆
1.1.4.1 Balanced Non-Taxonomic Relationships Availability	**0.80**	37.50■	73.50◆	68◆	95.65●	66.34◆	95.05●
1.1.4.2 Defined Non-Taxonomic Relationships Availability	**0.20**	100●	0■	0■	54■	100●	0■
1.2 Domain-specific Terminological Coverage Quality	**0.20**	50■	75◆	100●	50■	100●	50■
1.2.1 Static Testing Terms Availability	**0.25**	0■	100●	100●	0■	100●	0■
1.2.2 Dynamic Testing Terms Availability	**0.25**	100●	100●	100●	100●	100●	100●
1.2.3 Functional Testing Terms Availability	**0.25**	100●	100●	100●	100●	100●	0■
1.2.4 Non-Functional Testing Terms Availability	**0.25**	0■	0■	100●	0■	100●	100●
1.3 Compliance to other Vocabularies	**0.30**	8.50■	50■	50■	100●	52.50■	50■
1.3.1 Terminological Compliance to International Standard Glossaries	**0.50**	0■	100●	100●	100●	85●	100●
1.3.2 Terminological Compliance to other Domain/Core Ontologies	**0.10**	85●	0■	0■	100●	100●	0■
1.3.3 Terminological Compliance to Foundational Ontologies	**0.40**	0■	0■	0■	100●	0■	0■

Characteristics & *Attributes* / Ontologies (**PIDs**)	**Weights**	**424**	**118**	**366**	**359**	**397**	**1000**
1. Ontological Quality (root)	-	32.75 ■	30.95 ■	25 ■	32.79 ■	42.36 ■	39.52 ■
1.1 Ontological Structural Quality	**0.50**	0 ■	16.40 ■	0 ■	20.08 ■	28.72 ■	59.03 ■
1.1.1 Defined Terms Availability	**0.40**	0 ■	0 ■	0 ■	6.7 ■	30.8 ■	30.2 ■
1.1.2 Defined Properties Availability	**0.10**	0 ■	0 ■	0 ■	0 ■	0 ■	72.7 ◆
1.1.3 Specified Axioms Availability	**0.20**	0 ■	0 ■	0 ■	0 ■	0 ■	100 ●
1.1.4 Balanced Relationships Availability	**0.30**	0 ■	55 ■	0 ■	58 ■	55 ■	66 ◆
1.1.4.1 Balanced Non-Taxonomic Relationships Availability	**0.80**	0 ■	68.34 ◆	0 ■	72.50 ◆	68.34 ◆	57 ■
1.1.4.2 Defined Non-Taxonomic Relationships Availability	**0.20**	0 ■	0 ■	0 ■	0 ■	0 ■	100 ●
1.2 Domain-specific Terminological Coverage Quality	**0.20**	100 ●	50 ■	50 ■	50 ■	50 ■	50 ■
1.2.1 Static Testing Terms Availability	**0.25**	100 ●	0 ■	0 ■	0 ■	0 ■	0 ■
1.2.2 Dynamic Testing Terms Availability	**0.25**	100 ●	100 ●	100 ●	100 ●	100 ●	100 ●
1.2.3 Functional Testing Terms Availability	**0.25**	100 ●	100 ●	100 ●	100 ●	100 ●	100 ●
1.2.4 Non-Functional Testing Terms Availability	**0.25**	100 ●	0 ■	0 ■	0 ■	0 ■	0 ■
1.3 Compliance to other Vocabularies	**0.30**	42.50 ■	42.50 ■	50 ■	42,50 ■	60 ◆	0 ■
1.3.1 Terminological Compliance to International Standard Glossaries	**0.50**	85 ●	85 ●	100 ●	85 ●	100 ●	0 ■
1.3.2 Terminological Compliance to other Domain/Core Ontologies	**0.10**	0 ■	0 ■	0 ■	0 ■	100 ●	0 ■
1.3.3 Terminological Compliance to Foundational Ontologies	**0.40**	0 ■	0 ■	0 ■	0 ■	0 ■	0 ■

RQ1. ***¿Cuáles son las ontologías conceptualizadas para el dominio de las pruebas de software?***

Las 12 ontologías conceptualizadas seleccionadas están documentadas en (Arnicans et al., 2013; Asman & Srikanth, 2015; Bai et al., 2008; Barbosa et al., 2008; Cai et al., 2009; de S. Campos Junior et al., 2017; Ferreira de Souza et al., 2017; Freitas & Vieira, 2014; Sapna & Mohanty, 2011; Vasanthapriyan, Tian, Zhao, et al., 2017; Vasanthapriyan, Tian, & Xiang, 2017; Zhu & Huo, 2005). La Tabla 3-9 resume algunos datos extraídos de los formularios de estos 12 estudios primarios. Los artículos (Arnicans et al., 2013; Bai et al., 2008; Barbosa et al., 2008; Cai et al., 2009; Sapna & Mohanty, 2011; Zhu & Huo, 2005) fueron identificados en la RSL realizada en (Ferreira de Souza et al., 2013) en el 2013. Los restantes (Asman & Srikanth, 2015; de S. Campos Junior et al., 2017; Ferreira de Souza et al., 2017; Freitas & Vieira, 2014; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017) se desarrollaron después del 2013. Además, como se indica en (Ferreira de Souza et al., 2013), los autores desarrollaron ROoST (Ferreira de Souza et al., 2017) (PID 468). Observe que la ontología de Vasanthapriyan *et al.* aparece dos veces (a saber, PID 383 (Vasanthapriyan, Tian, Zhao, et al., 2017) y PID 457 (Vasanthapriyan, Tian, & Xiang, 2017)). Si bien ambos estudios tienen 35 términos en común, (Vasanthapriyan, Tian, Zhao, et al., 2017) introduce 15 términos no considerados en (Vasanthapriyan, Tian, & Xiang, 2017), mientras que (Vasanthapriyan, Tian, & Xiang, 2017) tiene 10 términos nuevos que no están incluidos en (Vasanthapriyan, Tian, Zhao, et al., 2017). Además, (Vasanthapriyan, Tian, Zhao, et al., 2017) aporta 57 atributos mientras que (Vasanthapriyan, Tian, & Xiang, 2017) solo aporta 4 atributos. También hay dos nuevos coautores en (Vasanthapriyan, Tian, Zhao, et al., 2017). Por lo tanto, se decidió incluir ambos estudios primarios debido a que EC5 y EC7 (criterios de exclusión en Tabla 3-3) no se aplican adecuadamente para esta situación. A continuación, se realiza un resumen de cada ontología y luego se ilustran aspectos relacionados con su calidad ontológica en relación con los fundamentos introducidos anteriormente.

En Campos *et al.* (de S. Campos Junior et al., 2017) los autores proponen una arquitectura basada en el uso de una ontología de dominio denominada RTE-Ontology (*Regression Tests Execution*) y un modelo de ontología de procedencia (modelo PROV-O), utilizada para capturar y proporcionar datos de pruebas de regresión para respaldar la mejora continua de los procesos de pruebas de software. Esta ontología solo cubre pocos términos (14 en total) relacionados con actividades de prueba dinámica, y sus artefactos y entorno. Además, RTE-Ontology se utilizó para intermediar la entrada y salida de datos en la arquitectura, traduciendo y descubriendo información utilizando datos sin procesar. En resumen, el objetivo principal de (de S. Campos Junior et al., 2017) es proponer un enfoque capaz de capturar y proporcionar información sobre ejecuciones pasadas de pruebas de regresión.

En Vasanthapriyan *et al.* (Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017), una ontología de pruebas de software es diseñada para representar el conocimiento de las pruebas de software necesario dentro del contexto de los agentes de software. Usando esta ontología los autores desarrollan un marco de conocimiento basado en ontología para ayudar a los *testers* en las empresas de software a administrar el conocimiento de las pruebas de software. Esta ontología tiene términos relacionados con actividades de prueba, artefactos, entornos y técnicas (o métodos). Aunque ambos estudios comparten términos comunes, (Vasanthapriyan, Tian, Zhao, et al., 2017) tiene más términos relacionados con las pruebas estáticas (por ejemplo, *walkthrough, inspection, technical review, informal review*). En cambio,

(Vasanthapriyan, Tian, & Xiang, 2017) agrega términos para prueba funcionales y no funcionales, entre otros.

(Ferreira de Souza et al., 2017) construyeron una ontología de dominio de pruebas llamada ROoST (*Reference Ontology on Software Testing*), que tiene como objetivo definir un vocabulario compartido con respecto al dominio de pruebas que se utilizará en iniciativas de gestión del conocimiento. Además, ROoST fue desarrollada para establecer una conceptualización común sobre el dominio de las pruebas de software, centrándose en el proceso de prueba para apoyar la comunicación entre las partes interesadas involucradas en dicho proceso. Esta ontología tiene una buena cobertura en términos de actividades de prueba, artefactos, técnicas, agentes, entornos y niveles. Pero los términos solo están relacionados con pruebas dinámicas y funcionales, por lo que no se mencionan términos de pruebas estáticas y/o no funcionales.

Asman *et al.* (Asman & Srikanth, 2015) presentan una ontología de pruebas de software de dominio superior que contiene conocimientos generales para dicho dominio. Desarrollaron esta ontología para que sirviera como base para el desarrollo de nuevas ontologías de dominio de nivel inferior y poder vincular las existentes. La ontología documentada en (Asman & Srikanth, 2015) tiene muchos términos relacionados con agentes de prueba, técnicas de prueba estáticas, técnicas de diseño de prueba, artefactos, herramientas, defectos, niveles de prueba, entre otros (también incluye actividades de prueba solo para revisiones formales). Por otro lado, Freitas *et al.* (Freitas & Vieira, 2014) presentan una ontología denominada PTOntology (*Performance Testing*) y la comparan con otras relacionadas. Esta ontología cubre el dominio de las pruebas de rendimiento del software considerando términos relacionados con actividades, roles, artefactos y herramientas. Además, se exploran tecnologías semánticas para demostrar la viabilidad práctica de desarrollar aplicaciones basadas en ontologías para ayudar a los *testers* con la planificación y gestión de las pruebas de rendimiento.

Arnicans *et al.* (Arnicans et al., 2013) proponen una metodología para obtener de forma semiautomática una ontología liviana para el dominio de las pruebas de software basada en el glosario creado por ISTQB (ISTQB, 2021b). Sin embargo, solo muestran una taxonomía de técnicas de prueba en lugar de una ontología liviana. Afirman que las ontologías livianas incluyen conceptos, propiedades, y relaciones taxonómicas y no taxonómicas entre conceptos. Sin embargo, los autores solo consideran conceptos y relaciones taxonómicas en su modelo conceptual, mientras que las propiedades y las relaciones no taxonómicas no están incluidas.

Sapna *et al.* (Sapna & Mohanty, 2011) ilustran un marco para pruebas de caja negra en el cual las especificaciones se capturan usando UML. Este marco utiliza una ontología para dar soporte a la gestión de las pruebas de software. En particular, la ontología se usa para representar y administrar casos de uso y escenarios, y luego enumerar escenarios de prueba para producir un conjunto de casos de pruebas. La ontología presentada en Sapna *et al.* (Sapna & Mohanty, 2011) cubre niveles de prueba, artefactos y defectos.

Cai *et al.* (Cai et al., 2009) presentan un método de construcción de ontologías de pruebas de software basadas en SWEBOK (Bourque & Fairley, 2014), así como una ontología de clasificación de pruebas de software basada en el modelo de calidad de software ISO 9126 (ISO, 2001). Al igual que Arnicans *et al.* (Arnicans et al., 2013), la ontología documentada es más bien una taxonomía de técnicas de prueba. Por otro lado, Bai *et al.* (Bai et al., 2008) desarrolló una ontología de prueba de dominio llamada TOM (*Test Ontology Model*) que es compatible con U2TP (OMG, 2019). Además, enriquece la semántica del modelo U2TP con propiedades de clase y restricciones utilizando

información ontológica. TOM tiene términos principalmente relacionados con artefactos y considera un rol de prueba llamado *test arbiter*, que evalúa los resultados de la prueba.

Barbosa *et al.* (Barbosa et al., 2008) desarrollaron OntoTest cuyo objetivo es respaldar la adquisición, organización, reutilización e intercambio del conocimiento en el dominio de las pruebas de software. Esta ontología tiene una ontología de pruebas de software principal y 5 sub-ontologías que abordan conceptos específicos de la misma, a saber: proceso de prueba; artefactos de prueba; pasos de prueba; estrategias y procedimientos de prueba; y recursos de prueba. Desafortunadamente, (Barbosa et al., 2008) solo documenta la conceptualización de la ontología de pruebas de software principal y la sub-ontología de recursos de prueba de forma gráfica (usando UML), pero no lo hace para las otras sub-ontologías relacionadas aunque si las describen en el texto. Además, la sub-ontología de recursos de prueba es más bien una taxonomía.

Finalmente, Zhu *et al.* (Zhu & Huo, 2005) diseñaron una arquitectura multi agente para entornos de desarrollo y mantenimiento de software. Además, desarrollaron un prototipo de sistema para probar aplicaciones basadas en la web que utiliza una ontología de pruebas de software llamada STOWS, la cual facilita las comunicaciones entre agentes de software y entre agentes y desarrolladores/*testers* humanos. Esta ontología divide los términos de prueba en conceptos básicos y compuestos. Los conceptos básicos son *testers*, contexto, actividades, métodos, artefactos y entorno. En cambio, los conceptos compuestos son tareas de prueba y la capacidad del agente. Este estudio primario seleccionado (Zhu & Huo, 2005) documenta más bien taxonomías de ambos tipos de conceptos (es decir, básicos y compuestos), aunque incluyen algunas relaciones no taxonómicas.

Con respecto a los resultados de la calidad ontológica de las ontologías conceptualizadas seleccionadas, los cuales se presentan en la Tabla 3-13, se observaron oportunidades de mejora en todas ellas en diferentes grados. Tenga en cuenta que utilizando los niveles de aceptabilidad de los indicadores elementales (consulte las descripciones de los criterios de decisión en la Figura 3-12), se puede proponer acciones de cambio relacionadas con cada atributo que ha alcanzado un nivel insatisfactorio (■) o marginal (◆). A continuación, se ilustran algunos ejemplos relacionados con estos resultados de calidad ontológica.

Según la Tabla 3-13, los dos mejores puntajes para la subcaracterística Calidad Estructural Ontológica (1.1) fueron obtenidos por ROoST (Ferreira de Souza et al., 2017) (79,08% - nivel marginal ◆) y Asman *et al.* (Asman & Srikanth, 2015) (61,92% ◆). Aunque (Asman & Srikanth, 2015) tiene 100% (nivel satisfactorio ●) de disponibilidad de términos definidos contra ROoST que tiene 82,2% (◆), ROoST tiene mejor calidad estructural ya que tiene todos los axiomas especificados (100% ●, es decir, tiene 9 axiomas en total según Tabla 3-11) contra (Asman & Srikanth, 2015) que no tiene (0% - nivel insatisfactorio ■). Además, Asman *et al.* tiene una disponibilidad de relaciones no taxonómicas definidas del 100% (●) contra ROoST que solo alcanza el 54% (■). Sin embargo, ROoST tiene un mejor balance entre la cantidad de relaciones taxonómicas y no taxonómicas (95,65% ●) que (Asman & Srikanth, 2015) (66,34% ◆). Mirando la Tabla 3-11, el lector puede evaluar la diferencia en las medidas ya que ROoST tiene 27 relaciones taxonómicas y 24 no taxonómicas (es decir, está bien balanceada la cantidad de ambas), mientras que (Asman & Srikanth, 2015) tiene 108 y 23 respectivamente. Finalmente, ambas ontologías tienen cuatro atributos pero sin sus definiciones, por lo tanto, para la disponibilidad de propiedades definidas ambas han alcanzado el 0% ■.

En cuanto a los resultados de la subcaracterística Calidad Estructural Ontológica (1.1) de las otras ontologías con puntuaciones más bajas, ninguna alcanza ni siquiera el nivel marginal ◆. En general, este problema se debe a que la mayoría de las ontologías tienen un nivel insatisfactorio (■) o un nivel marginal (◆) en la disponibilidad de términos y propiedades definidas, disponibilidad de axiomas especificados y/o disponibilidad de relaciones balanceadas. Notar que la única ontología que tiene disponibilidad de propiedades definidas es STOWS (Zhu & Huo, 2005) (que alcanza el 72,7% ◆ teniendo 8 propiedades definidas de 11, según la Tabla 3-11). Además, solo 3 de las 10 ontologías restantes seleccionadas tienen axiomas especificados (Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017; Zhu & Huo, 2005). STOWS tiene 6 axiomas especificados (100% ●). Las ontologías de Vasanthapriyan *et al.* (Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017) especificaron el 50% (■), es decir, 6 especificados de 12 axiomas en total. Teniendo en cuenta el atributo de disponibilidad de términos definidos, solo RTE-Ontology (de S. Campos Junior et al., 2017) alcanza el nivel satisfactorio (●) con un 100%. Las 9 ontologías restantes (Arnicans et al., 2013; Bai et al., 2008; Barbosa et al., 2008; Cai et al., 2009; Freitas & Vieira, 2014; Sapna & Mohanty, 2011; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017; Zhu & Huo, 2005) tienen un nivel insatisfactorio (■) que oscila entre 0 y 30% aproximadamente. Esto significa que el 75% (9 de 12) de las ontologías seleccionadas no consideran documentar explícitamente las definiciones de la mayoría de sus términos.

Con respecto a la subcaracterística Disponibilidad de Relaciones Balanceadas (1.1.4) en las 10 ontologías restantes (es decir, sin incluir a ROoST y a Asman *et al.* debido a que se han analizado anteriormente), solo PTOntology y STOWS (Freitas & Vieira, 2014; Zhu & Huo, 2005) han alcanzado el nivel marginal (◆) mientras que las demás solo el nivel insatisfactorio (■). Esta debilidad se debe a que la mayoría de las ontologías no tienen un buen equilibrio entre la cantidad de relaciones taxonómicas y no taxonómicas y/o tampoco han definido las relaciones no taxonómicas. Tenga en cuenta que solo RTE-Ontology y STOWS (de S. Campos Junior et al., 2017; Zhu & Huo, 2005) tienen relaciones no taxonómicas definidas (100% ●). Sin embargo, observando la Tabla 3-11, se puede afirmar que estas 2 ontologías no tienen un buen balance entre relaciones taxonómicas y no taxonómicas, ya que RTE-Ontology (de S. Campos Junior et al., 2017) tiene 15 relaciones taxonómicas y 1 no taxonómica (37,50% ■), y STOWS (Zhu & Huo, 2005) tiene 67 taxonómicas y 7 no taxonómica (57% ■).

Notar que, las otras 8 ontologías restantes (sin incluir a RTE-Ontology, STOWS, ROoST y tampoco a Asman *et al.*) tienen 0% (■) de disponibilidad de relaciones no taxonómicas definidas (1.1.4.2). Aunque PTOntology (Freitas & Vieira, 2014) tiene un 0% (■) del atributo 1.1.4.2, tiene un 76% (◆) de la subcaracterística Disponibilidad de Relaciones Balanceadas (1.1.4) debido a la contribución del atributo 1.1.4.1 que obtuvo un 95,05% (●). Por otro lado, las ontologías documentadas en (Bai et al., 2008; Barbosa et al., 2008; Sapna & Mohanty, 2011; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017) están moderadamente equilibradas entre la cantidad de relaciones taxonómicas y no taxonómicas (◆) ya que su valor indicador oscila entre el 68 y el 73% aproximadamente. En este punto es importante señalar que las ontologías de Arnicans *et al.* y Cai *et al.* (Arnicans et al., 2013; Cai et al., 2009) tienen 0% (■) para este atributo. Esto se debe a que estas ontologías tienen relaciones totalmente taxonómicas (%TxCptuaLvl = 100 en la Tabla 3-11). Por lo tanto, se puede afirmar que son taxonomías más que ontologías, tal como se mencionó anteriormente. En ambos casos, todas las relaciones son “tipo de” (o relación “es un”).

En el resumen anterior de cada ontología seleccionada se mencionó brevemente aspectos de la cobertura de las ontologías a nivel de proceso, es decir, teniendo en cuenta conceptos de actividades de pruebas de software, roles, agentes, recursos, técnicas o métodos, y otros términos como niveles y entornos de prueba. A continuación, se describe la cobertura de las ontologías en cuanto a términos relacionados con pruebas dinámicas, estáticas, funcionales y/o no funcionales.

Observando la Tabla 3-13, solo 3 ontologías alcanzan el 100% (●) en la subcaracterística de Calidad de Cobertura Terminológica específica del Dominio (1.2), a saber: Asman *et al.*, Vasanthapriyan *et al.* y Arnicans *et al.* (Arnicans et al., 2013; Asman & Srikanth, 2015; Vasanthapriyan, Tian, & Xiang, 2017). Esto significa que estas ontologías tienen al menos un término relacionado con pruebas estáticas, pruebas dinámicas, pruebas funcionales y pruebas no funcionales. Aunque Vasanthapriyan *et al.* (Vasanthapriyan, Tian, & Xiang, 2017) tiene 100% (●), la ontología presentada en Vasanthapriyan *et al.* (Vasanthapriyan, Tian, Zhao, et al., 2017) ha alcanzado el 75% (◆). Esta diferencia se debe a que (Vasanthapriyan, Tian, & Xiang, 2017) tiene en cuenta el término "pruebas no funcionales" y (Vasanthapriyan, Tian, Zhao, et al., 2017) no. El resto de ontologías solo alcanzan el 50% (■) de cobertura.

Es importante indicar que todas las ontologías tienen 100% (●) en los atributos de disponibilidad de términos de pruebas funcionales y/o no funcionales (códigos 1.2.3 y 1.2.4 en la Tabla 3-13, respectivamente). Se esperaba obtener este resultado ya que las pruebas de software implican un proceso que verifica/valida el cumplimiento de una entidad frente a los requisitos funcionales y no funcionales (ISO, 2013a; ISTQB, 2021b). Sin embargo, solo 4 ontologías (Arnicans et al., 2013; Asman & Srikanth, 2015; Freitas & Vieira, 2014; Vasanthapriyan, Tian, & Xiang, 2017) cubren términos relacionados con pruebas no funcionales y casi todas tienen disponibilidad de términos de pruebas funcionales (excepto PTOntology (Freitas & Vieira, 2014) que es una ontología para pruebas de rendimiento, las cuales son un tipo de pruebas no funcionales).

La verificación/validación a través de las pruebas de software se puede realizar de forma dinámica o estática (ISO, 2013a; ISTQB, 2021b). La diferencia es que la primera implica la ejecución del código de la entidad a probar mientras que la segunda no. Sin embargo, solo unas pocas ontologías (Arnicans et al., 2013; Asman & Srikanth, 2015; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017) consideran las pruebas estáticas y todas las ontologías tienen disponibilidad de términos de pruebas dinámicas, como se muestra en la Tabla 3-12.

Con respecto a la subcaracterística Adherencia a otros Vocabularios (1.3), casi todas las ontologías adhieren su terminología a uno o más glosarios de estándares internacionales. Las únicas que no especifican el uso de estos glosarios son RTE-Ontology y STOWS (de S. Campos Junior et al., 2017; Zhu & Huo, 2005). Por otro lado, solo unas pocas ontologías tienen en cuenta la terminología de otras ontologías centrales o de dominio, a saber: Asman *et al.*, ROoST, RTE-Ontology y OntoTest (Asman & Srikanth, 2015; Barbosa et al., 2008; de S. Campos Junior et al., 2017; Ferreira de Souza et al., 2017). Finalmente, ROoST (Ferreira de Souza et al., 2017) es la única que se construyó sobre una ontología fundacional (llamada UFO). Por lo tanto, ROoST es la única ontología que alcanza el 100% (●) para la subcaracterística Adherencia a otros Vocabularios (1.3). La segunda ontología mejor clasificada es OntoTest (Barbosa et al., 2008) con un 60% (◆). Los restantes tienen puntuaciones entre 0 y 52,50% (■). En la séptima columna de la Tabla 3-9 se muestran las terminologías específicas utilizadas por cada ontología.

Considerando el análisis anterior, las ontologías mejor clasificadas son ROoST (79,54% ◆) y Asman *et al.* (66,71% ■) con respecto a la calidad ontológica (la raíz del árbol de NFR). Los demás tienen puntuaciones que van del 25 al 51,28% (■).

Otras características importantes a destacar son si la ontología está modularizada y si se utilizó algún método para su desarrollo. Las únicas ontologías modularizadas son Asman *et al.*, ROoST, OntoTest y STOWS. La ontología de dominio superior presentada en Asman *et al.* se divide en 11 sub-ontologías. Por otro lado, OntoTest adoptó un enfoque en capas con 2 niveles, a saber, el nivel de ontología y el nivel de sub-ontología. Por su parte, ROoST está dividida en 4 sub-ontologías o módulos. Finalmente, STOWS se divide en conceptos básicos y compuestos, donde los conceptos compuestos se definen a partir de conceptos básicos.

Por otro lado, en cuanto a la metodología de desarrollo utilizada para construir la conceptualización de la ontología, 4 de 12 ontologías no especifican ninguna (ver columna 6 en la Tabla 3-9). Estos son: OntoTest, RTE-Ontology, TOM, y STOWS.

RQ2. ***¿Cuáles son los conceptos incluidos más frecuentes, sus relaciones, atributos y axiomas necesarios para describir el dominio de las pruebas de software?***

Con respecto a la frecuencia de los términos incluidos, la Figura 3-13 muestra la nube de palabras de los términos registrados tomados de los 12 formularios de extracción de datos. 'Test Case' es la palabra más utilizada con 7 ocurrencias. Luego, sigue 'Test Plan' (6 ocurrencias), 'Test Result' (5) y otros cuatro términos con 4 ocurrencias, a saber: 'Testing Activity', 'Testing Artifact', 'Test Execution' y 'Thing'. Los términos con 3 ocurrencias son 'Test Design Technique', 'Integration Testing', 'Structural Testing', 'Test Case Design', 'System Testing', 'Test Technique', 'Test Planning', 'Test Script', 'Test Suite' y 'Tester'. Finalmente, se obtuvieron 52 palabras con 2 ocurrencias y 239 con solo 1. De ahora en adelante, se indicará entre paréntesis el número de frecuencia para cada término.

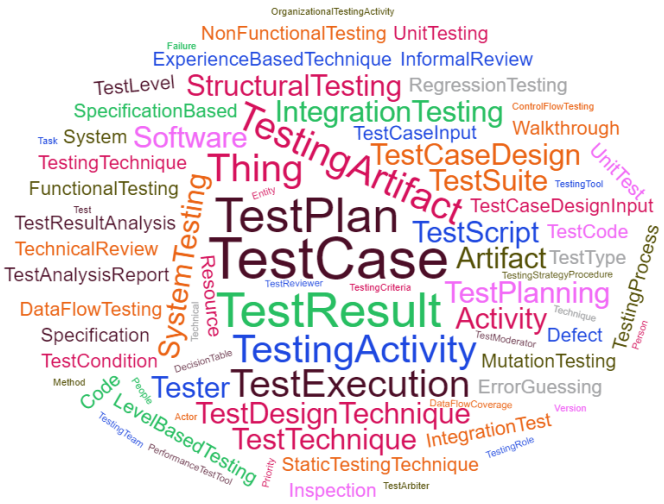


Figura 3-13: Nube de palabras (https://www.nubedepalabras.es/) de los términos registrados de los 12 formularios de extracción de datos. Tenga en cuenta que los nombres de los atributos (propiedades) y las relaciones no están incluidos. También tenga en cuenta que el tamaño de la palabra está relacionado con la frecuencia del término, y los colores del término no tienen significado. Figura tomada de (Tebes et al., 2020).

El lector puede acceder a la fuente de la nube de palabras en https://bit.ly/SLR-TestingOntologies y observar el número de ocurrencias por cada término. Tenga en cuenta que para la ontología de Vasanthapriyan *et al.* la cual aparece dos veces (ver PID

383 y PID 457 en la Tabla 3-11), no se consideraron los términos repetidos para el cálculo de frecuencia. Por ejemplo, se puede ver en la Figura 3-13 los términos 'Integration Testing' (3) e 'Integration Test' (2). ¿Tienen ambos términos la misma semántica? Comparando las definiciones existentes difieren ligeramente en la semántica. Mirando el primer término, en OntoTest es una fase de prueba, mientras que en ROoST tiene una semántica de prueba basada en nivel que es un tipo de actividad. Por otro lado, en Asman *et al.* 'Integration Testing' tiene una semántica de estrategia de prueba que describe objetivos principales, objetos típicos de pruebas, agentes involucrados, entre otros. En cambio, el término 'Integration Test' en STOWS es un tipo de contexto, con semántica de etapa pero no de actividad, mientras que en Sapna *et al.* es un tipo de prueba. La misma brecha semántica ocurre con los términos 'System Testing' (3) y 'System Test' (2).

Al igual que 'Integration/System Testing' e 'Integration/System Test', se encuentran los términos 'Unit Test' (2), 'Unit Testing' (2) y 'Component Testing' (2). Tenga en cuenta que este último no aparece en la nube de palabras por razones de espacio, ya que solo muestra un número limitado de palabras, pero en el enlace dado el lector puede verificar que 'Component Testing' tiene 2 ocurrencias. Una vez más, el término 'Unit Test' es un tipo de contexto en STOWS, mientras que es un tipo de prueba en Sapna *et al*. Por otro lado, el término 'Unit Testing' es una prueba basada en niveles en ROoST y una fase de prueba en OntoTest. En Asman *et al.* el término 'Component Testing' es sinónimo de 'Unit Testing' y es una estrategia de prueba en el mismo sentido que se describió anteriormente para 'Integration Testing'. En cambio, en Cai *et al.*, 'Component Testing' es una técnica de prueba basada en la generación de casos de prueba.

Para los términos relacionados con las técnicas de prueba se encontraron muchas variantes. Algunas de ellas son 'Test Design Technique' (3), 'Test Technique' (3), 'Testing Technique' (2), 'Method' (1), y 'Testing Method' (1). Todas ellas son técnicas/métodos destinados a diseñar casos de prueba, excepto por el término 'Testing Technique' de (Barbosa et al., 2008) que es una técnica para establecer criterios de prueba a adoptar (por ejemplo, flujo de control, flujo de datos, análisis de mutaciones). Además, otros sinónimos de técnicas de diseño de pruebas se presentan en (Arnicans et al., 2013), a saber: 'Test Case Design Technique' (1) y 'Test Specification Technique' (1). Por otro lado, Asman *et al.* define 'Test Method' (1) como tipos de objetivos de prueba y, por lo tanto, no está relacionado con técnica/método para diseñar casos de prueba. Finalmente, otro término relacionado con las técnicas de prueba es 'Test Execution Technique' (1) presentado en (Arnicans et al., 2013), que no está relacionado con el diseño de los casos de prueba sino con su ejecución o realización.

Otras técnicas de diseño más específicas son las técnicas de diseño de pruebas de caja blanca, basadas en la experiencia y de caja negra. Se identificaron muchos sinónimos para técnicas de caja blanca y caja negra. Por ejemplo, algunos sinónimos de las técnicas de caja blanca son: 'White Box Testing' (1), 'White Box Test Design Technique' (1), 'Structural Test Design Technique' (1), 'Structure-based Test Design Technique', (1) 'White Box Technique' (1), 'White-box Testing Technique' (1), entre otros. El término más frecuente relacionado con las técnicas de caja blanca es 'Structural Testing' (3). Sin embargo, en Asman *et al*. tiene semántica de un tipo de objetivo en lugar de una técnica de prueba y en Vasanthapriyan *et al.* (Vasanthapriyan, Tian, & Xiang, 2017) no está definido. Algo muy similar sucede con términos para técnicas de diseño de caja negra, solo que todos los sinónimos tienen una frecuencia de 1. En cuanto a técnicas basadas en la experiencia, algunos sinónimos encontrados son: 'Tester Experience Based' (1), 'Experience Based Test Design Technique' (1) y 'Experience Based Technique' (2).

Otros términos analizados fueron los relacionados con las pruebas funcionales y no funcionales. Se incluyeron términos específicos para 'Functional Testing' (2) y 'Non-Functional Testing' (2) en (Asman & Srikanth, 2015; Vasanthapriyan, Tian, & Xiang, 2017). Sin embargo, en (Vasanthapriyan, Tian, & Xiang, 2017) estos términos no están definidos, mientras que en (Asman & Srikanth, 2015) ambos términos tienen la semántica de 'Test Method' (es decir, es un tipo de objetivo de prueba según (Asman & Srikanth, 2015)). Además, términos de pruebas funcionales y no funcionales relacionados con las técnicas fueron identificados. Por ejemplo, (Arnicans et al., 2013) tiene los términos 'Functional Test Design Technique' (1) y 'Non-Functional Test Design Technique' (1), y (Barbosa et al., 2008) tiene el término 'Functional Testing Technique' (1). Por último, PTOntology que es una ontología para pruebas de rendimiento, tiene muchos términos relacionados con pruebas no funcionales, todos con una ocurrencia de 1. Algunas de ellos son 'Performance Test' (1), 'Performance Test Activity' (1), 'Performance Test Artifact' (1), entre otros.

Con respecto a los roles de prueba, el rol más frecuente es 'Tester' (3). En segundo lugar, es el gestor de prueba con 2 ocurrencias: 'Test Manager' (1) y 'Testing Manager' (1). Por último, con solo 1 ocurrencia, se encuentran: 'Test Case Designer', 'Performance Tester', 'Test Moderator', 'Test Team Member', 'Test Reviewer' y 'Test Arbiter'. En cuanto a las actividades más frecuentes involucradas en las pruebas de software se identificaron las siguientes: planificación, diseño, ejecución y análisis. Para la actividad de planificación de pruebas se hallaron los términos 'Performance Test Plan Elaboration' (1) y 'Test Planning' (3). El término más frecuente para el diseño de pruebas es 'Test Case Design' (3), pero otros términos utilizados son 'Test Design' (1) y 'Test Case Generation' (1). En cambio, para la actividad de ejecución de pruebas, el término más frecuente es 'Test Execution' (4). Otros términos relacionados con esta actividad son Test Case Execution' (1) y 'Test Suite Execution' (1). Finalmente, para la actividad de análisis de pruebas se encontraron 2 términos diferentes, a saber: 'Test Result Analysis' (2) y 'Test Analysis and Measurement' (1).

Teniendo en cuenta artefactos de prueba, los más frecuentes son 'Test Case' (7), 'Test Plan' (6), 'Test Result' (5), 'Test Suite' (3), y 'Test Script' (3). Otros artefactos citados con menos frecuencia son: 'Test Analysis Report' (2), 'Test Case Design Input' (2) (este término se relaciona con el concepto de 'Test Basis' de la ISO 29119-1), 'Test Code' (2), 'Driver' (2), 'Stub' (2), 'Test Condition' (2), 'Test Schedule' (2), 'Testing Incident Report' (1) y 'Error Report' (1). Es importante señalar que algunos de estos artefactos están relacionados con las pruebas dinámicas (por ejemplo, 'Test Code', 'Test Script', entre otros). Además, algunas ontologías establecen que la actividad de planificar las pruebas produce un 'Test Plan', el diseño produce 'Test Cases', la ejecución produce 'Test Results', y el análisis genera un 'Test Analysis Report'.

Pocas ontologías consideran el término 'Defect' (2). Además, en (Asman & Srikanth, 2015) se definen dos tipos de defectos, a saber: 'Failure' (1) y 'Bug' (1). Por otro lado, muchas ontologías tienen un término que se refiere al objeto a probar. Algunas de ellas lo llaman 'Code To Be Tested' (2), 'System Under Test Concept' (1), 'Artifact Under Testing' (1), 'Subject Under Test' (1), 'Object Under Test' (1), y 'Test Object' (2). En cuanto a los términos relacionados con herramientas de prueba que soportan actividades de prueba, pocas ontologías los consideran. Entre ellos se encuentran 'Tool for Testing' (2) y los siguientes, todos con una única ocurrencia: 'Testing Tool', 'Tool Support for Management of Testing', 'Tool Support For Test Execution', 'Test Specification Tool', 'Performance Testing Tool', y 'Performance Test Tool'.

Con respecto a términos para el entorno (o contexto) en las pruebas, se hallaron los términos 'Testing Environment' (2), 'Test Environment' (2) and 'Environment' (1). En resumen, teniendo en cuenta todos los términos analizados anteriormente se puede concluir que, o bien las ontologías no tienen términos definidos o, si lo están, la semántica de un mismo término puede diferir en cierta medida entre diferentes ontologías.

Por otro lado, en cuanto a las relaciones identificadas, se encontraron 599 en total, y la frecuencia para #Tx-is_a = 370 (relaciones taxonómicas es_un), #Tx-part_of = 115 (relaciones taxonómicas parte-todo), y #NoTxRh = 114 (relaciones no taxonómicas), que representan aproximadamente el 62%, 19% y 19%, respectivamente. Analizando atributos (o propiedades) de las ontologías que los contienen (Asman & Srikanth, 2015; Ferreira de Souza et al., 2017; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017; Zhu & Huo, 2005), se notó una alta heterogeneidad entre los mismos ya que las propiedades son muy específicas de cada una de ellas. Al igual que las propiedades, ocurrió lo mismo para los axiomas ya que también son muy específicos de cada ontología.

RQ3. *¿Cómo se clasifican las ontologías de pruebas de software existentes?*

Con respecto a esta RQ, se identificó la clasificación de cada ontología dada por los autores, a saber, ontología fundacional (o de nivel superior), de dominio superior, de dominio o de aplicación/instancia. La Tabla 3-9 resume esta información en la tercera columna. Tenga en cuenta que 5 de las 12 ontologías conceptualizadas (Arnicans et al., 2013; Barbosa et al., 2008; Cai et al., 2009; Sapna & Mohanty, 2011; Zhu & Huo, 2005) no especificaron ninguna categoría. El resto son ontologías de dominio (Bai et al., 2008; de S. Campos Junior et al., 2017; Ferreira de Souza et al., 2017; Freitas & Vieira, 2014; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017) y solo Asman *et al.* la denominó ontología de dominio superior, justificando en (Asman & Srikanth, 2015) esta clasificación. Como se mencionó anteriormente, ROoST (Ferreira de Souza et al., 2017) se basa en una ontología fundacional llamada UFO. Desde el punto de vista del autor de la presente tesis, (Arnicans et al., 2013; Barbosa et al., 2008; Cai et al., 2009; Sapna & Mohanty, 2011; Zhu & Huo, 2005) también son ontologías de dominio.

3.3.4 Discusión

Dentro de esta subsección, en 3.3.4.1 se discuten los principales hallazgos de la RSL realizada. Por otro lado, en 3.3.4.2 se justifica por qué se considera a este estudio una revisión sistemática y no un mapeo. Finalmente, se discuten algunos trabajos relacionados en 3.3.4.3.

3.3.4.1 Principales Hallazgos de la RSL

Considerando los resultados de la evaluación de la Tabla 3-13, ROoST (Ferreira de Souza et al., 2017) es la ontología de *testing* más formalmente rigurosa que se encontró, alcanzando el valor de indicador global más alto (79,54 % 🔸). Es la única ontología seleccionada para el dominio de pruebas de software que se basa en una ontología fundacional. Está bien modularizada y equilibrada en cuanto a la cantidad de relaciones taxonómicas y no taxonómicas (95,65% 🟢). Sin embargo, considerando los resultados de la evaluación realizada, esta ontología también tiene algunas limitaciones. Como se muestra en la Tabla 3-13, ROoST alcanza el 82,2 % (🔸) para el atributo de disponibilidad de términos definidos. Esto significa que ROoST no presenta todos los términos definidos explícitamente en (Ferreira de Souza et al., 2017). Además, no presenta todas las

definiciones de términos de una manera fácil de identificar y visualizar, por ejemplo, mediante el uso de una tabla o una lista. Las definiciones de los diferentes términos se encontraron embebidos en el texto de los párrafos del articulo citado, dificultando la recopilación de la medida del atributo "Disponibilidad de Términos Definidos" (1.1.1). Al presentar las definiciones de esta manera existe el riesgo de introducir un sesgo en la medición por parte del Recolector de Datos, ya que, en algunas situaciones que se presentaron, no se sabía exactamente si el texto representaba una breve descripción del término o su definición formal. Por lo tanto, algunas mejoras que ROoST puede considerar son agregar las definiciones ausentes de algunos términos para alcanzar el 100% para el atributo (1.1.1) además de empaquetar todas las definiciones en una tabla.

Otras limitaciones de ROoST relacionadas con la calidad estructural ontológica son la ausencia de definiciones de relaciones no taxonómicas (54% ■) y propiedades (0% ■), además de tener muy pocas de estas últimas. Por lo tanto, se podría recomendar aumentar el número de propiedades definidas (solo hay 4 y no están definidas) y agregar las definiciones ausentes de algunas relaciones no taxonómicas. En este punto, vale la pena mencionar que un beneficio de usar el árbol de NFRs presentado en la Tabla 3-10, que tiene diferentes niveles de granularidad considerando atributos y subcaracterísticas en conjunto con los resultados de su evaluación (Tabla 3-13), es que las fortalezas y las debilidades pueden ser identificadas objetivamente. Por lo tanto, se pueden recomendar acciones de mejora para los indicadores que no dan resultados satisfactorios.

Otro problema detectado es la cobertura terminológica del dominio de prueba. Aunque ROoST tiene una buena cobertura de términos que incluyen actividades de prueba, técnicas, artefactos, entorno y agentes, solo considera términos relacionados con pruebas dinámicas y funcionales (50% ■). ROoST se basa en SWEBOK, que no incluye términos de prueba estáticas. Sin embargo, ROoST también se basa en los estándares ISO 29119 e ISTQB, que sí consideran términos de pruebas estáticas y no funcionales.

Una observación destacable considerando la Tabla 3-13 es que la mayoría de las ontologías adhieren sus vocabularios a estándares internacionales. En ese momento (2018), los estándares internacionales de testing de software más utilizados son ISTQB (5 ontologías lo utilizan), IEEE 829 (5, ya que 2 ontologías utilizan el 829-2008 (IEEE, 2008b) y 3 el 829-1998 (IEEE, 1998)), y SWEBOK (Bourque & Fairley, 2014) (4). Además, las ontologías (Arnicans et al., 2013; Asman & Srikanth, 2015; Vasanthapriyan, Tian, & Xiang, 2017; Vasanthapriyan, Tian, Zhao, et al., 2017) que usan el estándar ISTQB consideran términos de prueba estáticas excepto ROoST (Ferreira de Souza et al., 2017) que también considera el estándar SWEBOK. También se observa que hay un desacuerdo entre algunos estándares como SWEBOK, que considera que las pruebas incluyen solo actividades dinámicas y, por lo tanto, las pruebas estáticas son llamadas "análisis estático" ya que no pertenecen al dominio de las pruebas de software. En cambio, otros estándares como ISO 29119 e ISTQB consideran que el "análisis estático" es parte del dominio de las pruebas de software estáticas.

Con respecto a la reutilización de otras ontologías, muy pocas utilizaron otras ontologías de *testing*. Entre las ontologías de pruebas de software reutilizadas se identificaron OntoTest, STOWS y SwTOI (Barbosa et al., 2008; Bezerra et al., 2009; Zhu & Huo, 2005). No se seleccionó la última ontología para este estudio secundario ya que es solo una implementación y no presenta una conceptualización del dominio de las pruebas de software (recordar el criterio de inclusión 3 en la Tabla 3-3). Además, se recuperó una tesis de maestría del primer autor de (Bezerra et al., 2009) utilizando el

método *backward snowballing* que presenta la conceptualización de SwTOI, pero se tuvo que excluir ya que está escrita en portugués y satisface el criterio de exclusión 3.

Otro tema a destacar es que, para documentar una conceptualización explícita para cualquier ontología de dominio en general y para documentar la complejidad inherente del dominio de las pruebas de software en particular, la cantidad de páginas del manuscrito debería ser un indicador importante a tener en cuenta (ver valores en la quinta columna de la Tabla 3-9). Una conceptualización formal y explícita no solo debe transmitir la estructura ontológica utilizando diagramas bien formados, sino también agrupar en tablas o listas la semántica (definiciones explícitas) de términos, propiedades, axiomas y relaciones no taxonómicas. Desafortunadamente, casi ninguna ontología del conjunto seleccionado utiliza tablas o listas para comunicar las definiciones. Pero lo que es peor, muy frecuentemente se descuidan las definiciones de estos elementos ontológicos, como se puede verificar en la Tabla 3-13. Además, cuando las definiciones están disponibles, en general se incluyen embebidas en el texto principal del manuscrito tal como se comentó anteriormente para ROoST. Desde el punto de vista de la recolección y medición de datos, el hecho de que las definiciones no estén bien agrupadas hace que estas tareas sean más propensas a errores. En última instancia, una recomendación sencilla para contribuir a la claridad y "belleza" (d'Aquin & Gangemi, 2011) de las ontologías es "presentar las definiciones ontológicas de conceptos en tablas (o listas)".

Teniendo en cuenta que d'Aquin *et al.* (d'Aquin & Gangemi, 2011) considera la modularidad de una ontología como una buena práctica, podríamos haber agregado un atributo llamado "disponibilidad de estructura modular" perteneciente con la subcaracterística de Calidad Estructural Ontológica (1.1) en el árbol representado en la Tabla 3-10. Solo se encontraron 4 ontologías que son modulares, a saber: Asman *et al.*, ROoST, OntoTest y STOWS (Asman & Srikanth, 2015; Barbosa et al., 2008; Ferreira de Souza et al., 2017; Zhu & Huo, 2005). Al agregar este atributo y luego de realizar la evaluación, estas ontologías podrían alcanzar un mejor porcentaje en la calidad estructural ontológica. Incluso si no lo incluimos en esta evaluación, el lector puede apreciar la flexibilidad que tiene el árbol de NFRs para agregar o eliminar atributos cuando sea necesario.

3.3.4.2 ¿Por qué este Estudio es una RSL y no un Mapeo Sistemático?

Otro tema importante a abordar son las diferencias entre RSL y mapeos sistemáticos (MS), y por qué se considera que este trabajo sigue más bien el enfoque de una RSL. Según Kitchenham *et al.* (B. Kitchenham & Charters, 2007) una RSL se define como "*una forma de estudio secundario que utiliza una metodología bien definida para identificar, analizar e interpretar toda la evidencia disponible relacionada con una pregunta de investigación específica de una manera imparcial y (hasta cierto punto) repetible*". En cambio, para los mismos autores, un MS es *"una revisión amplia de estudios primarios en un área temática específica que tiene como objetivo identificar qué evidencia está disponible sobre el tema*". Ambos enfoques producen estudios secundarios, como el trabajo actual, ya que también se revisaron estudios primarios. La Tabla 3-14 resume las similitudes y diferencias entre ambos tipos de estudios secundario según Napoleão *et al.* (Napoleão et al., 2017), que han considerado las bases provistas en (B. A. Kitchenham et al., 2015; Petersen et al., 2015).

Desde un punto de vista teórico, el enfoque de RSL sugiere que los estudios primarios a recuperar y seleccionar deben basarse en evidencia empírica (B. Kitchenham & Charters, 2007). Esto puede implicar que se debe incluir un criterio de selección para

descartar aquellos estudios primarios que no se basen en evidencia empírica. Para el presente trabajo, ¿qué significa que una ontología conceptualizada se base en evidencia empírica? La respuesta a este caso particular es si el artefacto (conceptualización de la ontología) ha sido verificado y/o validado. Si se hubiera incluido un criterio de selección para este aspecto, solo 6 estudios primarios de 12 habrían sido seleccionados debido a que los autores de las ontologías mencionan el uso de algún método de verificación y/o validación. Las únicas ontologías de pruebas de software que mencionan algún método de evaluación utilizado son Asman *et al.*, ROoST, Vasanthapriyan *et al.*, PTOntology y Arnicans *et al*.

Tabla 3-14: Comparación de las características entre RSLs y MSs (tabla tomada de (Napoleão et al., 2017)).

Features	SLRs	SMs
Focus of the review	Identify, analyze and interpret all available evidence related to a specific RQ	Identify and classify what evidence is available (broad review) in a specific topic of an area
	Identify best practices based on empirical evidence	Establish the state of evidence
Research Questions	Narrow RQs	Broader RQs
	Specific RQs	Multiple RQs
Methods for searching	Search string highly focused	Search string less highly focused
Methods for selecting	Generally few studies are considered	A large number of studies are considered (broad coverage)
	The studies are evaluated in details	The studies are not evaluated in details
Methods for data extraction	The primary studies are assessed regarding their quality (the main goal is to establish the state of evidence)	The primary studies are not assessed regarding their quality
	Include data extraction procedures. It is a time-consuming task.	Much broader (classification and categorization stage). It is not a time-consuming task
Synthesis	Include depth analysis techniques such as meta-analysis and narrative synthesis	Include no-depth analysis techniques such as totals and summaries
Dissemination of the results	Higher importance for practitioners (relevant to industry)	May be more limited, the aim is to influence the future of the research in a specific topic

Considerar este criterio de selección o no fue una decisión crítica a tomar ya que, según los autores en (Lam & Kennedy, 2005), si los criterios de selección para una RSL son demasiado amplios se pueden incluir estudios de mala calidad, lo que reduce la confianza en el resultado final. Pero si los criterios son demasiado estrictos, los resultados se basan en menos estudios y es posible que la evidencia obtenida no sea generalizable. Por lo tanto, si se consideraba incluir solo estudios con validación/verificación de la conceptualización ontológica, los resultados se habrían basado en pocos estudios primarios (6 en total) y se podrían haber perdido otros aspectos interesantes a analizar como, por ejemplo, la cobertura terminológica.

Además, en (Napoleão et al., 2017) los resultados muestran que tanto las RSLs como los MSs tienen RQs genéricas, cadenas de búsqueda amplias y ambos adoptan la búsqueda automática como estrategia de búsqueda. Sin embargo, la evaluación de la calidad de los estudios primarios se ha adoptado más ampliamente en RSLs que en MSs. Por lo tanto, concluyeron que en la práctica, solo la evaluación de la calidad se realiza de manera diferente entre ambos enfoques. Teniendo en cuenta las características de la Tabla 3-14 y los hallazgos prácticos de (Napoleão et al., 2017), se describe a continuación por qué se considera que el trabajo actual es más bien una RSL.

Con respecto a la característica "*Focus of the review*", se identificó, analizó e interpretó toda la evidencia disponible relacionada con las ontologías de pruebas de

software conceptualizadas. Se ha incluido estudios primarios que no se basan obligatoriamente en evidencia empírica, es decir, si la ontología conceptualizada está validada/verificada o no. Además, se considera que las 3 RQs establecidas en este estudio son RQs específicas, que están limitados a ontologías conceptualizadas para el dominio de *testing*.

Con respecto a la característica "*Methods for selecting*", se han seleccionado pocos estudios primarios ya que se han limitado los criterios de selección solo a ontologías conceptualizadas. Se han recuperado 796 documentos en total (tras realizar las subactividades Aplicar los Criterios de Selección y Agregar los Estudios Relevantes No Recuperados) y se han seleccionado solo 12, lo que representa aproximadamente el 1,5% de la población. Si se hubieran incluido solo estudios con evidencia empírica, se habrían seleccionado alrededor del 0,75%.

Teniendo en cuenta la característica "*Methods for data extraction*", se ha seguido un riguroso procedimiento de extracción de datos para obtener todos los datos de las ontologías conceptualizadas seleccionadas, a saber: términos, propiedades, relaciones y axiomas. Para la "*Synthesis*", se ha realizado un análisis cuantitativo utilizando métricas e indicadores para evaluar las subcaracterísticas Calidad Estructural Ontológica (1.1), Calidad de Cobertura Terminológica específica del Dominio (1.2) y Adherencia a otros Vocabularios (1.3) de las ontologías seleccionadas (Tabla 3-13). Además, se ha utilizado una nube de palabras para contar y analizar la frecuencia de los términos ontológicos. Por último, se ha incluido una síntesis narrativa que contiene el objetivo y alcance de cada ontología.

Finalmente, con respecto a la característica "*Dissemination of the results*", se ha publicado esta RSL inicialmente en la conferencia QUATIC'19 (Tebes et al., 2019), y luego se ha ampliado en un artículo de revista (Tebes et al., 2020) con el objetivo de llegar a una audiencia más amplia de investigadores y profesionales.

3.3.4.3 Trabajos Relacionados

Desde lo mejor del conocimiento de los autores de (Tebes et al., 2020), hasta marzo de 2019 había solo dos estudios secundarios realizados previamente para ontologías de pruebas de software. El primer estudio secundario realizado en 2013 está documentado en (Ferreira de Souza et al., 2013), mientras que el segundo realizado en 2015 está documentado en (Asman & Srikanth, 2015). Además, en 2016 se publicó un estudio terciario de Garousi *et al.* (Garousi & Mäntylä, 2016) titulado "*A systematic literature review of literature reviews in software testing*", cuyo objetivo fue clasificar sistemáticamente los estudios secundarios en pruebas de software. En la Tabla 15 de (Garousi & Mäntylä, 2016), para la subcategoría de mapeo "*Testing-related terminology*", los autores indican que "no esperaban encontrar estudios sobre este tema". Así, tanto (Ferreira de Souza et al., 2013) como (Asman & Srikanth, 2015) no fueron incluidos en (Garousi & Mäntylä, 2016), seguramente debido a los criterios de selección de inclusión/exclusión utilizados por Garousi *et al.*

Teniendo en cuenta el objetivo declarado para esta RSL, se trató de capturar ontologías de dominio de pruebas de software que documenten una conceptualización para este dominio. Por el contrario, (Ferreira de Souza et al., 2013) incluye ontologías tanto conceptualizadas como implementadas. Establecieron dos RQ, a saber: RQ1) ¿Cuál es la cobertura del dominio de pruebas de software en las ontologías existentes sobre este dominio?; y RQ2) ¿Cómo se desarrollaron? Sus hallazgos para las conceptualizaciones

incluidas están en línea con los hallazgos del presente trabajo para los siguientes mismos 6 estudios primarios: (Arnicans et al., 2013; Bai et al., 2008; Barbosa et al., 2008; Cai et al., 2009; Sapna & Mohanty, 2011; Zhu & Huo, 2005). Sin embargo, se ha ampliado el análisis cualitativo realizado por Ferreira de Souza *et al.* mediante la realización de un análisis cuantitativo utilizando los resultados de métricas e indicadores, así como un modelo de puntuación de agregación para interpretar el nivel de satisfacción alcanzado por las características y subcaracterísticas. Además, se consideraron otras subcaracterísticas y atributos de calidad, como la disponibilidad de términos/propiedades/axiomas/relaciones no taxonómicas definidas, entre otros, para la actividad del análisis. La evaluación cuantitativa aquí realizada ha permitido confirmar no solo sus hallazgos, sino también ampliar el análisis mejorando la comprensión actual de la calidad ontológica de las ontologías seleccionadas. Además, el proceso de RSL que se ha seguido aquí fomenta un enfoque más sistemático, reproducible y auditable.

En cuanto a Asman *et al.* (Asman & Srikanth, 2015), su objetivo principal era construir una ontología de pruebas de software de dominio superior. En este contexto, utilizaron el enfoque de RSL como metodología de investigación para obtener evidencia previa sobre otras ontologías existentes para este dominio. En particular, consideran dos RQs, a saber: RQ1) ¿Cuáles son las ontologías o marcos de pruebas de software existentes, su propósito y su evaluación?; y RQ2) ¿Cuáles son los conceptos, relaciones y restricciones relevantes que se necesitan para describir el conocimiento de las pruebas de software en general? Los autores encontraron 8 ontologías de pruebas de software, la mayoría de ellas incluidas en el presente trabajo, mientras que 2 de ellas fueron excluidas debido a los criterios de selección de inclusión/exclusión establecidos. Sin embargo, en (Asman & Srikanth, 2015) falta una documentación rigurosa del proceso de RSL seguido (es decir, actividades y artefactos), lo que representa una amenaza para la repetibilidad y auditabilidad de su estudio.

3.4 Conclusiones

En esta sección se describen los principales hallazgos de este capítulo. Por un lado, con respecto al proceso para RSL y mapeos sistemáticos especificado en la Sección 3.2, el mismo brinda un flujo recomendado para llevar a cabo estos tipos de estudios secundarios ya que se tiene consciencia de que en la instanciación particular de un proceso pueden existir algunos puntos de variación (por ejemplo, la paralelización de algunas tareas). Además, con respecto a los beneficios del modelado de procesos mencionados en la Sección 3.2.1, las especificaciones de procesos para RSL propuestas tienen como objetivo principal facilitar la comprensión y la comunicación, así como brindar apoyo a la mejora y gestión de procesos. Sin embargo, una discusión exhaustiva y una ilustración detallada del modelado de procesos para admitir completamente la automatización de procesos de RSL han quedado fuera del alcance de este trabajo.

Además, se destacaron los beneficios y fortalezas del modelo de proceso de RSL propuesto en comparación con otros ya existentes, concluyendo que estos últimos tienen una especificación menos robusta. Una contribución importante es la inclusión de la actividad de prueba piloto, la cual promueve la validación del artefacto “Protocolo de la RSL” no solo en la actividad A1 sino también en la subactividad A2.1. Realizar un estudio piloto (no para un estudio replicado) puede ser muy útil para mejorar el Protocolo de la RSL y fomentar en cierta medida la calidad de los resultados basados en la evidencia. Dado que hasta ahora los autores de esta RSL han realizado muy pocos estudios piloto, no tienen la evidencia empírica suficiente para afirmar cuál es un tamaño de muestra

efectivo. Kitchenham *et al.* (B. Kitchenham & Charters, 2007) tampoco mencionan un tamaño de muestra adecuado pero indican que el mismo depende de los recursos disponibles.

Los autores de (Olsina et al., 2019) consideran que en un estudio piloto, además del tamaño de la muestra también es importante seleccionar un conjunto de documentos (o toda la muestra) y realizar la extracción de datos de cada documento por más de un recolector de datos de forma independiente. Tener más de un formulario de extracción de datos completo para el mismo documento permite verificar posibles inconsistencias en los artefactos diseñados. El Validador de la RSL y el Experto en el Dominio son roles muy importantes que deben ser desempeñados por al menos una persona con experiencia para verificar adecuadamente (junto con Recolectores de Datos independientes) los formularios de extracción de datos en busca de inconsistencias y, además, detectar oportunidades de mejora en la plantilla de metadatos para el posterior esfuerzo de análisis en la actividad A3. En pocas palabras, se considera que varias variables pueden ser importantes para hacer una prueba piloto efectiva. Sin embargo, se necesitaría analizar más evidencia para abordar este problema y, por lo tanto, es un aspecto interesante que podría ser investigado más a fondo por la comunidad.

Otra conclusión de este trabajo es que el modelo de proceso propuesto para RSL proporciona una buena base para comprender los detalles y discutir alternativas o personalizaciones de este proceso. De hecho, el rigor proporcionado por el modelado de procesos, donde se combinan varias perspectivas pero también se pueden separar de forma independiente, proporciona una mayor riqueza de expresividad.

En la Sección 3.3 se utilizó el proceso de RSL propuesto para llevar a cabo un estudio secundario sobre ontologías de pruebas de software. Se seleccionaron y analizaron un total de 12 ontologías de *testing*, y los resultados de este análisis confirman que existe ambigüedad entre los mismos conceptos de diferentes ontologías. Además, también existe incompletitud dado que muchas ontologías no cubren conceptos relacionados a pruebas estáticas, dinámicas, funcionales y/o no funcionales. Por ejemplo, los hallazgos relacionados con la RQ2 indican que existe ambigüedad e incompletitud en conceptos de artefactos, roles, actividades y métodos de prueba. Otro descubrimiento fue que no hay una ontología de *testing* directamente relacionada con conceptos de Requisitos No Funcionales y Requisitos Funcionales.

Por otro lado, analizando la cobertura terminológica de las ontologías seleccionadas, se observó que la mayoría de ellas abarcan términos relacionados con pruebas dinámicas y funcionales. Por el contrario, solo unas pocas ontologías consideran términos relacionados con pruebas estáticas y no funcionales. Los términos de pruebas estáticas y pruebas dinámicas funcionales que se encuentran entre las diferentes ontologías cubren principalmente los términos incluidos en estándares de prueba como ISTQB (ISTQB, 2021b), ISO 29119 (ISO, 2013a) y SWEBOK (Bourque & Fairley, 2014). Sin embargo, no sucede lo mismo con los términos de prueba no funcionales encontrados. Por lo tanto, se puede concluir que principalmente los glosarios ISTQB e ISO 29119 son más ricos que las ontologías de pruebas de software actuales con respecto a términos de pruebas no funcionales.

Por último, dado que no se encontró la ontología adecuada para el objetivo declarado y las ontologías seleccionadas presentaban algunas limitaciones en su calidad ontológica, se decidió desarrollar una nueva ontología la cual se documenta en el próximo capítulo.

CAPÍTULO 4: CONSTRUYENDO UNA ONTOLOGÍA DE ALTO NIVEL PARA EL DOMINIO DE LAS PRUEBAS DE SOFTWARE

4.1 Introducción

En el capítulo anterior se ilustró la realización de una RSL sobre ontologías de pruebas de software, la cual fue llevada a cabo con un objetivo en particular: "poder establecer una ontología de prueba adecuada que se puede reutilizar en una familia de estrategias de prueba". Luego de analizar las características de las 12 ontologías identificadas y evaluar su calidad ontológica, se llegó a la conclusión de que ninguna de las ontologías existentes para el dominio de las pruebas de software era adecuada para el objetivo perseguido. Por lo tanto, se decidió construir una nueva ontología para este dominio y que sea adecuada como base conceptual de una familia de estrategias de *testing*. Esta ontología recibe el nombre de TestTDO y se documenta en la Sección 4.3. Recordar que esta nueva ontología está asociada al objetivo específico 2 y contribución 4 (OE_2 y C_4, respectivamente, en Tabla 1-1).

Por otro lado, para poder construir TestTDO se tuvo que elegir una metodología de investigación rigurosa a utilizar que sirva de guía para este proceso. Dado que *Design Science Research* (DSR) es un enfoque de investigación riguroso que respalda el diseño y la construcción de artefactos útiles e innovadores para resolver problemas no triviales (A. Hevner & Chatterjee, 2010), se decidió seguir este enfoque para construir la ontología de pruebas de software. Además, teniendo en cuenta que se observó que el proceso de DSR estaba débilmente especificado, se realizó la mejora en la especificación de su proceso como un trabajo adicional relacionado. Todos los detalles respectos a la mejora de la especificación del proceso de DSR se encuentran en la siguiente Sección 4.2. También es importante recordar que este trabajo realizado se encuentra relacionado al OE_2 y es considerado como la contribución 3 (C_3 en Tabla 1-1) de esta tesis doctoral.

4.2 Metodología de Investigación Utilizada para Construir la Ontología de Pruebas de Software: Design Science Research

4.2.1 Introducción y Motivación

DSR es un enfoque de investigación que da soporte al diseño y construcción de artefactos útiles e innovadores para resolver problemas de dominio. Los problemas que se abordan con DSR son problemas no triviales (A. Hevner & Chatterjee, 2010), es decir, aquellos de difícil resolución ya sea debido a que están incompletos, no son rutinarios, son contradictorios o presentan requisitos cambiantes que, a menudo, son difíciles de relevar, entre otras causas. Recordar que, en este caso, el problema a resolver es la ausencia de una base conceptual ontológica para el dominio de *testing* que sea adecuada para el desarrollo de una familia de estrategias de *testing* y se pueda integrar en la arquitectura ontológica FCD-OntoArch.

DSR establece que el artefacto que se diseña debe ser evaluado para demostrar que no sólo resuelve el problema, sino que también lo hace de manera efectiva y eficiente, brindando utilidad al usuario (A. Hevner & Chatterjee, 2010). Un artefacto se puede definir como un objeto creado por una persona con un subsecuente sentido de uso. Esta definición enfatiza dos aspectos importantes de un artefacto: que no ocurre naturalmente

y que tiene una utilidad determinada (McKay & Marshall, 2005). De acuerdo al análisis realizado en (March & Smith, 1995) se distinguen cuatro tipos de artefactos, a saber: constructores (vocabularios y símbolos), modelos (abstracciones y representaciones), métodos (algoritmos y prácticas) y también, un artefacto puede ser una instanciación (implementaciones y prototipos). En efecto, los artefactos creados a partir de DSR incluyen, pero no están limitados a, algoritmos, metodologías, sistemas de computación y modelos. Es importante aclarar que la ontología de *testing* construida es, dependiendo el punto de vista, un artefacto de tipo constructor, modelo e instanciación. Dado que el vocabulario provisto por TestTDO va a ser utilizado para enriquecer especificaciones de métodos y procesos de una familia de estrategias de *testing*, entonces TestTDO es un constructor desde este punto de vista. A su vez, TestTDO tiene como modelo un diagrama de clases UML como representación abstracta de sus conceptos, relaciones y atributos. Además, esta ontología esta implementada en OWL, por lo que sería también una instanciación.

Para la construcción de cualquier tipo de los artefactos mencionados se llevan a cabo actividades de diseño-construcción-evaluación, que iteran tantas veces como sean necesarias antes que el artefacto sea finalmente generado y comunicado para su utilización. En el área de Sistemas de la Información se reconoce el trabajo de Peffers *et al.* (Peffers et al., 2007) como uno de los primeros en proponer el enfoque de DSR para dar soporte a las investigaciones realizadas en esa área. Sin embargo, la debilidad en la especificación de un proceso general y estandarizado que guíe las actividades principales del enfoque constituye una de las principales razones de su poca divulgación en el área (Geerts, 2011; A. R. Hevner et al., 2004; McKay & Marshall, 2005; Offermann et al., 2009; Peffers et al., 2007; Pfeffers et al., 2006). Las propuestas existentes coinciden en determinar que, para aplicar el enfoque de DSR, se debe transitar por una primera etapa de identificar el problema y establecer su relevancia. Luego se debe diseñar, construir y verificar/validar el artefacto que dé solución a ese problema para evaluar que realmente significa una mejora. Finalmente, se tienen que comunicar y difundir efectivamente los resultados del artefacto.

Un análisis de la literatura existente respecto al enfoque de DSR, realizado en el 2019, permite determinar que se encuentra débilmente especificado su proceso. Tener un proceso formalmente especificado permite repetitividad y reproducibilidad en su ejecución, determinando para eso un conjunto de actividades con sus entradas y salidas, interdependencias, entre otros aspectos. Las cuatro perspectivas (o vistas) propuestas por Curtis *et al.* (Curtis et al., 1992) para modelar un proceso (recordar que eran la vista funcional, de comportamiento, informacional y organizacional, y se describieron en la Sección 2.3) fueron útiles para fortalecer el proceso de RSL descrito en la Sección 3.2 y, por lo tanto, también pueden ser útiles para fortalecer el proceso de DSR. Además, recordemos que un proceso que no presenta sus actividades modeladas es difícil de comprender, debido a que puede generar cierta ambigüedad en las descripciones de las actividades y es difícil también de comunicar.

Debido a la debilidad detectada en la falta de un proceso robusto y formal que guíe las actividades sugeridas en el enfoque de DSR, se propone una especificación del proceso de DSR utilizando para tal fin el lenguaje SPEM (OMG, 2008) y haciendo uso de las perspectivas de modelado funcional y de comportamiento propuestas por Curtis *et al.*

A continuación, en la Sección 4.2.2, se especifica el proceso propuesto para el enfoque de DSR considerando diferentes vistas de modelado de procesos. Notar que, más detalles sobre este trabajo en particular el cual especifica un proceso para DSR se pueden

encontrar en (Tebes et al., 2020). Además, algunos trabajos relacionados a DSR se presentan en el Apéndice B.

4.2.2 Especificación del Proceso Propuesto para DSR

Considerando los tres ciclos presentados en la Figura B-1 para DSR del Apéndice B, la Figura 4-1 muestra el modelo de proceso para DSR que se propone en este trabajo, que contempla la vista funcional y de comportamiento. Las actividades que se corresponden con el ciclo de Relevancia están agrupadas en la actividad que se denomina "A1 Identificar el Problema/Solución". Por su parte, las actividades comprendidas en el ciclo de Diseño se corresponden con la actividad "A2 Diseñar y Desarrollar la Solución" y a la actividad "A3 Ejecutar Verificación y Validación (VyV)". Las actividades para el ciclo de Rigor están relacionadas con el flujo de información desde la base de conocimiento, principalmente para las actividades 1, 2 y 3 de la Figura 4-1, aunque también se relacionan con la difusión y la diseminación del conocimiento obtenido como producto de la investigación. Esto último se corresponde con la actividad "A4 Comunicar la Investigación". Las subsecciones siguientes describen cada una de estas 4 actividades. Además, la aplicación de estas actividades se va a ilustrar en la Sección 4.3 para la construcción de TestTDO.

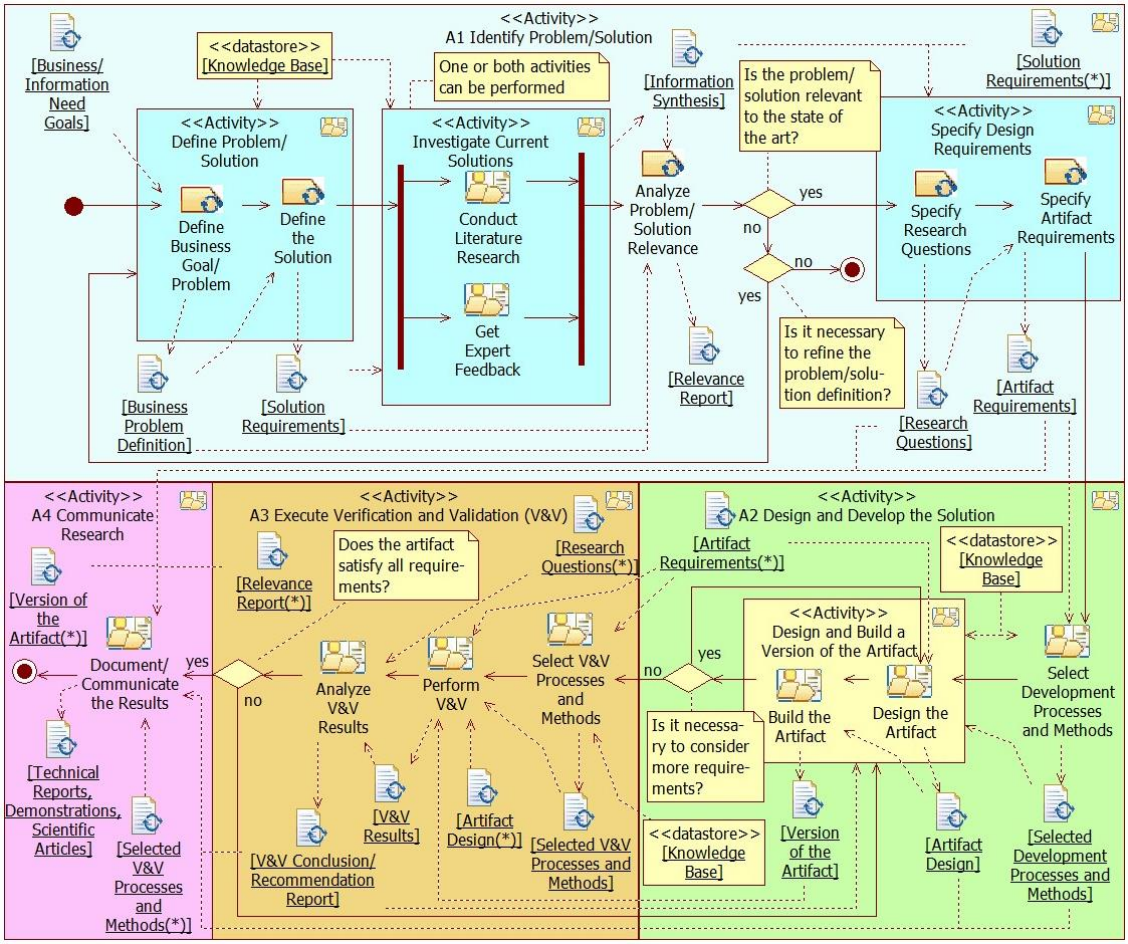


Figura 4-1: Perspectiva funcional y de comportamiento del proceso propuesto para DSR, especificado en SPEM. Notar que aquellos nombres de artefactos terminados en "(*)" indican que están replicados para mejorar la legibilidad del modelo. Figura tomada de (Tebes et al., 2020).

4.2.2.1 A1 Identificar el Problema/Solución

De acuerdo a la Figura 4-1, la primera actividad "A1 Identificar el Problema/Solución" comprende las siguientes subactividades: i) Definir el Problema/Solución, ii) Investigar soluciones actuales y iii) Especificar requisitos de diseño. Para la primera subactividad ("Definir el Problema/Solución") se tienen las tareas de "Definir el objetivo/problema de negocio" y "Definir la solución". Para definir el problema se requiere de las metas de necesidad de información / metas de negocio organizacionales, es decir, del conocimiento que tiene la organización sobre problemas y/u objetivos a alcanzar. La salida producida por esta tarea es el artefacto "Definición del problema de negocio".

Con el problema especificado se procede a definir la solución del mismo. Para esto, se requiere de la base de conocimiento para indagar sobre otras soluciones a problemas similares como así también cualquier documentación que ayude a establecer los requisitos de la solución. Luego, la segunda subactividad en A1 se denomina "Investigar soluciones actuales" y consiste en otras dos subactividades: "Realizar investigación de la literatura" y "Obtener retroalimentación con expertos". Se puede llevar a cabo solo una de estas dos actividades o ambas en paralelo. Las entradas requeridas para realizarlas son la base de conocimiento y los "Requisitos de la solución". Lo fundamental es hacer un buen análisis del estado del arte de la situación actual respecto a la solución planteada llevando a cabo, por ejemplo, revisiones sistemáticas de literatura / mapeos sistemáticos y, en caso de ser necesario, consultar con expertos del dominio. Como salida de "Investigar soluciones actuales" se genera el documento "Síntesis de información".

Siguiendo con el proceso de la Figura 4-1, se debe ejecutar la tarea "Analizar la relevancia del problema/solución". La misma requiere como entrada los documentos de "Síntesis de información", "Requisitos de la solución" y "Definición del problema de negocio". Con todo lo documentado se genera el "Informe de relevancia" el cual expondrá la importancia del problema/solución para ser abordado mediante el enfoque de DSR. Si el problema no es relevante respecto al estado del arte, el proceso puede terminar en esta etapa o bien se puede continuar refinando la definición del problema/solución, volviendo a realizar las primeras subactividades de A1. En caso contrario, es decir, si es relevante, se continua con la subactividad "Especificar requisitos de diseño". Para esto, primero se realiza la tarea de "Especificar preguntas de investigación" y luego la tarea de "Especificar requisitos del artefacto". Para llevar a cabo estas dos tareas se requiere como entradas la "Síntesis de información" y los "Requisitos de la solución". Se genera como salida los documentos denominados "Preguntas de investigación" y "Requisitos del artefacto". Las preguntas de investigación servirán posteriormente para evaluar el artefacto construido. La clara formulación de las mismas representa una actividad fundamental en cualquier estudio de investigación debido a que direccionan la investigación y transmiten su esencia (Thuan et al., 2019). En este punto finaliza A1.

4.2.2.2 A2 Diseñar y Desarrollar la Solución

De acuerdo a la Figura 4-1 que ilustra el proceso propuesto para DSR, la primera actividad que da inicio al diseño y construcción de la solución se denomina "Seleccionar procesos y métodos de desarrollo". Tiene como entrada el documento de "Requisitos del artefacto" y la base de conocimiento. Como salida se genera un artefacto con la especificación de los procesos y métodos de desarrollo seleccionados.

A continuación, sigue la subactividad "Diseñar y construir una versión del Artefacto" que se compone de las subactividades "Diseñar el artefacto" y "Construir el artefacto".

Cabe destacar que para ambas actividades se requiere del documento "Procesos y métodos de desarrollo seleccionados", como así también la base de conocimiento. Si bien en la actividad anterior se seleccionan procesos y métodos desde la base de conocimiento para diseñar y desarrollar el artefacto, eventualmente se podría requerir de algún otro proceso y/o método que no se haya previsto. Ahora, para la primera de ellas ("Diseñar el artefacto"), se utilizan también los requisitos del artefacto. Se genera como salida el "Diseño del artefacto", el cual sirve como entrada para la actividad siguiente (esto es, "Construir el artefacto").

Con el diseño del artefacto se inicia su construcción, obteniendo como salida una versión del mismo. Si es necesario redefinir el diseño ya que se deben considerar más requisitos, entonces se itera otra vez a la subactividad de diseño del artefacto. Caso contrario, el proceso continúa con la actividad A3, la cual se detalla en la siguiente subsección. Cabe aclarar que, aunque no se ilustre en el proceso de la Figura 4-1 para evitar sobrecargarlo, si se volviera a realizar la actividad de diseñar el artefacto también tendría como entrada alguna versión del diseño para poder redefinirlo.

4.2.2.3 A3 Ejecutar Verificación y Validación (VyV)

El proceso continúa con la tercera actividad denominada "A3 Ejecutar Verificación y Validación (VyV)". La primera subactividad que comprende se denomina "Seleccionar procesos y métodos de VyV" que produce como salida el documento "Procesos y métodos de VyV seleccionados". Para producirlo utiliza la base de conocimiento. Ejemplos de procesos y/o métodos de VyV pueden ser métricas e indicadores específicos, técnicas de testing de caja blanca o caja negra como cobertura de sentencias o particionado de equivalencias, respectivamente, entre otros. Para verificar y validar ontologías se suelen utilizar preguntas de competencia, revisiones de expertos y/o casos de uso.

La siguiente actividad a realizar es "Ejecutar la VyV", que requiere como entrada los requisitos del artefacto, su diseño, el artefacto construido y el detalle de los procesos y/o métodos seleccionados. Es importante remarcar que se pueden verificar y/o validar tanto el artefacto como su diseño contra los requisitos. Con los procesos y/o métodos seleccionados (que pueden ser cuantitativos, con heurísticas, de cuestionario, entre otros) se lleva a cabo la verificación y validación de una versión del artefacto y se produce como resultado el informe "Resultados de VyV".

Una vez obtenidos los resultados de la VyV, en la actividad "Analizar resultados de VyV" se corroboran los mismos contra lo establecido en las preguntas de investigación, puesto que estas deben ser respondidas en la mayor totalidad posible para asegurarse de que el artefacto tiene la utilidad planteada inicialmente. Se genera un documento denominado "Informe de Conclusión/Recomendación de VyV" en el cual se determina si es necesario volver a la etapa de diseño y construcción debido a que el artefacto no satisface todos los requisitos o, por el caso contrario, se puede avanzar hacia la última etapa del proceso. De aquí la importancia de establecer las preguntas de investigación y de iterar en ellas tanto cuanto sea necesario hasta asegurarse que cubren todos los aspectos propuestos en el diseño y construcción del artefacto.

4.2.2.4 A4 Comunicar la Investigación

La última actividad del proceso es "A4 Comunicar la Investigación" (ver Figura 4-1). La subactividad que comprende se denomina "Documentar/Comunicar los Resultados" y recibe como entrada ocho artefactos, a saber: 1) el "Informe de

Conclusión/Recomendación de VyV"; 2) una versión del artefacto generado; 3) el "Informe de relevancia"; 4) las preguntas de investigación; 5) los requisitos del artefacto; 6) los procesos y métodos de VyV seleccionados; 7) el diseño del artefacto; y 8) los procesos y métodos de desarrollo seleccionados. Haciendo uso de todos ellos se produce como salida reportes técnicos, artículos científicos, tesis, entre otros, los cuales permitan documentar y comunicar la investigación realizada para el desarrollo del artefacto.

Si el impacto del artefacto generado en la comunidad científica relacionada es positivo y de interés, es decir, el artefacto es ampliamente adoptado debido a su utilidad, relevancia y carácter innovador, entonces se almacena tanto el artefacto resultante como todo el material producido y utilizado para construirlo en la base de conocimiento. De esta manera, este aporte de conocimiento queda disponible para el diseño y construcción de nuevos artefactos.

4.3 TestTDO: una Ontología de Alto Nivel para el Dominio de las Pruebas de Software

4.3.1 Introducción

La Sección 4.3 tiene como propósito documentar los resultados de aplicar el proceso de DSR, especificado en la Sección 4.2, para la construcción de una ontología de dominio de alto nivel llamada TestTDO. Notar que el domino de las pruebas de software es complejo ya que tiene una gran cantidad de métodos, procesos y estrategias específicas. Todos ellos involucran muchos conceptos de dominio específico. Por lo tanto, es valioso tener una base conceptual robusta, es decir, una ontología de pruebas de software conceptualizada que defina explícitamente y sin ambigüedades los términos, propiedades, relaciones y axiomas o restricciones.

Un beneficio de tener una ontología de pruebas adecuada es mejorar el intercambio de información relacionada con las pruebas de software entre agentes, evitando problemas de ambigüedad. Además, una característica deseable de una ontología de pruebas de software es que cubra conceptos relacionados con las pruebas estáticas y dinámicas, ya que los estándares de pruebas de software como ISO 29119-1 (ISO, 2013c) e ISTQB (ISTQB, 2021b) consideran estos tipos de pruebas. Adicionalmente, la ontología de *testing* debe estar vinculada a ontologías de requisitos no funcionales (NFRs) y requisitos funcionales (FRs) porque las estrategias de pruebas de software son útiles para verificar y validar ambos tipos de requisitos. Recordar que el objetivo perseguido es tener una ontología de pruebas de software adecuada que "nutra" (o enriquezca) terminológicamente las especificaciones de métodos y procesos de una familia de estrategias de prueba.

Como se mencionó anteriormente, se siguió el enfoque de DSR para la construcción de TestTDO. En primer lugar, para encontrar las soluciones existentes (es decir, ontologías de pruebas de software conceptualizadas) para el problema en cuestión, se llevó a cabo la RSL documentada en el Capítulo 3. Recordar que se seleccionaron 12 estudios primarios que documentan ontologías de prueba conceptualizadas y que se evaluaron desde el punto de vista de la calidad ontológica.

Como se detalló en el capítulo anterior, la mayoría de las ontologías encontradas tienen una falta de cobertura terminológica de pruebas no funcionales y pruebas estáticas. Además, las 12 ontologías recuperadas presentan oportunidades para mejorar su calidad estructural por diferentes razones, tales como: i) no tienen todos sus términos, relaciones

no taxonómicas y propiedades definidas, así como los axiomas especificados; ii) no tienen relaciones no taxonómicas o, si las tienen, no tienen relaciones taxonómicas y no taxonómicas bien equilibradas. A su vez, ninguna de ellas está directamente relacionada con conceptos de NFRs y FRs.

Dado que las ontologías de *testing* actuales no son totalmente adecuadas para el objetivo perseguido, es decir, "nutrir" terminológicamente las especificaciones de los métodos y procesos de una familia de estrategias de prueba a desarrollar, se decidió construir una nueva ontología de pruebas de software llamada TestTDO. Tenga en cuenta que se utilizó el enfoque de DSR como marco genérico para llevar a cabo este trabajo de investigación en su conjunto, es decir, desde la RSL realizada hasta la verificación, validación y comunicación de la ontología. También se ha utilizado *METHONTOLOGY* (Fernández-López et al., 1997) para construir TestTDO, pero solo en una actividad de DSR en particular.

En este punto es importante mencionar que TestTDO está situada en la arquitectura ontológica llamada FCD-OntoArch (Olsina, 2020, 2021). Es una arquitectura ontológica de 5 capas que considera los niveles fundacional, central (*core*), dominio de alto nivel (o dominio superior), dominio de bajo nivel (o dominio inferior), e instancia. En FCD-OntoArch, las ontologías del mismo nivel se pueden relacionar entre sí. Además, las ontologías de niveles inferiores pueden enriquecerse semánticamente con ontologías de niveles superiores. Por ejemplo, TestTDO a nivel de dominio superior se enriquece con conceptos de la ontología ProcessCO (Becker et al., 2022) ubicada en el nivel central o *core*. A su vez, este último se enriquece con conceptos de ThingFO (Olsina, 2021) a nivel fundacional, como se abordará más adelante.

En resumen, en esta Sección 4.3, se especifica y discuten aspectos de la conceptualización de TestTDO, es decir, sus términos, propiedades, relaciones y axiomas. A su vez, se ilustra la evaluación de su calidad ontológica y, mediante el uso de diferentes métodos de prueba de caja negra y caja blanca, se analizan aspectos de su verificación estática y dinámica. Además, se utilizó un enfoque de evaluación dirigido por agentes humanos para validar TestTDO.

El resto de la Sección 4.3 está estructurado de la siguiente manera: la Sección 4.3.2 describe brevemente la arquitectura FCD-OntoArch y algunas de sus ontologías que se relacionan con TestTDO. Por último, la Sección 4.3.3 detalla la construcción, verificación y validación de TestTDO siguiendo el proceso de DSR ilustrado anteriormente.

4.3.2 Resumen de FCD-OntoArch y algunas de sus Ontologías

Esta subsección tiene como objetivo ilustrar brevemente algunas de las ontologías que pertenecen a FCD-OntoArch (Olsina, 2020) ya que TestTDO está relacionado con ellas. Además, como se muestra en 4.3.3.3, se aplicó un método de verificación estático a TestTDO para verificar su integración en el contexto de FCD-OntoArch. Notar que el lector puede acceder a más descripciones de algunas ontologías de FCD-OntoArch en las siguientes referencias: ThingFO (Olsina, 2021), ProcessCO (Becker et al., 2022) (o su predecesor (Becker et al., 2015)), y SituationCO en https://arxiv.org/abs/2107.10083. Además, por un lado, NFRsTDO se encuentra documentada en http://dx.doi.org/10.13140/RG.2.2.34457.65129 mientras que FRsTDO se puede encontrar en http://dx.doi.org/10.13140/RG.2.2.31659.26400.

Como se comentó anteriormente, TestTDO se encuentra en la capa de dominio de alto nivel en FCD-OntoArch. Recordar que esta es una arquitectura ontológica de cinco capas,

que considera los niveles fundacional, central o *core*, dominio de alto nivel, dominio de bajo nivel e instancia. Como se muestra en la Figura 3-8, las ontologías del mismo nivel se pueden relacionar entre sí, excepto en el nivel fundacional, donde solo existe la ontología ThingFO. Las ontologías de niveles inferiores pueden enriquecerse semánticamente con ontologías de niveles superiores. Por ejemplo, TestTDO ubicado en el nivel de dominio superior se enriquece principalmente con términos, propiedades y relaciones de las ontologías SituationCO y ProcessCO ubicadas en el nivel central. A su vez, ambas se enriquecen con los conceptos de ThingFO. De ahora en adelante, en este capítulo, los términos de las ontologías se escribirán primero en inglés, comenzado con mayúscula y con cursiva, seguido de su traducción al español entre paréntesis. Luego, se usará solo su traducción al español empezando con mayúscula.

Los términos de ThingFO como *Thing* (Cosa), *Thing Category* (Categoría de Cosa) y *Assertion* (Aserción) enriquecen semánticamente los términos de los componentes de los niveles inferiores. Cosa representa un objeto particular o concreto, tangible o intangible de un mundo particular dado, pero no una categoría o clase universal, la cual está representada por el término Categoría de Cosa.

Adicionalmente, el término Aserción se define como "*positive and explicit statement that somebody makes about something concerning Things, or their categories, based on thoughts, perceptions, facts, intuitions, intentions, and/or beliefs that is conceived with an attempt at furnishing current or subsequent evidence*". Para ser valiosa, procesable y, en última instancia, útil para cualquier ciencia, una aserción debe ser en gran medida verificada y/o validada por evidencia teórica y/o empírica. Las aserciones se pueden representar y modelar utilizando lenguajes de especificación informales, semiformales o formales.

Utilizando aserciones sobre las cosas, podemos especificar aspectos de su sustancia, relaciones, estructura, comportamiento, restricciones, intención, situación, cantidad y calidad, entre otros aspectos. Por ejemplo, la conceptualización de una ontología como un artefacto (por ejemplo, TestTDO en la Figura 4-2) representa principalmente una combinación de aserciones relacionadas con la sustancia, la relación, la estructura, la intención y la situación. Los axiomas de una ontología pueden considerarse aserciones relacionadas con restricciones.

SituationCO incluye términos –algunos tomados de otros componentes centrales (*core*)- tales como *Human Agent* (Agente Humano), *Organization* (Organización), *Project* (Proyecto), *Target Entity* (Entidad Objetivo), *Context Entity* (Entidad de Contexto) con semántica de Cosa, y el término *Goal* (Meta) con semántica de Aserción. A su vez, una Meta puede ser específica o genérica. En resumen, un Agente Humano/Organización concibe/establece Metas que son operacionalizadas por Proyectos. Una Meta implica una *Situation* (Situación), que puede especificarse mediante un *Situation Model* (Modelo de Situación). Por lo tanto, un Modelo de Situación representa un *Artefact* (Artefacto) que especifica y modela Situaciones de un determinado mundo particular o genérico.

Además, una Situación puede ser particular o genérica. Una *Particular Situation* (Situación Particular) es una *Assertion on Particulars* (Aserción sobre Particulares) relacionada con la Situación que establece y especifica explícitamente la combinación de circunstancias, episodios y relaciones/eventos particulares que abarcan las Entidades Objetivo y las Entidades de Contexto que las rodean, la cual es de interés y relevante para ser representada por un Agente Humano/Organización con una *Specific Goal* (Meta Específica) establecida. Dependiendo del propósito de la Meta Específica, las Entidades

Objetivo pueden ser, por ejemplo, *Developable Entity* (Entidad Desarrollable, por ejemplo, un documento, un código fuente, etc.), *Evaluable Entity* (Entidad Evaluable, por ejemplo, un producto de trabajo, un sistema, etc.) o *Testable Entity* (Entidad Comprobable), que tiene la semántica de Desarrollable o Evaluable en una Situación Particular dada.

La ontología ProcessCO es una ontología central (o *core*) que especifica los términos, relaciones y propiedades de los *Work Processes* (Procesos de Trabajo). ProcessCO incluye términos con semántica de Cosa como *Work Entity* (Entidad de Trabajo), que puede ser, a su vez, un Proceso de Trabajo, *Activity* (Actividad) o *Task* (Tarea). Otros términos con semántica de Cosa son *Product Entity* (Entidad de Producto), Artefacto, *Outcome* (Resultado), *Service* (Servicio), *Resource Entity* (Entidad de Recurso), *Agent* (Agente), *Method* (Método), *Strategy* (Estrategia), entre otros. Cualquier proceso bien especificado para un dominio determinado debe documentar aspectos tales como tareas y actividades específicas, artefactos, agentes, métodos, entre otros recursos. Además, cualquier método bien especificado debe documentar su procedimiento y reglas. Por lo tanto, tener una ontología que defina explícitamente estos términos y sus relaciones y propiedades es muy útil para evitar ambigüedades, inconsistencias e incompletitud en las especificaciones de un proceso de un determinado dominio de aplicación como, por ejemplo, en el dominio de *testing*. Dado que TestTDO tiene como objetivo "nutrir" o enriquecer terminológicamente las especificaciones de métodos y procesos para desarrollar una familia de estrategias de prueba, es muy importante considerar los términos de ProcessCO para enriquecer los términos de dominio de TestTDO.

Los términos Proceso de Trabajo, Actividad y Tarea son tipos de Entidades de Trabajo. El Proceso de Trabajo se compone de subprocesos o Actividades. A su vez, una Actividad está formada por subactividades o Tareas. Una Tarea es una Entidad de Trabajo atómica y detallada que no se puede descomponer. Además, involucran *Roles* (Roles) comunes, consumen Entidades de Producto, producen *Work Products* (Productos de Trabajo) y tienen *Conditions* (Condiciones), tanto previas como posteriores. Asimismo, una Entidad de Trabajo tiene asignados *Work Resources* (Recursos de Trabajo), tales como Métodos, Estrategias, Agentes, entre otros.

Otro concepto importante de ProcessCO es Producto de Trabajo, que es una Entidad de Producto. A su vez, Resultado, Artefacto y Servicio son tipos de Productos de Trabajo. Por ejemplo, Resultado se define como "*Work Product that is intangible, storable and procesable*", mientras que Artefacto "*is a tangible or intangible, versionable Work Product, which can be delivered*".

Una Entidad de Trabajo tiene una descripción (o descripción del trabajo), que especifica los pasos para lograr su objetivo. Representa "qué" debe hacerse en lugar de "cómo" debe realizarse. El "cómo" está representado por el término Método, es decir, la forma específica y particular de realizar los pasos especificados, por ejemplo, en una tarea. Tenga en cuenta que el concepto de Método tiene las propiedades de procedimiento y reglas. Un procedimiento es un conjunto ordenado de instrucciones de método u operaciones, que especifica cómo se deben realizar los pasos de una descripción de una entidad de trabajo. Mientras que una regla es un conjunto de principios, condiciones, heurísticas, axiomas, etc., asociados con el procedimiento.

Por otra parte, es importante describir algunos términos de las ontologías NFRsTDO y FRsTDO, las cuales se encuentran a nivel de dominio superior. Entre ellos tenemos los términos Entidad Evaluable y Entidad Desarrollable, ambos con semántica de Entidad Objetivo. Una Entidad Evaluable es una Entidad Objetivo a ser evaluada considerando

características y atributos relacionados con la calidad. En cambio, una Entidad Desarrollable es una Entidad Objetivo a ser desarrollada o modificada considerando sus características funcionales y capacidades.

Además, otros dos conceptos importantes de NFRsTDO y FRsTDO que tienen semántica de Aserción son, respectivamente, *Non-Functional Requirement* (Requisito No Funcional) y *Functional Requirement* (Requisito Funcional). Un Requisito No Funcional representa una restricción, es decir, una Característica o Atributo a ser evaluado sobre cómo o qué tan bien se desempeña o se desempeñará una Entidad Evaluable. En cambio, un Requisito Funcional establece lo que hace/hará la nueva/existente Entidad Desarrollable, refiriéndose a sus características funcionales y/o capacidades. Eventualmente, los Requisitos Funcionales específicos deben satisfacerse por restricciones que están representadas a través de Requisitos No Funcionales.

4.3.3 Proceso de DSR Aplicado para la Construcción de TestTDO

A continuación, se describe la ejecución de las actividades A1-A3 del proceso de DSR ilustrado en la Figura 4-1, las cuales fueron realizadas para construir TestTDO, una ontología de dominio de alto nivel para pruebas de software. En la Sección 4.3.3.1 se discute la actividad A1 Identificar el Problema/Solución, en donde se obtuvieron los requisitos del artefacto (es decir, los requisitos de TestTDO), entre otras cosas. Luego, en la Sección 4.3.3.2, se muestran los resultados de ejecutar la actividad A2 Diseñar y Desarrollar la Solución, ilustrando la conceptualización de TestTDO. Finalmente, en la Sección 4.3.3.3, se documenta la actividad A3 Ejecutar Verificación y Validación (VyV) en la cual se utilizaron varios métodos para verificar y validar TestTDO.

4.3.3.1 A1 Identificar el Problema/Solución

De acuerdo con el proceso de DSR ilustrado en la Figura 4-1, la primera actividad a realizar es Definir el Problema/Solución. A su vez, esta actividad tiene dos tareas, a saber: Definir el objetivo/problema de negocio y Definir la solución. Para definir el objetivo o problema de negocio, se requiere como entrada el artefacto Metas de necesidad de información / metas de negocio organizacionales, es decir, el conocimiento de la organización sobre el problema u objetivo a alcanzar. Para el caso de TestTDO se planteó la siguiente Meta de negocio: "el grupo de investigación GIDIS_Web requiere una conceptualización ontológica para el dominio de las pruebas de software, la cual debe ser integrada en la arquitectura FCD-OntoArch. Este artefacto debe estar relacionado con los componentes de FRsTDO y NFRsTDO". Luego, el resultado de la tarea Definir el objetivo/problema de negocio es la siguiente Definición del problema de negocio: "**obtener** una conceptualización rigurosa del dominio de *testing* **mediante** la estructuración del dominio como una ontología, **tal que** la ontología debe estar relacionada con los componentes de FRsTDO y NFRsTDO de la arquitectura FCD-OntoArch. Además, debe estar en el nivel ontológico de dominio de alto nivel. El **objetivo** es dar soporte a la especificación de una familia de estrategias integradas de pruebas que ayuden a alcanzar metas de *testing*". Notar que se utilizó la plantilla de especificación de problemas propuesta en (Wieringa, 2014) para especificar este artefacto.

La siguiente tarea es Definir la solución y, para poder definirla, es necesario en primer lugar indagar sobre otras soluciones a problemas similares, así como cualquier otra documentación que permita establecer los requisitos de la solución. Para ello, se recuperaron los siguientes documentos (Asman & Srikanth, 2015; d'Aquin & Gangemi, 2011; Ferreira de Souza et al., 2017; ISO, 2013a; ISTQB, 2021b) de la base de

conocimiento (BC). Después de revisar estos documentos y considerando el problema inicial, se establecieron 9 Requisitos de la solución los cuales ya se listaron en la Sección 3.3.3.1, llamados en esa instancia "requisitos ontológicos". Igualmente, por cuestiones prácticas, se listan a continuación nuevamente los 9 requisitos de la solución (RS):

- RS#1. Debe estar conceptualizada pero no necesariamente implementada.
- RS#2. Debe tener una amplia cobertura para el dominio de pruebas de software. Por lo tanto, debe contener conceptos relacionados con las pruebas estáticas y dinámicas, así como con las pruebas funcionales y no funcionales.
- RS#3. Debe considerar diferentes tipos de relaciones, es decir, relaciones taxonómicas ("tipo-de/es-un" y "todo-parte/parte-de") y relaciones no taxonómicas (asociaciones).
- RS#4. Debe estar a nivel ontológico de dominio superior (*top-domain ontological level* en Figura 3-8).
- RS#5. Debe desarrollarse siguiendo una metodología adecuada y rigurosa.
- RS#6. Debe desarrollarse considerando términos de estándares internacionales y/u otras ontologías, taxonomías o glosarios para el dominio de pruebas de software.
- RS#7. Debe contener términos del dominio de pruebas que puedan relacionarse con conceptos de requisitos funcionales y no funcionales.
- RS#8. Debe tener una conceptualización representada gráficamente en un lenguaje de modelado apropiado.
- RS#9. Debe tener términos que se basen o estén enriquecidos con términos de otras ontologías en niveles superiores, como ontologías de nivel central y/o fundamental (*core ontological level* y *foundational ontological level*, respectivamente, en Figura 3-8).

La segunda actividad de A1 es Investigar soluciones actuales la cual incluye 2 subactividades, a saber: Realizar investigación de la literatura y Obtener retroalimentación con expertos. Recordar que, aunque la Figura 4-1 muestra estas 2 actividades en paralelo, no es necesario realizar ambas. Para el caso de TestTDO se llevó a cabo solamente la subactividad Realizar investigación de la literatura, en la cual se realizó la RSL sobre ontologías de *testing* que fue documentada en el Capítulo 3. Notar que en esta actividad se utilizaron los 9 Requisitos de la solución (o requisitos ontológicos) como insumos, además de la BC para obtener todos los estudios primarios y otros documentos necesarios como por ejemplo (d'Aquin & Gangemi, 2011; ISO, 2013a; ISTQB, 2021b; Olsina et al., 2019). Como resultado, se obtiene la Síntesis de información la cual documenta, para este caso, un resumen de las 12 ontologías de pruebas seleccionadas como así también su evaluación ontológica (todos los detalles se encuentran en la Sección 3.3.3). Además, esta síntesis también contiene otros aspectos importantes, los cuales se ilustran a continuación.

Ninguna de las 12 ontologías seleccionadas y analizadas en la RSL fue construida con el mismo objetivo que se persigue para TestTDO, es decir, dar soporte a la especificación de una familia de estrategias integradas de pruebas que ayuden a alcanzar metas de *testing*. La mayoría de ellas habían sido construidas para lograr un objetivo muy específico y, por lo tanto, la mayoría no tienen un alcance similar al que se espera de TestTDO, que es de dominio superior y más amplio.

Hasta cierto punto, las ontologías con el objetivo más cercano a TestTDO son ROoST (Ferreira de Souza et al., 2017) y la ontología presentada por Asman *et al.* (Asman & Srikanth, 2015). ROoST fue desarrollada para establecer una conceptualización común sobre el dominio de las pruebas de software, centrándose en el proceso de pruebas, pero su alcance solo alcanza las pruebas dinámicas y funcionales sin considerar las pruebas estáticas y no funcionales. Por otro lado, la ontología documentada en (Asman & Srikanth, 2015) fue construida para representar el conocimiento general de las pruebas de software. En cuanto a su alcance, esta ontología solo tiene términos para el proceso de revisión formal, y no considera términos genéricos para el proceso de prueba. Adicionalmente, no contiene términos de dominio superior como *Test Basis* (Base de Prueba), *Test Result* (Resultado de Prueba), *Actual Result* (Resultado Real), además de términos relacionados con el proyecto de prueba, los objetivos, los requisitos y el entorno. A su vez, algunos términos no comparten la visión ontológica de nivel *core* de FCD-OntoArch. Por ejemplo, los métodos de prueba son tipos de metas de prueba, mientras que en FCD-OntoArch un método tiene una semántica diferente a una meta.

Las ontologías seleccionadas en la RSL fueron evaluadas desde el punto de vista de la calidad ontológica. Recordar que se utilizó un árbol de NFRs (1ra columna de la Tabla 3-10) que especifica características y atributos relacionados con la calidad ontológica (1), a saber: Calidad Estructural Ontológica (1.1), Calidad de Cobertura Terminológica específica del Dominio (1.2) y Adherencia a otros Vocabularios (1.3). Los resultados de la evaluación de las 12 ontologías seleccionadas se muestran en la Tabla 3-13. En resumen, la ontología mejor clasificada en cuanto a la calidad ontológica (1) fue ROoST (Ferreira de Souza et al., 2017), aunque no alcanzó un nivel satisfactorio. Esta ontología, alcanzó el 79,54% (🔸). Carece de cobertura terminológica de pruebas no funcionales (1.2.4) y estáticas (1.2.1), y no hay un vínculo directo con términos de requisitos funcionales y no funcionales (recordar el RS#7 mencionado anteriormente).

ROoST es la única ontología seleccionada que se basa en una ontología fundacional. Aunque ROoST está situada en una red de ontologías cuya raíz es la ontología fundacional UFO (Guizzardi, 2005), se observó que las ontologías utilizadas para enriquecer ROoST (es decir, UFO y principalmente su ontología central del proceso derivado) son algo difíciles de adaptar y armonizar con los componentes ontológicos que se presentan en la Sección 4.3.2. Además, UFO es un poco compleja ya que se compone de un conjunto de ontologías, a saber: UFO-A (*endurants*), UFO-B (*perdurants* o eventos) y UFO-C (entidades sociales, construidas sobre UFO-A y B). En cambio, FCD-OntoArch incluye solo una ontología fundacional, a saber, ThingFO, que tiene un pequeño conjunto de términos que facilita la reutilización y la especialización en ontologías de nivel inferior.

La siguiente tarea a realizar es Analizar la relevancia del problema/solución. Recordar que la misma requiere como entrada los artefactos Síntesis de información, Requisitos de la solución y la Definición del problema de negocio para poder producir el Informe de relevancia. Este informe describe la relevancia del problema/solución para decidir si será abordado por el enfoque de DSR. A continuación, se detallan los principales factores que impactaron positivamente en la decisión de abordar la construcción de la nueva ontología de pruebas de software mediante el enfoque de DSR: “i) no existe una ontología seleccionada en la RSL que cumpla satisfactoriamente con todos los requisitos de la solución establecidos y que también cumpla con un alto nivel de aceptabilidad de calidad ontológica, ii) construir/adaptar una ontología no es una tarea rutinaria, por el contrario, es una tarea compleja debido a que: es necesario considerar diferentes fuentes de definiciones de los términos para el dominio de *testing* (como estándares internacionales,

otras ontologías, etc.) para obtener una mayor cobertura que las soluciones actuales; el nuevo diseño ontológico debe permitir la vinculación con los componentes de FRsTDO y NFRsTDO en el contexto de la arquitectura FCD-OntoArch; y la nueva ontología debería estar basada en (enriquecida con) términos de ontologías de nivel superior, ya sea en el nivel central (*core*) y/o fundacional".

Dado que el problema/solución es relevante para el estado del arte, entonces el mismo será abordado utilizando el enfoque de DSR. En consecuencia, la siguiente actividad a realizar es Especificar requisitos de diseño, la cual incluye las tareas Especificar preguntas de investigación y Especificar requisitos del artefacto. Para llevar a cabo estas dos tareas se requiere como entradas la Síntesis de información y los Requisitos de la solución. Luego se genera, por un lado, las Preguntas de investigación (RQ). Para este caso se generaron 3 RQs principales, a saber:

RQ#1: ¿Cuáles son los requisitos más apropiados para la ontología de pruebas de software que ayude a respaldar estrategias de *testing* que den soporte para alcanzar objetivos de *testing*? A su vez, a partir de la misma se derivaron las siguientes 5 sub-RQs:

RQ#1.1: ¿La ontología a desarrollar debe ser de dominio de bajo nivel o de dominio de alto nivel?

RQ#1.2: ¿Cuáles son las terminologías documentadas más robustas y ricas (estructuradas como glosarios, taxonomías u ontologías) para el dominio de las pruebas de software?

RQ#1.3: ¿Cuáles son los principales términos que debe incluir la ontología de pruebas de software para alcanzar una buena cobertura?

RQ#1.4: ¿La ontología a desarrollar debe estar en la etapa de conceptualización y/o implementación?

RQ#1.5: ¿Cuál es una metodología adecuada para el desarrollo de ontologías a ser utilizada?

RQ#2: ¿Cómo integrar la ontología de *testing* con los componentes conceptuales existentes (ontologías) y los niveles del marco FCD-OntoArch? A su vez, a partir de la misma se derivaron las siguientes 2 sub-RQs:

RQ#2.1: ¿Con qué ontologías y niveles de FCD-OntoArch se debe relacionar directamente la nueva ontología de prueba?

RQ#2.2: ¿Qué términos son necesarios para relacionar e integrar las diferentes ontologías de FCD-OntoArch con la nueva ontología de pruebas de software?

RQ#3: ¿Cómo asegurar la calidad de la ontología de pruebas de software? A su vez, a partir de la misma se derivaron las siguientes 2 sub-RQs:

RQ#3.1: ¿Qué características de calidad de la ontología se deben evaluar?

RQ#3.2: ¿Qué estrategias de verificación y validación son las más adecuadas para evaluar la ontología?

Por otro lado, después de llevar a cabo la tarea Especificar requisitos del artefacto utilizando como entrada también las RQs anteriores (recordar que además se utilizan como entrada la Síntesis de información y los Requisitos de la solución), se produjeron los siguientes Requisitos del artefacto (AR):

AR#1) Diseñar y construir una ontología de dominio de alto nivel. Notar que este requisito se relaciona con RQ#1.1 y el requisito de la solución RS#4. Además, este requisito surge ya que se prevé desarrollar un conjunto de estrategias de *testing* para algunos tipos específicos de pruebas como las pruebas de rendimiento, seguridad, estáticas por revisiones, entre otras, y, por lo tanto, será útil tener una ontología de alto nivel como base semántica para enriquecer conceptos de *testing* de más bajo nivel.

AR#2) Considerar principalmente: i) glosarios de estándares internacionales documentados en ISO 29119-1 (ISO, 2013a) e ISTQB (ISTQB, 2021b); y ii) la ontología de dominio ROoST (Ferreira de Souza et al., 2017) y la ontología de dominio superior de Asman *et al.* (Asman & Srikanth, 2015), que fueron las dos mejor clasificadas entre las 12 ontologías seleccionadas y evaluadas en la RSL conducida en la actividad Realizar investigación de la literatura. Notar que este requisito se relaciona con RQ#1.2 y RS#6, y a su vez utiliza los datos obtenidos en la Síntesis de información. Además, es muy importante considerar y reutilizar otras terminologías (principalmente glosarios de estándares) para la construcción de una nueva ontología dado que una ontología es una conceptualización compartida.

AR#3) La cobertura terminológica de la ontología de dominio superior debe ser la necesaria y suficiente para ser extendida por otras ontologías de dominio inferior. Para ello, debe considerar términos relacionados con pruebas estáticas y dinámicas, así como pruebas funcionales y no funcionales. Además, debe considerar conceptos de *testing* para entidades de trabajo (procesos de trabajo, actividad y tarea), productos de trabajo (artefactos, resultados), métodos, agentes, proyectos, metas, requisitos y entidades (como entidad desarrollable o evaluable), los cuales deben ser enriquecidos semánticamente por conceptos de otras ontologías de los niveles *core* y fundacional. Notar que este requisito se relaciona con RQ#1.3 y los requisitos RS#2, RS#7, y RS#9.

En este punto es importante mencionar que, como parte de este AR#3, se definieron 25 Preguntas de competencia (CQ), las cuales especifican los requisitos del alcance de la ontología TestTDO con respecto a su cobertura terminológica. A continuación, se listan las mismas.

CQs relacionadas con Productos de Trabajo de Prueba:

CQ1. ¿Cuáles son los Productos de Trabajo producidos por una actividad de Diseñar la Prueba?

CQ2. ¿Cuáles son los Productos de Trabajo consumidos por una actividad de Diseñar la Prueba?

CQ3. ¿Cuáles son los Productos de Trabajo producidos por una actividad de Realizar la Prueba?

CQ4. ¿Cuáles son los Productos de Trabajo consumidos por una actividad de Realizar la Prueba?

CQ5. ¿Cuáles son los Productos de Trabajo producidos por una actividad de Analizar Resultados de Prueba?

CQ6. ¿Cuáles son los Productos de Trabajo consumidos por una actividad de Analizar Resultados de Prueba?

CQs relacionadas con Actividades de Prueba:

CQ7. ¿Cuáles son los tipos de Actividades de Prueba dinámicas?

CQ8. ¿Cuál es el conjunto mínimo de Actividades de Prueba incluidas en un Proceso de Prueba?

CQ9. ¿Cuáles son los tipos de actividades de Realizar la Prueba que consumen Casos de Prueba?

CQ10. ¿Cuáles son los tipos de actividades de Realizar la Prueba que consumen Listas de Verificación?

CQ11. ¿Cuáles son los tipos de entidades de prueba requeridas por un Proceso de Prueba como entrada?

CQs relacionadas con Métodos de Prueba:

CQ12. ¿Cuáles son los tipos de Métodos de Prueba asignados a una Actividad de Prueba dinámica?

CQ13. ¿Cuáles son los tipos de Métodos de Prueba asignados a una Actividad de Prueba estática?

CQ14. ¿Cuáles son los tipos de Métodos de Prueba asignados a una actividad de Diseñar la Prueba?

CQs relacionadas con Agentes de Prueba:

CQ15. ¿Cuáles son los tipos de Agentes de Prueba asignados a una Actividad de Prueba?

CQ16. ¿Un Proceso de Prueba implica roles?

CQ17. ¿Puede un Agentes de Prueba usar herramientas?

CQs relacionadas con entidades de prueba:

CQ18. ¿En qué Situación Particular una Entidad Comprobable se considera una Entidad Evaluable?

CQ19. ¿En qué Situación Particular una Entidad Comprobable se considera una Entidad Desarrollable?

CQ20. En una Situación Particular, ¿puede una Entidad Comprobable estar rodeada de Entidades de Contexto de Prueba?

CQs relacionadas con Metas de Prueba:

CQ21. ¿Se puede derivar una Meta de Prueba en uno o más Requisitos de Prueba?

CQ22. ¿Existe una Actividad de Prueba que considere una Meta de Necesidad de Información de Prueba?

CQs relacionadas con Requisitos de Prueba:

CQ23. Para un Requisito de Prueba que se refiere a un Objeto de Prueba, ¿cuál es el nivel de prueba?

CQ24. ¿Un Requisito de Prueba está relacionado con Requisitos Funcionales y No Funcionales?

CQs relacionadas con Proyectos de Prueba:

CQ25. Para un Proyecto de Prueba que operacionaliza una Meta de Prueba, ¿tiene este proyecto una Estrategia de Prueba asociada que ayuda a alcanzar el propósito de la Meta de Prueba?

AR#4) Debe existir un equilibrio entre la disponibilidad de relaciones taxonómicas y no taxonómicas. Notar que este requisito se relaciona con el requisito de la solución RS#3. A su vez, es importante considerar esta característica ya que una base conceptual no sería una ontología sino tuviera relaciones no taxonómicas.

AR#5) La ontología de pruebas de software de dominio superior debe estar conceptualizada y no necesariamente implementada en un lenguaje formal, ya que el objetivo principal es utilizarla como un vocabulario común para enriquecer especificaciones de procesos y métodos de estrategias de prueba a ser desarrolladas. Notar que este requisito se relaciona con RQ#1.4 y RS#1.

AR#6) Utilizar *METHONTOLOGY* (Fernández-López et al., 1997) para el desarrollo de la ontología hasta su etapa de conceptualización. Notar que este requisito se relaciona con RQ#1.5 y los requisitos RS#1 y RS#5. Además, es importante considerar alguna metodología ingenieril a ser utilizada en la actividad A2 (Diseñar y desarrollar la solución) del proceso de DSR para construir una ontología. Se seleccionó *METHONTOLOGY* ya que es una metodología bien estructurada utilizada para desarrollar ontologías desde cero y proporciona buenas guías para organizar las actividades durante el desarrollo de la ontología (Asman & Srikanth, 2015).

AR#7) Dado que las pruebas de software se utilizan para verificar/validar requisitos funcionales/no funcionales, entonces es importante enlazar la ontología de *testing* con las ontologías de FRsTDO y NFRsTDO. Por lo tanto, se deben relacionar algunos términos de la ontología de pruebas de software con las ontologías FRsTDO y NFRsTDO en el nivel de dominio superior (como se muestra en la Figura 3-8). Además, algunos términos deben estar relacionados con las ontologías SituationCO y ProcessCO en el nivel *core*, y con ThingFO en el nivel fundacional. Notar que SituationCO contiene términos de ProjectCO y GoalCO (Becker et al., 2022) que también serán considerados para enriquecer TestTDO. Este requisito se relaciona con RQ#2.1 y los requisitos RS#6, RS#7 y RS#9.

AR#8) Los términos necesarios para relacionar e integrar las diferentes ontologías de FCD-OntoArch con la ontología de pruebas de software de dominio superior son principalmente: i) para NFRsTDO: los términos de Requisito no Funcional y Entidad Evaluable; ii) FRsTDO: Requisito Funcional y Entidad Desarrollable; iii) GoalCO: Meta de Negocio y Meta de Necesidad de Información; iv) ProjectCO: Proyecto, Estrategia y Plan del Proyecto; v) SituationCO: Entidad de Contexto y Situación Particular; y vi) ProcessCO: Proceso de Trabajo, Actividad, Producto de Trabajo, Artefacto, Resultado, Método, Rol, Herramienta y Agente. Notar que este requisito se relaciona con RQ#2.2 y los requisitos RS#6, RS#7 y RS#9.

AR#9) Las características de calidad que se evaluarán son las mismas que las utilizadas en la RSL para las 12 ontologías de pruebas de software seleccionadas, a saber: Calidad Estructural Ontológica, Disponibilidad de Relaciones Balanceadas, Calidad de Cobertura Terminológica Específica del Dominio y Adherencia a otros Vocabularios. Notar que este requisito se relaciona con RQ#3.1.

AR#10) La estrategia de evaluación para valorar la ontología resultante es GOCAME (*Goal-Oriented Context-Aware Measurement and Evaluation*) (Becker et al., 2015; Olsina & Becker, 2017). Además, las CQs deben ser verificadas con una matriz de verificación que incluya los términos, propiedades, relaciones y axiomas de la ontología resultante que corresponden a cada CQ. También se debe implementar la ontología para poder verificar las CQs dinámicamente. La validación de la ontología resultante debe

realizarse aplicando sus términos a un proyecto de prueba del mundo real, además de ser examinada con otros expertos del dominio para obtener retroalimentación. Notar que este requisito se relaciona con RQ#3.2.

En este punto finaliza la subactividad Especificar requisitos de diseño y, por lo tanto, también finaliza A1.

4.3.3.2 A2 Diseñar y Desarrollar la Solución

Una vez finalizada A1, la siguiente actividad a realizar es A2 Diseñar y Desarrollar la Solución. La primera subactividad a llevar a cabo es Seleccionar procesos y métodos de desarrollo, para la cual se utiliza como entrada los ARs y la BC. Según el AR#6, se debe utilizar *METHONTOLOGY* (Fernández-López et al., 1997) para el desarrollo de la ontología hasta su etapa de conceptualización. Por lo tanto, se utilizará dicha metodología para construir la ontología de pruebas de software en la siguiente subactividad.

La siguiente actividad a realizar es Diseñar y Construir una Versión del Artefacto, en la cual se obtuvo una versión de TestTDO. Durante la actividad de conceptualización de *METHONTOLOGY* se obtuvo el modelo conceptual de TestTDO. Como resultado, TestTDO tiene definidos 44 términos, 51 propiedades, 43 relaciones no taxonómicas y 17 axiomas especificados en lógica de primer orden. La Figura 4-2 muestra todos los conceptos de TestTDO, así como su relación con los términos de NFRsTDO y FRsTDO.

Para obtener la conceptualización de TestTDO, *METHONTOLOGY* propone primero construir un glosario, es decir, conceptos con sus definiciones. Este glosario debe incluir términos, propiedades y relaciones no taxonómicas. Para desarrollar el glosario preliminar de TestTDO se han utilizado de la BC los glosarios de los estándares ISO 29119 e ISTQB con el objetivo de recopilar el conjunto de conceptos primarios que se incluirán en TestTDO (tal como especifica AR#2). Además, la selección de los términos a considerar en este glosario preliminar se realizó teniendo en cuenta los patrones conceptuales de las ontologías centrales (*core*) de FCD-OntoArch y el alcance de TestTDO representado en las 25 CQs (recordar AR#3).

Luego de construir este glosario preliminar, se analizó la semántica de sus términos proporcionada por los glosarios de los estándares ISO 29119 e ISTQB, y también se consideraron las ontologías ROoST (Ferreira de Souza et al., 2017) y la de Asman *et al*. (Asman & Srikanth, 2015), tal como indica AR#2. El objetivo de esta tarea era establecer una correspondencia semántica con los términos de las ontologías FCD-OntoArch a nivel *core*. Por ejemplo, se identificó que *Design Testing* tiene semántica de Actividad de ProcessCO.

El siguiente paso fue identificar y establecer generalizaciones (o taxonomías) entre los conceptos, analizando la semántica de los mismo. En resumen, en este paso se identificaron relaciones taxonómicas (“tipo-de/es-un” y “todo-parte/parte-de”). Además, los patrones conceptuales de las ontologías de FCD-OntoArch a nivel *core* fueron muy útiles para realizar esta tarea. Por ejemplo, el patrón de Entidad de Trabajo de ProcessCO implica una relación de todo-parte en la que un Proceso de Trabajo tiene una o más Actividades.

Lo siguiente fue considerar la reutilización de algunas propiedades y relaciones no taxonómicas que pertenecen a las ontologías de FCD-OntoArch a nivel *core*. Por ejemplo, se reutilizó la relación “Proyecto *operacionaliza* Metas” en TestTDO entre el Proyecto de Prueba y la Meta de Prueba. Finalmente, se obtuvo un modelo conceptual utilizando un diagrama de clases UML representando el glosario de términos.

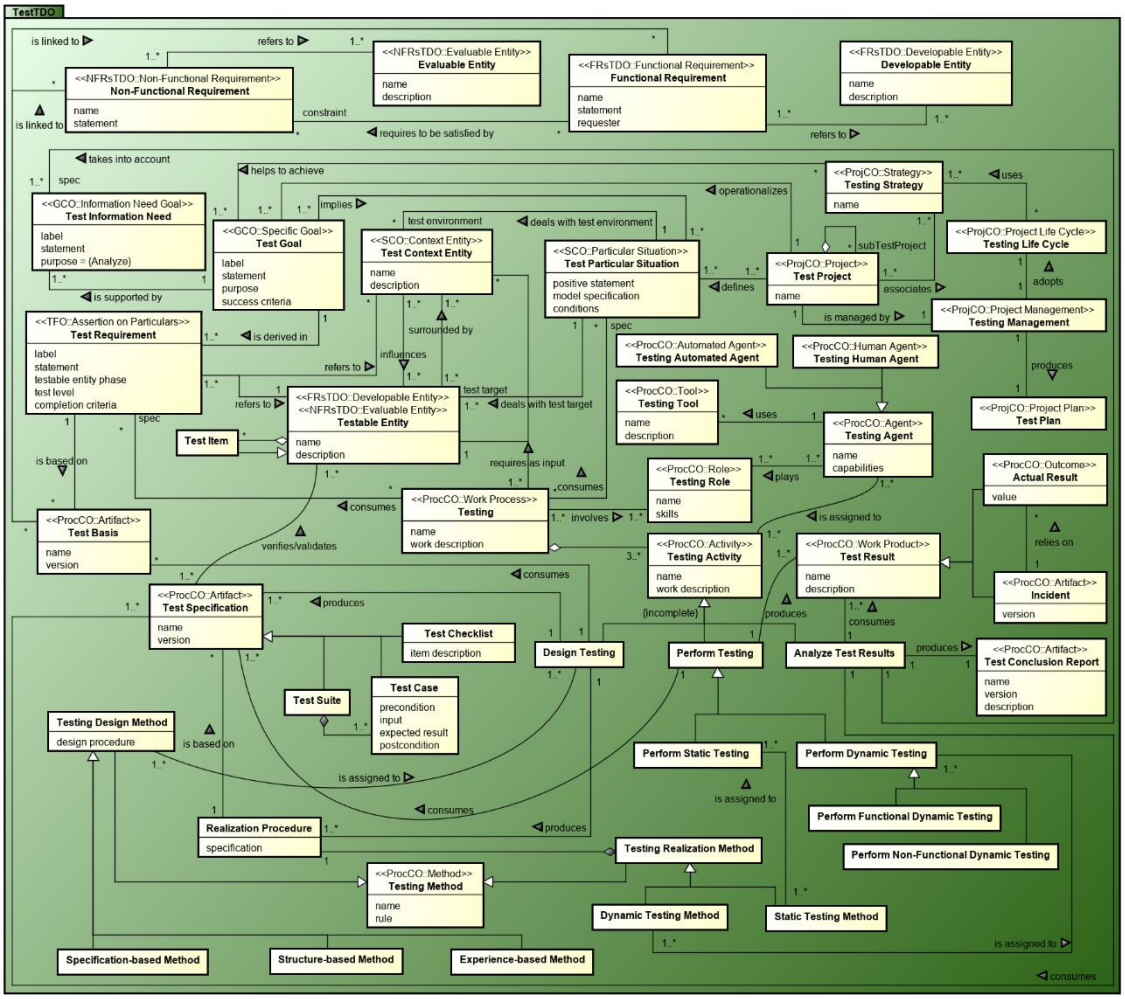


Figura 4-2: TestTDO: *Top-Domain Ontology for Testing*, que se ubica en la capa de dominio de alto nivel de FCD-OntoArch (recordar Figura 3-8). Tenga en cuenta que TFO significa *Thing Foundational Ontology* (Olsina, 2021), SCO *Situation Core Ontology* (Olsina et al., 2021), ProcCO *Process Core Ontology* (Becker et al., 2022), ProjCO *Project Core Ontology* (Becker et al., 2022; Rivera et al., 2016), GCO *Goal Core Ontology* (Rivera et al., 2016), NFRsTDO *Non-Functional Requirements Top-Domain Ontology* y FRsTDO *Functional Requirements Top-Domain Ontology* (Becker et al., 2019; Olsina et al., 2023; Tebes, Olsina, et al., 2020a).

Además, dado que el AR#10 establece que la ontología este implementada para verificarla dinámicamente, se consideró ejecutar la actividad de implementación de *METHONTOLOGY* para obtener una versión de OWL de TestTDO. Se darán más detalles de la implementación de TestTDO en la subsección siguiente.

A continuación, en los párrafos siguientes, se describe la conceptualización de TestTDO por partes. Recordar que, en este capítulo, los términos principales de las ontologías se escriben primero en inglés, comenzado con mayúscula y con cursiva, seguido de su traducción al español entre paréntesis. Luego, se utiliza solo su traducción al español empezando con mayúscula. Además, en esta sección, las propiedades y relaciones no taxonómicas de TestTDO estarán subrayadas, con cursiva y en minúscula.

Conceptos de TestTDO relacionados con Proyectos/Metas/Requisitos/Entidades

Para cubrir el alcance relacionado con requisitos y entidades de prueba, TestTDO tiene términos como *Test Requirement* (Requisito de Prueba), *Test Basis* (Base de Prueba), *Testable Entity* (Entidad Comprobable), *Test Item* (Elemento de Prueba), *Test Context Entity* (Entidad de Contexto de Prueba) y *Test Particular Situation* (Situación Particular

de Prueba), como se muestra en la Figura 4-3. Notar que estos términos se enriquecen semánticamente con términos de ThingFO, ProcessCO y SituationCO (por ejemplo, Situación Particular de Prueba tiene semántica de Situación Particular de SituationCO). Un Requisito de Prueba establece, teniendo en cuenta el *propósito* de la *Test Goal* (Meta de Prueba), lo que se debe verificar/validar de una Entidad Comprobable (y/o Elemento de Prueba) *basándose* en la Base de Prueba, si corresponde. Por lo tanto, un Requisito de Prueba tiene una *declaración* que *refiere a* una Entidad Comprobable. Además, un Requisito de Prueba puede incluir detalles de los requisitos del entorno de prueba, que siempre *refieren a* Entidades de Contexto de Prueba. Adicionalmente, un Requisito de Prueba debe incluir el *nivel de prueba*, el cual representa un tipo de prueba que delimita el alcance de la Entidad Comprobable y su contexto. Ejemplos de tipos de *niveles de prueba* comúnmente citados en la literatura para pruebas dinámicas son el nivel unitario, de integración, sistema y aceptación.

La Base de Prueba es un Artefacto utilizado por *Testing Design Methods* (Métodos de Diseño de Prueba) para diseñar *Test Cases* (Casos de Prueba) y *Test Checklists* (Listas de Verificación). La Base de Prueba puede producirse en las etapas de desarrollo y/o mantenimiento del software (por ejemplo, especificación de requisitos, diseño arquitectónico, código fuente documentado, entre otros), y a su vez podría estar *enlazada a* Requisitos No Funcionales y/o Requisitos Funcionales.

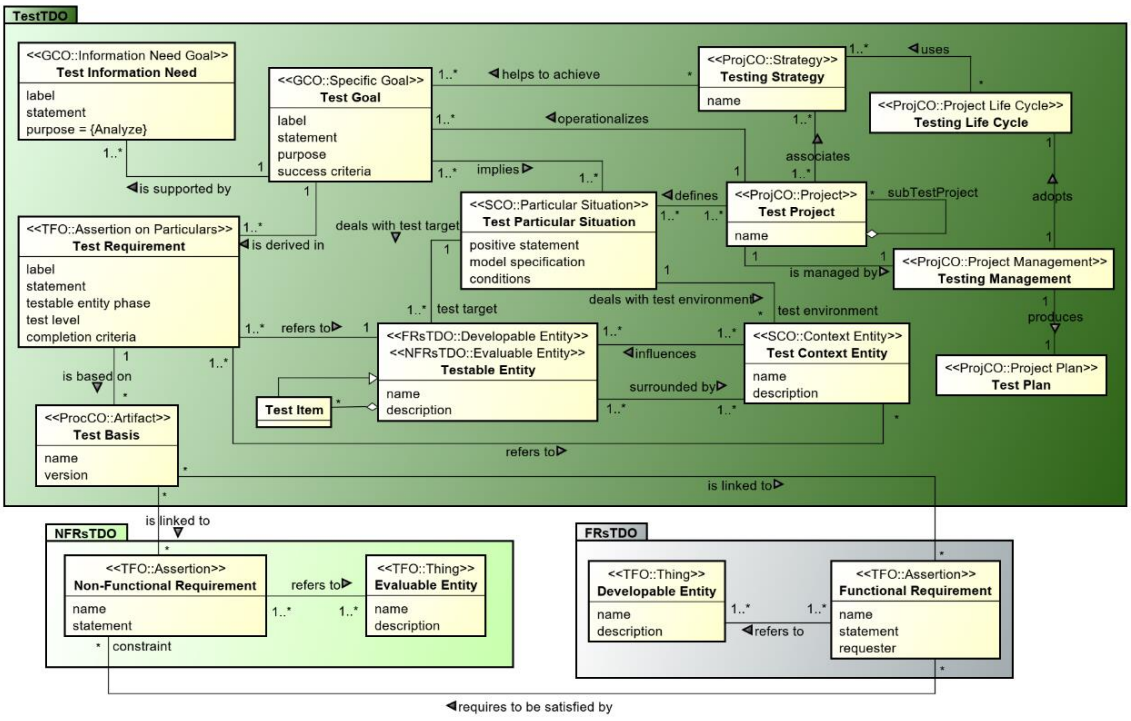


Figura 4-3: Fragmento de los términos, relaciones y propiedades de TestTDO relacionados con Proyectos/Metas/Requisitos/Entidades, y su relación con los términos de los componentes NFRsTDO y FRsTDO. Recordar que TFO significa *Thing Foundational Ontology*, SCO *Situation Core Ontology*, ProcCO *Process Core Ontology*, ProjCO *Project Core Ontology*, GCO *Goal Core Ontology*, NFRsTDO *Non-Functional Requirements Top-Domain Ontology* y FRsTDO *Functional Requirements Top-Domain Ontology.*

Por otro lado, una Entidad Comprobable es un objeto concreto que se puede probar. Una Entidad Comprobable siempre *está rodeada de* Entidades de Contexto de Prueba que la *influencian*. La Entidad de Contexto de Prueba representa el Objeto de Contexto concreto en el que se sitúa la Entidad Comprobable. Aunque siempre hay Entidades de

Contexto de Prueba que *rodean a* la Entidad Comprobable –ya que una Cosa nunca está aislada-, dependiendo de la Situación Particular de Prueba *definida por* el *Test Project* (Proyecto de Prueba), las *Testing Activities* (Actividades de Prueba) pueden considerar el entorno de prueba o no. Por lo tanto, una Situación Particular de Prueba (con semántica de Situación Particular de SituationCO) *trata con* una o más Entidades Comprobable (en el rol de objetivo de prueba) y ninguna o muchas Entidades de Contexto de Prueba (en el rol de entorno de prueba).

Además, dependiendo de la Situación Particular de Prueba *implicada por* la Meta de Prueba, la Entidad Comprobable tiene semántica de Entidad Desarrollable o Entidad Evaluable. Una Situación Particular de Prueba en la que la Entidad Comprobable tiene semántica de Entidad Evaluable es cuando el Requisito de Prueba que *hace referencia a* la Entidad Comprobable *está vinculado a* un Requisitos No Funcional a través de la Base de Prueba asociada. Por otro lado, cuando el Requisito de Prueba que *hace referencia a* la Entidad Comprobable *se vincula a* un Requisitos Funcional a través de la Base de Prueba asociada, la Entidad Comprobable tiene semántica de Entidad Desarrollable. Para establecer formalmente las afirmaciones anteriores y descartar interpretaciones no deseadas se especificaron los siguientes axiomas:

Especificación del axioma A_I:

$$\forall te, \exists tr, \exists tb, \exists nfr: [TestableEntity(te) \wedge EvaluableEntity(te) \leftrightarrow TestRequirement(tr) \wedge TestBasis(tb) \wedge NonFunctionalRequirement(nfr) \wedge refersTo(tr,te) \wedge isBasedOn(tr,tb) \wedge isLinkedTo(tb,nfr)]$$

Descripción del axioma A_I: Cualquier Entidad Comprobable es una Entidad Evaluable si y solo si el Requisito de Prueba que *hace referencia a* esta Cosa *está vinculado a* un Requisito No Funcional.

Especificación del axioma A_II:

$$\forall te, \exists tr, \exists tb, \exists fr: [TestableEntity(te) \wedge DevelopableEntity(te) \leftrightarrow TestRequirement(tr) \wedge TestBasis(tb) \wedge FunctionalRequirement(fr) \wedge refersTo(tr,te) \wedge isBasedOn(tr,tb) \wedge isLinkedTo(tb,fr)]$$

Descripción del axioma A_II: Cualquier Entidad Comprobable es una Entidad Desarrollable si y solo si el Requisito de Prueba que *se refiere a* esta Cosa *está vinculado a* un Requisito Funcional.

Por último, es importante resaltar que TestTDO tiene un bloque o patrón conceptual que se hereda de las ontologías a nivel central (*core*). Esto involucra la relación entre Proyectos de Prueba, Metas de Prueba, *Testing Strategies* (Estrategias de Prueba) y Situaciones Particulares de Prueba. Como se muestra en la Figura 4-3, un Proyecto de Prueba *operacionaliza* una Meta de prueba. Para ello, un Proyecto de Prueba *asocia* (o utiliza) una Estrategia de Prueba *que ayuda a alcanzar* la Meta de Prueba. Además, esta Meta de Prueba *implica* una Situación Particular de Prueba que *está definida por* el Proyecto de Prueba. Notar que este patrón conceptual no se encuentra presente en ninguna de las 12 ontologías seleccionadas de la RSL.

A continuación, se listan las definiciones de los términos, propiedades y relaciones no taxonómicas de TestTDO relacionados con Proyectos/Metas/Requisitos/Entidades en las Tablas 4-1, 4-2 y 4-3, respectivamente. Estas definiciones se dejaron en inglés en esta tesis para evitar traducciones que lleven a entendimientos erróneos de las mismas.

Tabla 4-1: Definiciones en inglés de los términos principales de TestTDO relacionados con Proyectos/Metas/Requisitos/Entidades.

Término	Definición
Test Context Entity	It represents the concrete Context Object in which the Testable Entity is situated. Note 1: Test Context Entity has a semantic of Context Entity (Olsina, 2021). Note 2: A Test Context Entity belongs to a Context Category, which test environment is one type of it. Note 3: A test environment is "*an environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test*" as per (ISTQB, 2021b).
Test Goal (synonym: Test Objective)	It is a business goal for testing that the organization intends to achieve. Note: A business goal is, in turn, a goal, which is a "*The statement of the aim to be achieved by the organization, which considers the propositional content of a purpose in a given time frame and context*" (Rivera et al., 2016).
Test Information Need	It is an Information Need Goal (Rivera et al., 2016) that is achieved by conducting Testing Activities. Note: Particularly, a specific Test Information Need is taken into account by the Analyze Test Results Activity.
Test Item	A part of a Testable Entity, which is a Test Object as well.
Test Particular Situation	It represents an association between one or more Testable Entities in the role of test target and none or many Test Context Entities in the role of test environment. Note: The term Test Particular Situation has semantic of Particular Situation (Olsina, 2021), which is an Assertion on Particulars from ThingFO (Olsina, 2021).
Test Plan	The document (Artifact) that describes how the Test Project will be planned, scheduled, executed, monitored, and controlled.
Test Project	It is a Project (Rivera et al., 2016) representing a temporary and goal-oriented endeavor for testing with definite start and finish dates, which considers a managed set of interrelated Testing Activities, tasks and resources aimed at producing and modifying unique work products (e.g., Test Specifications, Test Results, etc.) for satisfying a given requester need.
Test Requirement	It states, taking into account the Test Goal purpose, what must be verified/validated of a Testable Entity (and/or Test Item) based on the Test Basis, if any. Note 1: A Test Requirement must include the test level (e.g., unit, integration, system, acceptance, etc.) and the phase (e.g., development, operative, maintenance, etc.) of the Testable Entity. Note 2: The term Test Requirement has semantic of Assertion on Particulars from ThingFO (Olsina, 2021). Note 3: A Test Requirement can include details of test environment requirements, which always refer to Test Context Entities.
Testable Entity (synonym: Test Object)	A concrete object able to be tested. Note 1: Testable Entity may have none or many Test Items, which in turn are testable. Note 2: Depending on the Test Particular Situation implied by the Test Goal, Testable Entity has semantic of Developable Entity (from FRsTDO (Tebes, Olsina, et al., 2020a)) or Evaluable Entity (from NFRsTDO (Olsina et al., 2023)).

Testing Life Cycle	The series of phases that a Test Project passes through from its initiation to its closure.
Testing Management	It is the set of managerial processes and activities intended to achieve the Test Goal operationalized by a Test Project.
Testing Strategy	Principles, patterns, and particular test domain concepts and framework that can be specified by a set of core Testing Processes, in addition to a set of appropriated Testing Methods and Tools, as core resources, for helping to achieve the Project's Test Goal purpose.

Tabla 4-2: Definiciones en inglés de las propiedades/atributos de TestTDO relacionados con términos de Proyectos/Metas/Requisitos/Entidades.

Témino	Propiedad /Atributo	Definición
Test Context Entity	**name**	Label or name that identifies the Test Context Entity.
	description	An unambiguous textual statement describing the Test Context Entity.
Test Goal	**label**	Label that identifies a Test Goal uniquely.
	statement	An explicit declaration of the aim to be achieved. Note: A statement is usually a written assertion in a high-level or natural language.
	purpose	The rationale for achieving the specified Test Goal. Note: Examples of test purposes are: verify, validate, find defects, non-compliances, or vulnerabilities, etc.
	success criteria	The set of conditions by which the Test Goal will be judged as successful for stakeholders.
Test Information Need	**label**	Label that identifies a Test Information Need uniquely.
	statement	An explicit declaration of the aim to be achieved. Note: A statement is usually a written assertion in a high-level or natural language.
	purpose	The rationale for achieving the specified Test Information Need goal, which basically consists on analyze to provide information.
Test Particular Situation	**positive statement**	An explicit declaration of a Test Particular Situation to be defined, which can refer to a static or dynamic situation. Note 1: Regarding a particular Thing, a descriptive, positive statement refers to what it is, was, or will be, and contains no indication of approval or disapproval. Note 2: For a Test Particular Situation, the positive statement refers to things and relationships of Testable and Test Context Objects. Note 3: A positive statement should be based on current or subsequent empirical evidence.
	model specification	It represents an Artifact that specifies and models Test Particular Situations in a given language.

		Note 1: A Test Particular Situation model has the semantic of Artifact, which is a term coming from ProcessCO (Becker et al., 2022). Note 2: Test Particular Situations can be modeled using informal, semiformal or formal specification languages.
	conditions	Any kind of constraint that must be fulfilled in the Test Particular Situation. Note 1: In the scenario testing approach, some conditions could be preconditions (or postconditions), which are any kind of constraint that must evaluate true before (or after) starting (or finishing) the execution of the Test Particular Situation (i.e., the scenario). Note 2: This property has semantic of Constraint-related Assertion from ThingFO (Olsina, 2021).
Test Project	**name**	Label or name that identifies the Test Project.
Test Requirement	**label**	Label that identifies a Test Requirement uniquely.
	statement	An explicit declaration of the Test Requirement to be satisfied. Note: A statement is usually a written assertion in a high-level or natural language.
	testable entity phase	It indicates the stage of the testable-entity life cycle in which the Testable Entity is. Note: Examples of phases are “inception”, “development”, “deployment”, “operation” and “maintenance”.
	test level	It represents a kind of test that delimits the scope of the Testable Entity and its context taking into account the Test Requirement statement. Note: Examples of kinds of test levels commonly cited are “unit”, “integration”, “system” and “acceptance”. We can also include the “document” test level.
	completion criteria	The set of conditions by which the Test Requirement will be judged as complete for stakeholders.
Testable Entity	**name**	Label or name that identifies the Testable Entity.
	description	An unambiguous textual statement describing the Testable Entity.
Testing Strategy	**name**	Label or name that identifies the Testing Strategy.

Tabla 4-3: Definiciones en inglés de las relaciones no taxonómicas de TestTDO relacionados con términos de Proyectos/Metas/Requisitos/Entidades.

Relación no taxonómica	Definición
adopts	A Testing Management process adopts a Testing Life Cycle.

associates	A Test Project associates one or more Testing Strategies.
deals with test environment	A Test Particular Situation deals with none or many concrete Test Context Entities as a test environment.
deals with test target	A Test Particular Situation deals with one or more concrete Testable Entities as a test target.
defines	A Test Project defines one or several Test Particular Situations.
helps to achieve	A Testing Strategy gives support for achieving one or more Test Goals.
implies	A Test Goal implies one or more Test Particular Situations.
influences	A Test Context Entity influences one or several Testable Entities.
is based on	A Test Requirement is based on none or several Test Basis.
is derived in	A Test Goal is derived in one or more Test Requirements.
is linked to (x2)	A Test Basis is linked to none or several Non-Functional Requirements.
	A Test Basis is linked to none or several Functional Requirements.
is managed by	A Test Project is managed by means of a Testing Management process.
is supported by	A Test Goal is supported by one or more Test Information Needs.
operationalizes	A Test Project operationalizes one or more Test Goals.
produces	A Testing Management process produces a Test Plan as artifact.
refers to (x2)	A Test Requirement in its statement always refers to a Testable Entity.
	A Test Requirement can refer to a one or more Test Context Entities.
surrounded by	A Testable Entity is surrounded by one or several Test Context Entities.
uses	A Testing Life Cycle uses one or more Testing Strategies.

A seguir, se listan otros axiomas de TestTDO que incluyen conceptos relacionados con Proyectos/Metas/Requisitos/Entidades:

Especificación del axioma A_III:

$$\forall tr, \exists tg, \exists tp: [TestRequirement(tr) \wedge TestGoal(tg) \wedge isDerivedIn(tg, tr) \rightarrow TestProject(tp) \wedge operationalizes(tp, tg)]$$

Descripción del axioma A_III: Para todo Requisito de Prueba *derivado de* una Meta de Prueba hay al menos un Proyecto de Prueba que *operacionaliza* esta Meta de Prueba.

Especificación del axioma A_IV:

$$\forall tp, \forall tg, \forall ts: [TestProject(tp) \wedge TestGoal(tg) \wedge TestingStrategy(ts) \wedge operationalizes(tp, tg) \wedge associates(tp, ts) \leftrightarrow helpsToAchieve(ts, tg)]$$

Descripción del axioma A_IV: Todos los Proyectos de Prueba *operacionalizan* una Meta de Prueba y *asocian* una Estrategia de Prueba si y solo si esta Estrategia de Prueba *ayuda a alcanzar* la Meta de Prueba operacionalizada.

Conceptos de TestTDO relacionados con Productos de Trabajo/Actividades

Para cubrir el alcance relacionado con productos de trabajo y actividades de prueba, TestTDO tiene términos como Base de Prueba, *Test Specification* (Especificación de Prueba), *Test Result* (Resultado de Prueba), *Test Conclusion Report* (Informe de Conclusión de Prueba) y *Testing Activity* (Actividad de Prueba), como se muestra en la Figura 4-4. Los primeros 4 términos son Productos de Trabajo o, más específicamente, Artefactos (recordando que un Artefacto es un Producto de Trabajo en ProcessCO) que son *consumidos/producidos* por las Actividades de Prueba. Una Actividad de Prueba puede ser *Design Testing* (Diseñar la Prueba), *Perform Testing* (Realizar la Prueba) o *Analyze Test Results* (Analizar Resultados de Prueba). Tenga en cuenta que el *Testing work process* (Proceso de Prueba) se compone de al menos las tres Actividades de Prueba mencionadas anteriormente. Estas tres Actividades son el conjunto mínimo y necesario para todo Proceso de Prueba. Se pueden considerar otras actividades y subactividades, pero solo se hicieron explícitas las actividades genéricas ya que la ontología está en el nivel de dominio superior. A su vez, TestTDO formaliza lo mencionado anteriormente en el siguiente axioma:

Especificación del axioma A_V:

$$\forall p, \exists a_1, \exists a_2, \exists a_3 : [Testing(p) \wedge (a_1 \neq a_2) \wedge (a_1 \neq a_3) \wedge (a_2 \neq a_3) \wedge DesignTesting(a_1) \wedge PerformTesting(a_2) \wedge AnalyzeTestResults(a_3) \wedge partOf(a_1, p) \wedge partOf(a_2, p) \wedge partOf(a_3, p)]$$

Descripción del axioma A_V: Cualquier Proceso de Prueba tiene al menos tres Actividades diferentes, a saber: Diseñar la Prueba, Realizar la Prueba y Analizar Resultados de Prueba.

Diseñar la Prueba es una Actividad de Prueba que tiene como objetivo diseñar (es decir, *producir*) un conjunto de Especificaciones de Prueba y *Realization Procedures* (Procedimientos de Realización). La Especificación de Prueba tiene una semántica de Artefacto y hay tres tipos, a saber: Lista de Verificación, Caso de prueba y *Test Suite* (Conjunto de Casos de Prueba). Los Casos de Prueba contienen la información necesaria (*precondiciones*, *entradas*, *resultados esperados* y *postcondiciones*) para llevar a cabo principalmente pruebas dinámicas. Tenga en cuenta que Casos de Prueba con restricciones comunes para su realización se pueden agrupar en Conjuntos de Casos de Prueba. Por otro lado, las Listas de Verificación de pruebas contienen una lista de *descripciones de elementos* a ser comprobadas para realizar principalmente pruebas estáticas.

Considerando el Procedimiento de Realización, es un conjunto ordenado de instrucciones u operaciones del *Testing Realization Method* (Método de Realización de Pruebas), que especifica cómo se debe realizar la actividad de Realizar la Prueba utilizando o *basándose* en las Especificaciones de Prueba. Principalmente, cuando se realizan pruebas dinámicas que involucran la ejecución de Casos de Prueba, este término (Procedimiento de Realización) puede ser sinónimo de procedimiento de prueba (*test procedure* en (ISO, 2013a)). Un procedimiento de prueba según ISO 29119-1 (ISO, 2013a) es una "*sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution. Test procedures include detailed instructions for how to run a set of one or more test cases selected to be run consecutively, including set up of common preconditions, and providing input and evaluating the actual result for each included test case*".

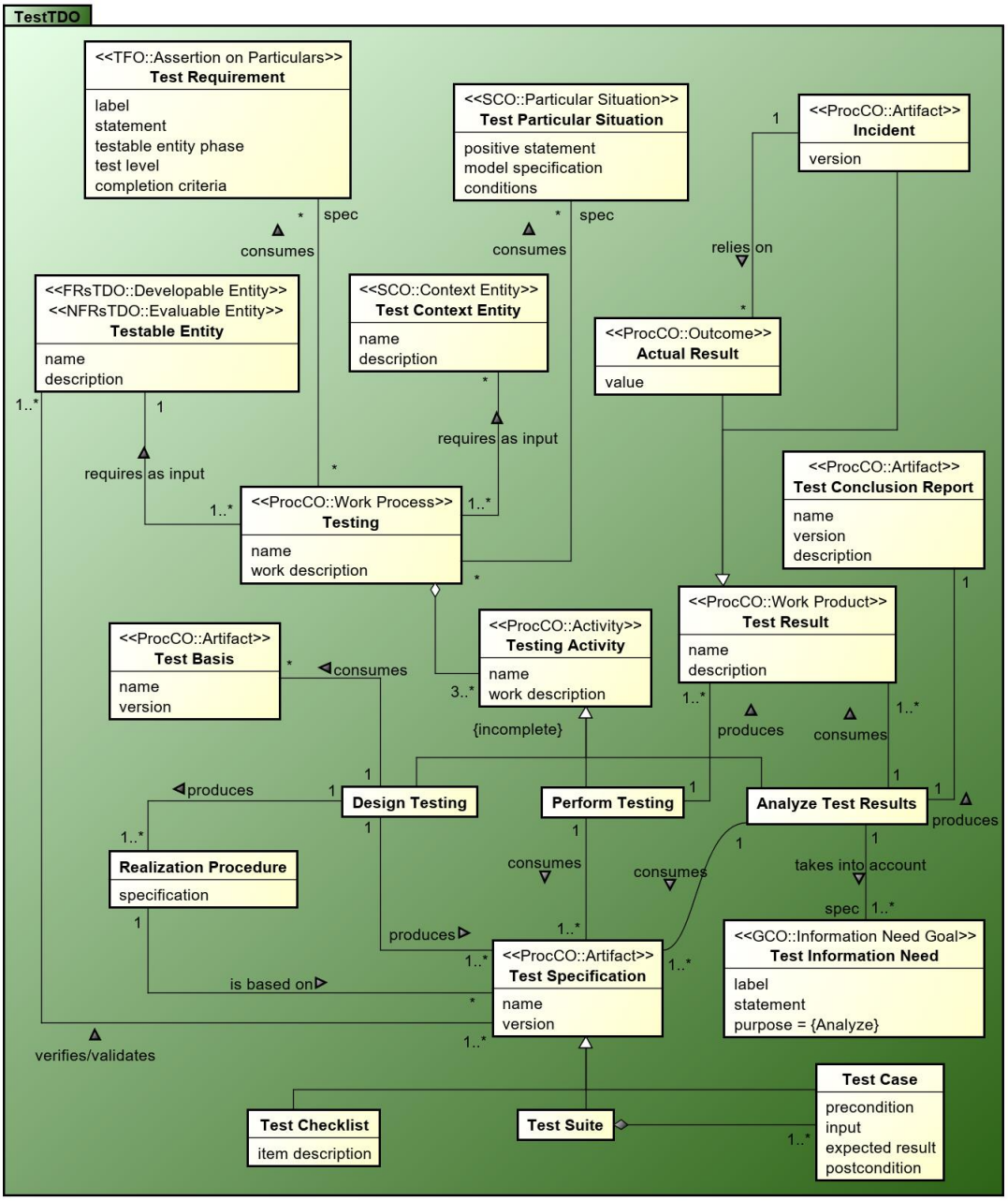


Figura 4-4: Fragmento de los términos, relaciones y propiedades de TestTDO relacionados con Productos de Trabajo/Actividades. Recordar que TFO significa *Thing Foundational Ontology*, SCO *Situation Core Ontology*, ProcCO *Process Core Ontology*, y GCO *Goal Core Ontology*.

Otra Actividad de Prueba es Realizar la Prueba, que *consume* una o más Especificaciones de Prueba para *producir* uno o más Resultados de Prueba. El Resultado de Prueba tiene semántica de Producto de Trabajo y hay dos tipos, a saber: *Actual Result* (Resultado Real) e *Incident* (Incidente). El primero es un Resultado que representa un *valor* numérico o categórico (esperado o inesperado). En cambio, un Incidente es un Artefacto o documento que informa desviaciones (por ejemplo, entre el *resultado esperado* del Caso de Prueba y el Resultado Real), anomalías (por ejemplo, un error o una falla) u otros problemas surgidos durante la actividad Realizar la Prueba. Cuando el Incidente ocurre porque hay una discrepancia entre el *resultado esperado* del Caso de Prueba y el Resultado Real, entonces este Incidente *se basa en* el Resultado Real

correspondiente. Además, en este punto es importante mencionar que TestTDO cuenta con los siguientes axiomas relacionados con lo detallado anteriormente:

Especificación del axioma A_VI:

$$\forall prt, \forall tr: [PerformTesting(prt) \land TestResult(tr) \land produces(prt, tr) \rightarrow ActualResult(tr) \veebar Incident(tr)]$$

Descripción del axioma A_VI: Para cualquier actividad de Realizar la Prueba que *produzca* un Resultado de Prueba, este resultado es, por lo tanto, un Resultado Real o un Incidente, pero no ambos al mismo tiempo.

Especificación del axioma A_VII:

$$\forall tr, \forall prt, \exists ts: [TestResult(tr) \land PerformTesting(prt) \land produces(prt, tr) \rightarrow TestSpecification(ts) \land consumes(prt, ts)]$$

Descripción del axioma A_VII: Cualquier Resultado de Prueba *producido* por una Actividad de Realizar la Prueba tiene al menos una Especificación de Prueba relacionada que es *consumida* por la misma actividad de Realizar la Prueba.

Especificación del axioma A_VIII:

$$\forall prt, \forall tc, \forall ar, \forall er, \forall val, \exists i: [PerformTesting(prt) \land TestCase(tc) \land ActualResult(ar) \land ExpectedResult(er) \land Value(val) \land consumes(prt, tc) \land partOf(er, tc) \land produces(prt, ar) \land partOf(val, ar) \land (er \neq val) \rightarrow Incident(i) \land produces(prt, i) \land reliesOn(i, ar)]$$

Descripción del axioma A_VIII: Si una actividad de Realizar la Prueba *consume* un Caso de Prueba para *producir* un Resultado Real, y el *valor* del Resultado Real no coincide con el *resultado esperado* del Caso de Prueba, entonces la actividad de Realizar la Prueba *produce* un Incidente que *se basa en* este Resultado Real.

Por otro lado, Analizar Resultados de Prueba es una Actividad de Prueba que *tiene en cuenta* la *Test Information Need* (Necesidad de Información de Prueba) específica para *producir* un Informe de Conclusión de Prueba mediante el *consumo* de uno o más Resultados de Prueba y Especificaciones de Prueba. El Informe de Conclusión de Prueba es un Artefacto que documenta el análisis de todos los Resultados de Prueba. Por ejemplo, este Artefacto podría contener detalles sobre el grado en que se alcanzaron las Metas de Prueba, el nivel de cobertura alcanzado por los Casos de Prueba ejecutados, entre otros análisis.

A continuación, se listan las definiciones de los términos, propiedades y relaciones no taxonómicas de TestTDO relacionados con Productos de Trabajo/Actividades en las Tablas 4-4, 4-5 y 4-6, respectivamente. Recordar que, como se mencionó anteriormente, estas definiciones se dejaron en inglés en esta tesis para evitar traducciones que lleven a entendimientos erróneos de las mismas.

Tabla 4-4: Definiciones en inglés de los términos principales de TestTDO relacionados con Productos de Trabajo/Actividades.

Término	Definición
Actual Result	It is a Test Result that represents a numerical or categorical value (expected or unexpected).
Analyze Test Results (synonym: Testing Analysis)	It is a Testing Activity that takes into account the specific Test Information Need in order to produce a Test Conclusion Report by consuming one or more Test Results and Test Specifications.
Design Testing (synonym: Testing Design)	It is a Testing Activity aimed at designing a set of Test Specifications (i.e., Test Cases, Test Suites and/or Test Checklists) as well as Realization Procedures.

Incident (synonym: Anomaly, Defect, Issue Report)	It is a Test Result that reports deviations (e.g. between the expected result and the Actual Result), anomalies (e.g. an error or a failure) or other arisen issues during the Perform Testing activity.
Perform Testing (synonym: Testing Realization)	It is a Testing Activity aimed at enacting a Static or Dynamic Testing. Note: This activity consumes one or more Test Specifications and produces one or more Test Results.
Realization Procedure	Arranged set of Testing Realization Method's instructions or operations, which specifies how must be performed the Perform Testing activity using the Test Specification. Note: Mainly when Dynamic Testing that involves Test Cases execution is carried out, this term (i.e., Realization Procedure) can be synonymous with Test Procedure. A Test Procedure as per ISO 29119-1 (ISO, 2013a) is a "*sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution. Test procedures include detailed instructions for how to run a set of one or more test cases selected to be run consecutively, including set up of common preconditions, and providing input and evaluating the actual result for each included test case.*".
Test Basis	Artifact consumed by a Design Testing activity for designing the Test Cases and Checklists. Note: Test Basis represents Artifacts (Becker et al., 2022) that may come from development and/or maintenance such as requirements specification, architectural design, documented source code, etc., which in turn could be linked to NFRs and FRs.
Test Case	It is a Test Specification that contains the necessary information (e.g. preconditions, inputs, expected results and postconditions) to perform mainly Dynamic Testing. Note: A Test Case, as per (ISTQB, 2021b) is "*a set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions*".
Test Checklist	It is a Test Specification that contains a list of items to be checked in order to perform mainly Static Testing.
Test Conclusion Report	It is an Artifact that documents the analysis of all Test Results.
Test Result	It is a Work Product (Becker et al., 2022) that represents both an Incident (Artifact) and an Actual Result (Outcome), which is produced by running the Perform Testing activity.
Test Specification	It is an Artifact that represents Test Checklists, Test Cases and their grouping in Test Suites. Note: Test Specification as per (ISO, 2013a) is the "*complete documentation of the test design, test cases and test procedures for a specific test item*".
Test Suite (synonym: Test Set)	It is a Test Specification that includes a set of one or more Test Cases with common constraints on their realization. Note: Test Suite in (ISTQB, 2021b) is a synonym of the Test Set term in (ISO, 2013a).
Testing (synonym: Testing Process)	It is a process (Work Process (Becker et al., 2022)) 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 (Olsina et al., 2023) of a Testable Entity.
Testing Activity	It is an Activity (Becker et al., 2022) that is formed by an interrelated set of sub-activities and tasks, aimed at designing, realizing or analyzing the testing endeavor for a particular Testable Entity.

Tabla 4-5: Definiciones en inglés de las propiedades/atributos de TestTDO relacionados con Productos de Trabajo/Actividades.

Témino	Propiedad /Atributo	Definición
Actual Result	**value**	Numerical or categorical result.
Incident	**version**	Unique identifier, which indicates the level of evolution of the Incident artifact.
Realization Procedure	**specification**	A formal or semiformal representation of a Realization Procedure.
Test Basis	**name**	Label or name that identifies the Test Basis artifact.
	version	Unique identifier, which indicates the level of evolution of the Test Basis artifact.
Test Case	**precondition**	Any kind of constraint that must evaluate to true before the Test Case's input be used in a Perform Testing activity. Note 1: The required state of a test item and its environment prior to test case execution (ISTQB, 2021b). Note 2: Test case preconditions include test environment, existing data (e.g. databases), software under test, hardware, etc. (ISO, 2013a).
	input	Test Case inputs are the data (numbers, categories, objects' instances, etc.) and/or events (e.g., button clicks) used to exercise a Testable Entity in a Perform Testing activity. Note: Inputs are the data information used to drive test execution (ISO, 2013a).
	expected result	Value and/or behavior that is expected to get after a Testable Entity is exercised using the Test Case inputs.
	postcondition	Any kind of constraint that must evaluate to true after the Test Case's input was used and the Actual Result was yielded in a Perform Testing activity.
Test Checklist	**item description**	An unambiguous textual statement describing an element of the Test Checklist.
Test Conclusion Report	**name**	Label or name that identifies the Test Conclusion Report.
	version	Unique identifier, which indicates the level of evolution of the Test Conclusion Report.
	description	An unambiguous textual statement describing the Test Conclusion Report.
Test Result	**name**	Label or name that identifies the Test Result work product.
	description	An unambiguous textual statement describing the Test Result.
Test Specification	**name**	Label or name that identifies the Test Specification artifact.
	version	Unique identifier, which indicates the level of evolution of the Test Specification artifact.

Testing	**name**	Label or name that identifies the Testing work process.
	work description	Specification of what to do for achieving the objective of a Testing Work Process. Note 1: The specification of what to do is a set of general actions, which implies both Testing Activities and Tasks. It represents what should be done instead of how it should be performed. Note 2: The specification of the work description can be formal, semi-formal or informal.
Testing Activity	**name**	Label or name that identifies the Testing Activity.
	work description	Specification of what to do for achieving the objective of a Testing Activity. Note: See note 1 and 2 in the work description attribute for the Testing term.

Tabla 4-6: Definiciones en inglés de las relaciones no taxonómicas de TestTDO relacionados con Productos de Trabajo/Actividades.

Relación no taxonómica	Definición
consumes (x6)	An Analyze Test Results activity consumes one or more Test Result as work product.
	A Perform Testing activity consumes one or more Test Specification as artifact.
	An Analyze Test Results activity consumes one or more Test Specification as artifact.
	A Design Testing activity consumes Test Basis, if any.
	A Testing work process can consume one or more Test Particular Situation's model specifications as Artifacts.
	A Testing work process can consume one or more Test Requirement's specifications as Artifacts.
is based on	A Realization Procedure is based on none or several Test Specifications.
produces (x4)	A Perform Testing activity produces a Test Result as work product.
	An Analyze Test Results activity produces a Test Conclusion Report as artifact.
	A Design Testing activity produces one or more Realization Procedure.
	A Design Testing activity produces one or more Test Specification as artifact.
relies on	An Incident detected in a Perform Testing activity relies on none or several Actual Results.
requires as input (x2)	A Testing process needs a Testable Entity as input.
	A Testing process needs none or many Test Context Entities as input.
takes into account	An Analyze Test Results activity consumes one or more Test Information Needs as a specification.

verifies/validates	A Test Specification verifies/validates one or more Testable Entities. In turn, a Testable Entity is verified/validated by one or more Test Specifications.

A seguir, se listan otros axiomas de TestTDO que incluyen conceptos relacionados con Productos de Trabajo/Actividades:

Especificación del axioma A_IX:

$$\forall p, \exists te, \exists ta: [Testing(p) \wedge TestableEntity(te) \wedge requiresAsInput(p, te) \rightarrow TestingActivity(ta) \wedge partOf(ta, p) \wedge requiresAsInput(ta, te)]$$

Descripción del axioma A_IX: Si un Proceso de Prueba *requiere como entrada* una Entidad Comprobable, entonces algunas de sus Actividades de Prueba también la requieren y la usan como entrada.

Especificación del axioma A_X:

$$\forall p, \forall tce, \exists ta: [Testing(p) \wedge TestContextEntity(tce) \wedge requiresAsInput(p, tce) \rightarrow TestingActivity(ta) \wedge partOf(ta, p) \wedge requiresAsInput(ta, tce)]$$

Descripción del axioma A_X: Si un Proceso de Prueba *requiere como entrada* una Entidad de Contexto de Prueba, entonces algunas de sus Actividades de Prueba también la requieren y la usan como entrada.

Especificación del axioma A_XI:

$$\forall p, \forall trs, \exists ta: [Testing(p) \wedge SpecificationOfTestRequirement(trs) \wedge consumes(p, trs) \rightarrow TestingActivity(ta) \wedge partOf(ta, p) \wedge consumes(p, trs)]$$

Descripción del axioma A_XI: Si un Proceso de Prueba *consume* la especificación de un Requisito de Prueba, algunas de sus Actividades de Prueba también la consumen.

Especificación del axioma A_XII:

$$\forall p, \forall tps, \exists ta: [Testing(p) \wedge SpecificationOfTestParticularSituation(tps) \wedge consumes(p, tps) \rightarrow \exists ta: TestingActivity(ta) \wedge partOf(ta, p) \wedge consumes(p, tps)]$$

Descripción del axioma A_XII: Si un Proceso de Prueba *consume* la especificación de una Situación Particular de Prueba, algunas de sus Actividades de Prueba también la consumen.

Conceptos de TestTDO relacionados con Actividades/Métodos/Agentes

Para cubrir el alcance relacionado con actividades, métodos y agentes de prueba, TestTDO tiene términos como *Perform Static Testing* (Realizar Pruebas Estáticas), *Perform Dynamic Testing* (Realizar Pruebas Dinámicas), Método de Realización de Pruebas, Método de Diseño de Prueba, *Testing Agent* (Agente de Prueba), *Testing Role* (Rol de Prueba) y *Testing Tool* (Herramienta de Prueba) como se muestra en la Figura 4-5. Recordar que Realizar la Prueba es una Actividad de Prueba que tiene como objetivo llevar a cabo pruebas estáticas o dinámicas (ver definición en Tabla 4-4). Realizar Pruebas Estáticas tiene el objetivo de verificar una Entidad Comprobable contra una o más Especificaciones de Prueba sin la ejecución de su código de software (si lo hubiere). En cambio, Realizar Pruebas Dinámicas tienen como objetivo verificar/validar una Entidad Comprobable contra una o más Especificaciones de Prueba con la ejecución de su código de software.

Tipos más específicos de actividades de Realizar Pruebas Dinámicas son *Perform Non-Functional Dynamic Testing* (Realizar Pruebas Dinámicas No Funcionales) y *Perform Functional Dynamic Testing* (Realizar Pruebas Dinámicas Funcionales). La primera tiene como objetivo evaluar el cumplimiento de Requisitos No Funcionales de

una Entidad Comprobable (Becker et al., 2019; Olsina et al., 2023), mientras que la segunda tiene como objetivo verificar/validar el cumplimiento de Requisitos Funcionales de una Entidad Comprobable (Becker et al., 2019; Tebes, Olsina, et al., 2020a). TestTDO contiene 2 axiomas que utilizan estos conceptos, a saber:

Especificación del axioma A_XIII:

$$\forall pfdt, \forall ts, \exists tb, \exists tdm, \exists fr, \exists dt: [PerformFunctionalDynamicTesting(pfdt) \wedge TestSpecification\ (ts) \wedge consumes(pfdt, ts) \rightarrow TestBasis(tb) \wedge TestingDesignMethod(tdm) \wedge FunctionalRequirement(fr) \wedge Design\ Testing(dt) \wedge isLinkedTo(tb, fr) \wedge consumes(dt, tb) \wedge isAssignedTo(tdm, dt) \wedge produces(dt, ts)]$$

Descripción del axioma A_XIII: Para cualquier actividad de Realizar Pruebas Dinámicas Funcionales que *consuma* una Especificación de Prueba, esta especificación se basa, por lo tanto, en un Requisito Funcional.

Especificación del axioma A_XIV:

$$\forall pnfdt, \forall ts, \exists tb, \exists tdm, \exists nfr, \exists dt: [PerformNonFunctionalDynamicTesting(pnfdt) \wedge TestSpecification\ (ts)\ \wedge consumes(pnfdt, ts) \rightarrow\ TestBasis(tb) \wedge TestingDesignMethod(tdm) \wedge NonFunctionalRequirement(nfr) \wedge DesignTesting(dt) \wedge isLinkedTo(tb, nfr) \wedge consumes(dt, tb) \wedge isAssignedTo(tdm, dt) \wedge produces(dt, ts)]$$

Descripción del axioma A_XIV: Para cualquier actividad de Realizar Pruebas Dinámicas No Funcionales que *consuma* una Especificación de Prueba, esta especificación se basa, por lo tanto, en un Requisito No Funcional.

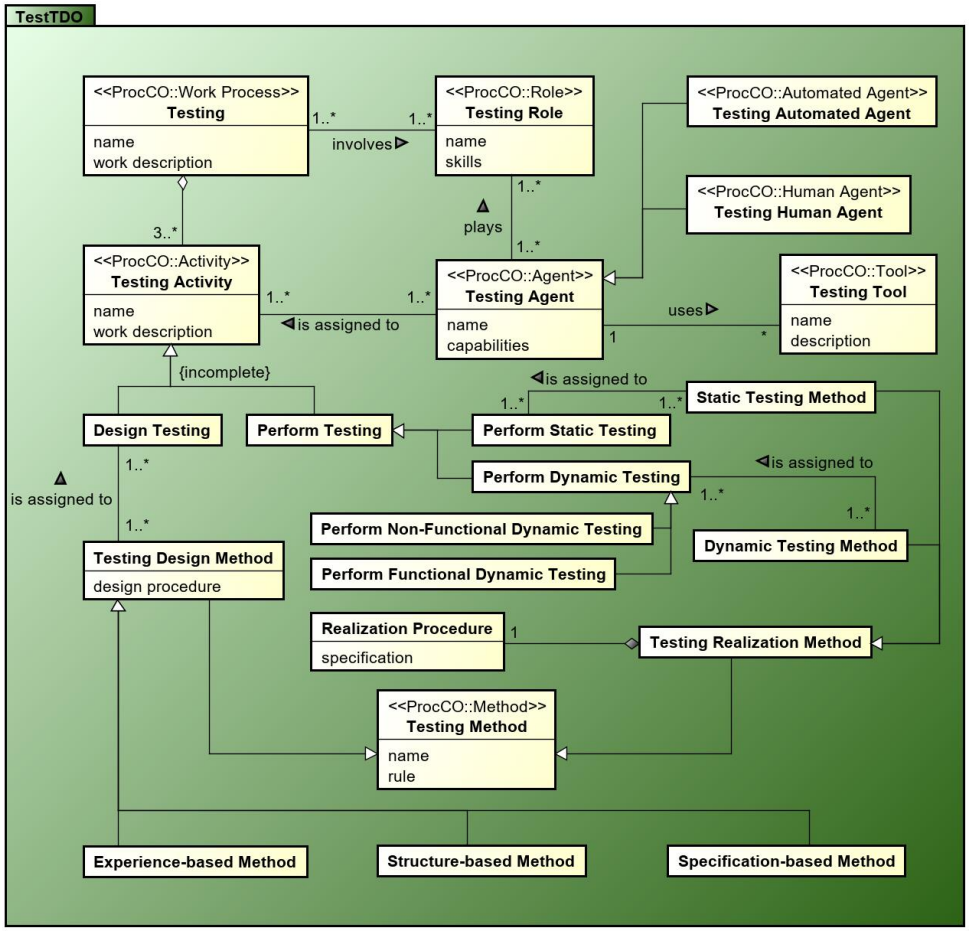


Figura 4-5: Fragmento de los términos, relaciones y propiedades de TestTDO relacionados con Actividades/Métodos/Agentes. Recordar que ProcCO significa *Process Core Ontology*.

Como se muestra en la Figura 4-5, un *Static/Dynamic Testing Method* (Método de Prueba Estática/Dinámica) *se asigna a* una o más actividades de Realizar la Prueba Estática/Dinámica. Ejemplos de Métodos de Pruebas Estáticas son *walkthroughs*, revisiones técnicas, inspección, entre otros. Estos Métodos de Prueba Estática/Dinámica son tipos de Métodos de Realización de Pruebas. Un Método de Realización de Pruebas es un *Testing Method* (Método de Pruebas) para una tarea incluida en una actividad de Realizar la Prueba, que incluye un Procedimiento de Realización.

Adicionalmente, el Método de Prueba (o Técnica de Prueba) se define como una forma específica y particular de realizar los pasos especificados para una tarea incluida en una Actividad de Prueba. La forma específica y particular de un Método de Prueba, es decir, cómo se deben realizar los pasos especificados en una tarea de prueba, está representada por un *procedimiento* (por ejemplo, *procedimiento de diseño* o Procedimiento de Realización) y *reglas*.

En este punto es importante remarcar que TestTDO adopta un patrón conceptual de ProcessCO que ninguna de las 12 ontologías seleccionadas en la RSL tiene excepto ROoST. Este patrón intenta hacer una clara separación entre el "qué" (actividades y tareas) y el "cómo" (métodos y procedimientos). Tenga en cuenta que, en TestTDO, cada Actividad de Prueba *tiene asignado* un Método de Prueba (excepto por la actividad de Analizar Resultados de Prueba que no incluye explícitamente su método asociado para no sobrecargar el modelo).

Al igual que el Método de Realización de Pruebas, el Método de Diseño de Pruebas es otro tipo de Método de Prueba, pero *se asigna a* actividades de Diseñar la Prueba. Además, este tiene un *procedimiento de diseño* que especifica cómo se debe realizar la actividad de Diseñar la Prueba utilizando la Base de Prueba, si la hubiera. Hay tres tipos de Métodos de Diseño de Pruebas, a saber: *Specification-based Method* (Método basado en la Especificación, también conocido como caja negra), *Structure-based Method* (Método basado en la Estructura, también conocido como caja blanca) y *Experience-based Method* (Método basado en la Experiencia).

Un Método basado en la Especificación siempre utiliza una Base de Prueba cuando se ejecuta la actividad de Diseñar la Prueba para derivar Especificaciones de Prueba sin hacer referencia a la estructura interna de la Entidad Comprobable, tal como lo establece el siguiente axioma:

Especificación del axioma A_XV:

$$\forall dt, \forall spbm, \forall ts, \exists tb, \exists te: [DesignTesting(dt) \wedge Specification\_basedMethod(spbm) \wedge TestSpecification(ts) \wedge isAssignedTo(spbm, dt) \wedge produces(dt, ts) \rightarrow TestBasis(tb) \wedge TestableEntity(te) \wedge consumes(dt, tb) \wedge \neg requiresAsInput(dt, te)]$$

Descripción del axioma A_XV: Si *se asigna* un Método basado en la Especificación a una Actividad de Prueba de Diseñar la Prueba que *produce* una Especificación de Prueba, siempre *consume* una Base de Prueba que es utilizada por el Método basado en la Especificación sin usar la estructura interna de la Entidad Comprobable.

Además, tipos más específicos de Métodos basado en la Especificación son: árbol de clasificación, pruebas de escenarios, pruebas aleatorias, pruebas de transición de estado, entre otros (ISO, 2015), que no se muestran en la Figura 4-5 ya que los términos TestTDO están en el nivel de dominio superior.

Por otro lado, ejemplos de Métodos basado en la Estructura pueden ser pruebas de sentencias, pruebas de decisión, pruebas de cobertura de decisión y condición modificada, entre otros (ISO, 2015). Un Método basado en la Estructura es un Método de Diseño de

Prueba que siempre utiliza la estructura interna de la Entidad Comprobable y, a veces, también utiliza una Base de Prueba cuando se ejecuta la actividad de Diseñar la Prueba para derivar Especificaciones de Prueba. El siguiente axioma está relacionado con lo mencionado anteriormente:

Especificación del axioma A_XVI:

$$\forall dt, \forall stbm, \forall ts, \exists te: [DesignTesting(dt) \wedge Structure\_basedMethod(stbm) \wedge TestSpecification(ts) \wedge isAssignedTo(stbm, dt) \wedge produces(dt, ts) \rightarrow TestableEntity(te) \wedge requiresAsInput(dt, te)]$$

Descripción del axioma A_XVI: Si *se asigna* un Método basado en la Estructura a una Actividad de Prueba de Diseñar la Prueba que *produce* una Especificación de Prueba, siempre *se requiere como entrada* la estructura interna de la Entidad Comprobable que es utilizada por el Método basado en la Estructura.

Por último, pero no menos importante, un Método basado en la Experiencia también es un Método de Prueba, pero utiliza el conocimiento, la experiencia y la intuición del *Testing Human Agent* (Agente Humano de Prueba) cuando se ejecuta la actividad de Diseñar la Prueba para derivar Especificaciones de Prueba. Algunos tipos de Métodos basados en la Experiencia citados en la literatura son *error guessing* (ISO, 2015) y pruebas exploratorias (ISTQB, 2021b).

Como muestra la Figura 4-5, TestTDO también tiene conceptos relacionados con agentes. Un Agente de Prueba es un Agente (Becker et al., 2022) que *se asigna a* una Actividad de Prueba y *juega* un determinado Rol de Prueba, el cual implica un conjunto de habilidades. Además, este agente puede *utilizar* una Herramienta de Prueba, la cual puede automatizar total o parcialmente la ejecución de un Método de Prueba. Tipos específicos de Agentes de Prueba pueden ser Agente Humano de Prueba y *Testing Automated Agent* (Agente Automatizado de Prueba). TestTDO tiene el siguiente axioma que implica conceptos de agentes:

Especificación del axioma A_XVII:

$$\forall p, \forall tr, \exists ta: [Testing(p) \wedge TestingRole(tr) \wedge involves(p, tr) \rightarrow TestingActivity(ta) \wedge partOf(ta, p) \wedge involves(p, tr)]$$

Descripción del axioma A_XVII: Si un Proceso de Prueba *involucra* un Rol de Prueba, entonces algunas de sus Actividades de Prueba también lo involucran.

A continuación, se listan las definiciones de los términos, propiedades y relaciones no taxonómicas de TestTDO relacionados con Actividades/Métodos/Agentes en las Tablas 4-7, 4-8 y 4-9, respectivamente. Recordar que, como se mencionó anteriormente, estas definiciones se dejaron en inglés en esta tesis para evitar traducciones que lleven a entendimientos erróneos de las mismas.

Tabla 4-7: Definiciones en inglés de los términos principales de TestTDO relacionados con Actividades/Métodos/Agentes.

Término	Definición
Dynamic Testing Method	It is a Testing Realization Method for a task included in a Perform Dynamic Testing activity.
Experience-based Method	It is a Testing Design Method that uses the Testing Human Agent's knowledge, expertise and intuition while enacting the Design Testing activity for deriving Test Specifications. Note: Examples of Experience-based Methods and practices (as per (ISO, 2013a)) are Error Guessing, Exploratory Testing, Software Attacks and Ad Hoc Testing, under the name of "*experience-based test design techniques*".

Perform Dynamic Testing (synonym: Dynamic Testing)	It is a Perform Testing activity aimed at verifying/validating a Testable Entity against one or more Test Specifications with the execution of its software code. Note 1: Dynamic Testing as per ISTQB v. 3.5 (ISTQB, 2021b) is "*Testing that involves the execution of the test item.*" Note 2: Dynamic Testing as per ISO 29119-1 (ISO, 2013a) is "*testing that requires the execution of the test item*".
Perform Functional Dynamic Testing (synonym: Functional Dynamic Testing)	It is a Dynamic Testing activity aimed at verifying/validating the compliance of a Testable Entity with Functional Requirements (Tebes, Olsina, et al., 2020a). Note: Functional Testing as per (ISTQB, 2021b) is "*testing conducted to evaluate the compliance of a component or system with functional requirements*".
Perform Non-functional Dynamic Testing (synonym: Non-functional Dynamic Testing)	It is a Dynamic Testing activity aimed at appraising the compliance of a Testable Entity with Non-Functional Requirements (Olsina et al., 2023). Note: Non-Functional Testing as per (ISTQB, 2021b) is "*testing conducted to evaluate the compliance of a component or system with non-functional requirements*".
Perform Static Testing (synonym: Static Testing)	It is a Perform Testing activity aimed at checking a Testable Entity against one or more Test Specifications without the execution of its software code (if any). Note 1: Static Testing as per (ISTQB, 2021b) is "*testing a work product without code being executed*". Note 2: Static Testing as per (ISO, 2013a) is "*testing in which a test item is examined against a set of quality or other criteria without code being executed*".
Specification-based Method (synonym: Black-box Method)	It is a Testing Design Method that always uses a Test Basis while enacting the Design Testing activity for deriving Test Specifications without referring to the internal structure of the Testable Entity. Note: Examples of Specification-based Methods are Boundary Value Analysis, State Transition Testing, and Decision Table Testing, among others (covered in (ISO, 2015) under the name of "*specification-based test design techniques*").
Static Testing Method	It is a Testing Realization Method for a task included in a Perform Static Testing activity. Note: Static Testing Method examples are Walkthrough, Technical Review, Inspection, etc.
Structure-based Method (synonym: White-box Method)	It is a Testing Design Method that uses the internal structure of the Testable Entity, and sometimes also uses a Test Basis, while enacting the Design Testing activity for deriving Test Specifications. Note: Examples of Structure-based Methods are Branch Testing, Condition Testing and Data Flow Testing, among others (covered in (ISO, 2015) under the name of "*structure-based test design techniques*").
Testing Agent	It is an Agent (Becker et al., 2022) –i.e., a work resource- assigned to a Testing Activity in compliance with one or more Testing Roles. Note: A Testing Agent may use Testing Tools.
Testing Automated Agent	It is a Testing Agent, which is not a human being.
Testing Design Method	It is a Testing Method for a task included in a Design Testing activity. Note: A Testing Design Method may use a Test Basis, while enacting the Design Testing activity.
Testing Human Agent	It is a Testing Agent, which is a human being.

Testing Method (synonym: Testing Technique)	A specific and particular way to perform the specified steps for a task included in a Testing Activity. Note 1: Testing Method is a Method as per (Becker et al., 2022). Note 2: 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 design or Realization Procedure and rules.
Testing Realization Method	It is a Testing Method for a task included in a Testing Realization activity.
Testing Role	It is a Role (Becker et al., 2022) that implies a set of testing skills. Note: Testing skills are abilities, competencies and responsibilities involved in the Testing process.
Testing Tool	It is a Tool (Becker et al., 2022) that partially or totally accomplishes the automatic execution of a Testing Method.

Tabla 4-8: Definiciones en inglés de las propiedades/atributos de TestTDO relacionados con Actividades/Métodos/Agentes.

Témino	Propiedad /Atributo	Definición
Testing Agent	**name**	Label or name that identifies the Testing Agent.
	capabilities	Set of abilities that the Testing Agent has.
Testing Design Method	**design procedure**	Arranged set of Testing Design Method's instructions or operations, which specifies how must be performed the Design Testing activity using the Test Basis, if any.
Testing Method	**name**	Label or name that identifies the Testing Method.
	rule	Set of principles, conditions, heuristics, axioms, etc. associated to the design procedure or Realization Procedure.
Testing Role	**name**	Label or name that identifies the Testing Role.
	skills	Set of capabilities, competencies and responsibilities of a role.
Testing Tool	**name**	Label or name that identifies the Testing Tool.
	description	An unambiguous textual statement describing the Testing Tool.

Tabla 4-9: Definiciones en inglés de las relaciones no taxonómicas de TestTDO relacionados con Actividades/Métodos/Agentes.

Relación no taxonómica	Definición
involves	A Testing Work Process involves one or more Testing Roles. In turn, a Testing Role may participate in one or more Testing Work Process.
is assigned to (x4)	A Static Testing Method is assigned to one or more Perform Static Testing activities.
	A Dynamic Testing Method is assigned to one or more Perform Dynamic Testing activities.
	A Testing Design Method is assigned to one or more Design Testing activities.
	A Testing Agent is assigned to one or more Testing Activities.

plays	A Testing Agent plays one or more Testing Roles. In turn, a Testing Role is played by one or more Testing Agents.
uses	A Testing Agent uses Testing Tools, if any.

Recordar que, según la Figura 4-1, existe un bucle en A2 que involucra la actividad Diseñar y Construir una Versión del Artefacto en la cual se puede considerar añadir nuevos requisitos a la última versión del artefacto. Para obtener la versión actual de TestTDO se realizaron más de una iteración, pero en este capítulo solo se ilustró el resultado de la última de ellas. Por lo tanto, en este punto, finaliza la actividad A2.

4.3.3.3 A3 Ejecutar Verificación y Validación (VyV)

Una vez finalizada A2 la siguiente actividad a realizar es A3. La primera subactividad de A3 es Seleccionar Procesos y Métodos de VyV. Las actividades de VyV son similares, pero con objetivos diferentes. La validación tiene como objetivo demostrar que el artefacto satisface el uso previsto, mientras que la verificación tiene como objetivo verificar si el artefacto cumple adecuadamente con los requisitos especificados. En otras palabras, la validación asegura que "se construyó el artefacto correcto", mientras que la verificación asegura que "se construyó el artefacto correctamente".

En la Figura 4-6 se ilustra una taxonomía de actividades de VyV (la cual fue adaptada de la figura A.1 de (ISO, 2013a)). Esta jerarquía muestra tres actividades en el primer nivel, a saber: *Testing*, *Formal Verification* y *V&V Analysis*. A su vez, *Testing* tiene dos subactividades: Pruebas Estáticas y Pruebas Dinámicas. Para este último se agregaron subactividades de Pruebas Dinámicas Funcionales y No Funcionales. Por otro lado, *V&V Analysis* tiene subactividades como Demostración, Valoración, Simulación y, Medición y Evaluación.

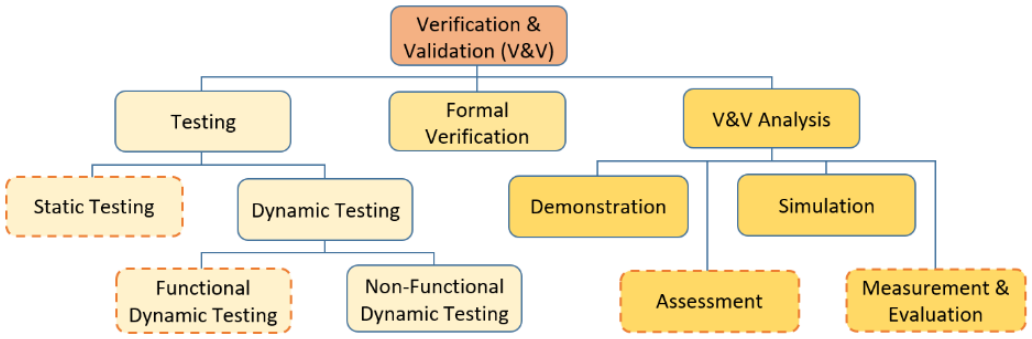


Figura 4-6: Taxonomía de actividades de VyV. Figura tomada de (Tebes, Olsina, et al., 2021).

Considerando lo establecido en el AR#10, para verificar y validar TestTDO se seleccionaron las actividades resaltadas con línea punteada en la Figura 4-6. En la actividad de Pruebas Estáticas se realizaron dos tipos de verificaciones, a saber: i) verificación estática del alcance previsto para TestTDO usando sus CQs; y ii) verificación de la coherencia semántica entre las relaciones de TestTDO y las ontologías de FCD-OntoArch que la enriquecen.

Además, se realizó una verificación dinámica (Pruebas Dinámicas Funcionales) de las CQs de TestTDO. Notar que para poder llevar a cabo este tipo de verificación fue necesario implementar la ontología. Por otro lado, para la actividad de Valoración, se hizo una prueba de concepto que instancia los términos de TestTDO en un proyecto de prueba académico de una aplicación de figuras geométricas, con el objetivo de validar si TestTDO es capaz de representar situaciones concretas del mundo. A su vez, se realizó una validación por parte de tres expertos en el dominio de las pruebas de software.

Finalmente, para la actividad de Medición y Evaluación, se utilizó la estrategia de evaluación GOCAME (Becker et al., 2015; Olsina & Becker, 2017) para verificar si TestTDO cumple con los mismos Requisitos No Funcionales que se evaluaron en las 12 ontologías seleccionadas en la RSL (recordar AR#9). A continuación, se describen detalladamente cada una de estas actividades seleccionadas de VyV aplicadas a TestTDO, las cuales fueron llevadas a cabo como parte de la actividad Ejecutar la VyV de A3.

Pruebas Estáticas

Con respecto a la verificación estática, primero se verificó la conceptualización de TestTDO frente a las CQs. Recordar que estas representan los requisitos relacionados con el alcance y están listadas en la Sección 4.3.3.1. El objetivo a alcanzar es verificar que todas las CQs fueran abordadas por alguno de los términos, propiedades, relaciones y/o axiomas de TestTDO. En esta dirección, se produjo una matriz de verificación (ver Tabla 4-10) utilizando un Método basado en la Especificación (caja negra). Para diseñar la matriz se utilizó como Base de Prueba las CQs y los elementos que tiene cualquier ontología, es decir, términos, propiedades, relaciones y axiomas. Notar que esta matriz tiene semántica de Lista de Verificación y contiene una fila por cada CQ en la cual se registra qué término, propiedad, relación o axioma se asocia o cubre la CQ. La matriz completa para todas las CQs se encuentra en el Apéndice C. Además, los códigos que identifican los axiomas en esta matriz se corresponden con los presentados en la Sección 4.3.3.2.

Tabla 4-10: Extracto de la matriz de verificación estática del alcance de TestTDO. Se puede acceder a la matriz de verificación completa con todas las CQs en el Apéndice C.

CQs	**Términos, relaciones y *propiedades***	**Axiomas**
CQ3. ¿Cuáles son los Productos de Trabajo producidos por una actividad de Realizar la Prueba?	Realizar la Prueba es-una Actividad de Prueba Realizar la Prueba produce Resultado de Prueba Resultado Real es-un Resultado de Prueba Incidente es-un Resultado de Prueba	A_VI, A_VII, A_VIII
CQ8. ¿Cuál es el conjunto mínimo de Actividades de Prueba incluidas en un Proceso de Prueba?	Actividad de Prueba es-parte-de Proceso de Prueba Diseñar la Prueba es-una Actividad de Prueba Realizar la Prueba es-una Actividad de Prueba Analizar Resultados de Prueba es-una Actividad de Prueba	A_V
CQ25. Para un Proyecto de Prueba que operacionaliza una Meta de Prueba, ¿tiene este proyecto una Estrategia de Prueba asociada que ayuda a alcanzar el propósito de la Meta de Prueba?	Proyecto de Prueba operacionaliza Meta de Prueba Proyecto de Prueba asocia Estrategia de Prueba Estrategia de Prueba ayuda a alcanzar Meta de Prueba Meta de Prueba tiene la propiedad llamada *propósito*	A_IV

Por ejemplo, la CQ3 establece "¿Cuáles son los Productos de Trabajo producidos por una actividad de Realizar la Prueba?". Si se observa la Figura 4-4, podemos notar que Realizar la Prueba es una Actividad de Prueba que produce Resultados de Prueba (con semántica de Producto de Trabajo). Además, Resultado Real e Incidente son tipos específicos de Resultado de Prueba, por lo tanto, una actividad de Realizar la Prueba produce Resultados Reales o Incidentes. A su vez, si se consideran los axiomas de

TestTDO, se puede determinar que A_VI, A_VII, A_VIII están relacionados con la CQ3. Por ejemplo, A_VI establece que cualquier Resultado de Prueba producido por una actividad de Realizar la Prueba es un Resultado Real o un Incidente, pero no ambos al mismo tiempo.

Por otro lado, la CQ25 especifica "Para un Proyecto de Prueba que operacionaliza una Meta de Prueba, ¿tiene este proyecto una Estrategia de Prueba asociada que ayuda a alcanzar el propósito de la Meta de Prueba?". Para cubrir el alcance de la CQ25, TestTDO cuenta con un patrón conceptual que se hereda de las ontologías del nivel central (*core*). Este patrón relaciona los términos Estrategia de Prueba, Proyecto de Prueba y Meta de Prueba. Además, TestTDO tiene el axioma A_IV, el cual establece que todos los Proyectos de Prueba operacionalizan una Meta de Prueba y asocian una Estrategia de Prueba si y solo si esta Estrategia de Prueba ayuda a alcanzar la Meta de Prueba operacionalizada.

Adicionalmente, se utilizó otro Método de Diseño de Pruebas para verificar estáticamente TestTDO. El objetivo era verificar la consistencia semántica de las relaciones de TestTDO contra las relaciones correspondientes de las ontologías de niveles superiores que pertenecen a la arquitectura FCD-OntoArch. En este caso, se realizó una prueba de integración en la que se consideró la siguiente Situación Particular de Prueba: i) las Entidades Comprobables son las dependencias entre TestTDO y las ontologías SituationCO y ProcessCO de FCD-OntoArch (recordar Figura 3-8); y ii) tanto SituationCO como ProcessCO son Entidades de Contexto de Prueba en esta situación. Tenga en cuenta que una dependencia entre TestTDO y otra ontología de nivel central (*core*) representa una relación de TestTDO que es heredada por otra relación de la ontología de más alto nivel.

En la actividad de Diseñar la Prueba se utilizó un Método de Diseño de Pruebas de caja blanca (i.e., un Método basado en la Estructura) para producir la Lista de Verificación que se muestra en la Tabla 4-11. Para diseñar la Lista de Verificación, este método de caja blanca usa como Base de Prueba la estructura interna de las Entidades Comprobable (recuerde que las Entidades Comprobables son las dependencias entre TestTDO y SituationCO y ProcessCO). Se identificaron estas dependencias analizando las conceptualizaciones de TestTDO, SituationCO y ProcessCO. Como se puede observar en la Tabla 4-11, las primeras tres columnas documentan las relaciones identificadas (es decir, las dependencias) y los términos principales relacionados con sus estereotipos. La lista completa para todas las relaciones se encuentra en el Apéndice D. Notar que estas relaciones pertenecen a TestTDO en su versión inicial (v1.0), la cual está documentada en (Tebes, Olsina, et al., 2020b). En esta tesis, en la Sección 4.3.3.2, se ha ilustrado TestTDO en su última versión (v1.3).

Después de identificar todas las dependencias, durante la actividad Realizar Pruebas Estáticas se inspeccionaron cada una de ellas para verificar su coincidencia semántica con alguna relación de las ontologías de niveles superiores. La Tabla 4-11 contiene un ejemplo en el que se generó un Incidente y, por lo tanto, el Resultado Real tiene el valor "fallo". La Figura 4-7 resalta en rojo la relación de TestTDO v1.0 llamada "*is in particular situation with*", la cual tiene una falta de coincidencia semántica con la relación de SituationCO resaltada en verde denominada "*is surrounded by*". Además, al analizar este incidente, se detectó que faltaba la relación "*influences*" en TestTDO v1.0.

Tabla 4-11: Extracto de la Lista de Verificación utilizada para inspeccionar la consistencia semántica de las relaciones de TestTDO v1.0 frente a las relaciones que las enriquecen de las ontologías de niveles superiores (por ejemplo, la ontología SituationCO). Se puede acceder a la Lista de Verificación completa con todas las relaciones en el Apéndice D.

Relación de TestTDO	**Término 1 de TestTDO <<Estereotipo>>**	**Término 2 de TestTDO <<Estereotipo>>**	**Inspeccionar**	**Resultado Real (paso/fallo)**
is in a particular situation with	Entidad Comprobable <<Entidad Evaluable/Desarrollable de NFRsTDO/FRsTDO>>	Entidad de Contexto de Prueba <<Entidad de Contexto de SituationCO>>	¿Existe una coincidencia semántica entre la relación denominada "*is in a particular situation with*" de TestTDO con alguna relación de los conceptos de Entidad Evaluable/Desarrollable (de NFRsTDO/FRsTDO) y Entidad de Contexto (de SituationCO)?	**fallo**
is derived in	Meta de Prueba <<Meta Específica de GoalCO>>	Requisito de Prueba <<Aserción de ThingFO>>	¿Existe una coincidencia semántica entre la relación denominada "*is derived in*" de TestTDO con alguna relación de los conceptos de Meta Específica (de GoalCO) y Aserción sobre Particulares (de ThingFO)?	**paso**

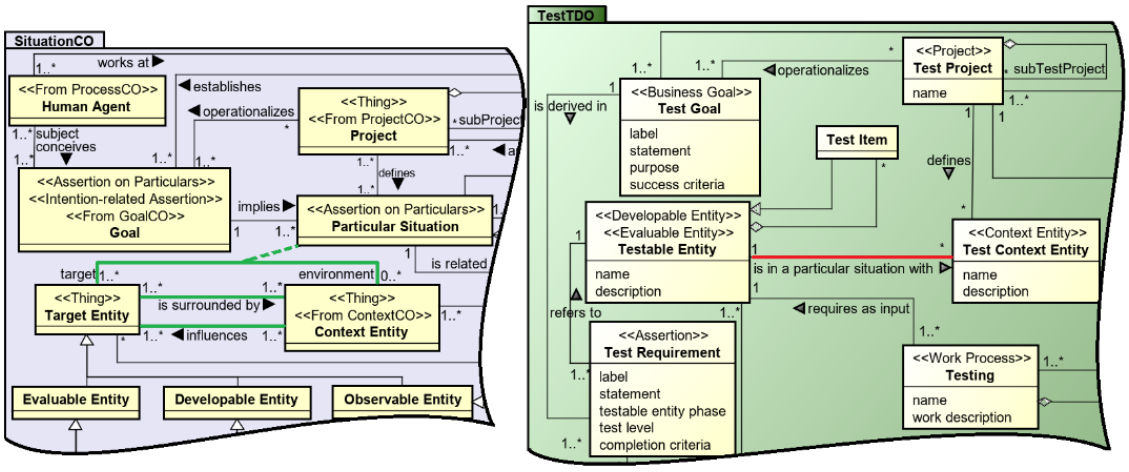


Figura 4-7: Falta de coincidencia semántica entre la relación resaltada en rojo de TestTDO v1.0 ("*is in particular situation with*") y la relación resaltada en verde de SituationCO ("*is surrounded by*").

Analizando las definiciones de ambas relaciones se encontró una inconformidad semántica, la cual condujo a la detección de otro problema de TestTDO v1.0. En el patrón conceptual de SituaciónCO que relaciona los términos Situación Particular, Entidad Objetivo y Entidad de Contexto, la Situación Particular representa una asociación entre la Entidad Objetivo y la Entidad de Contexto. A su vez, la Entidad Objetivo está rodeada por ("*is surrounded by*") Entidades de Contexto y, las Entidades de Contexto, influencian "*influences*" a la Entidad Objetivo. Sin embargo, este patrón conceptual ontológico de SituationCO no se veía reflejado en TestTDO v1.0.

En total se detectaron 6 Incidentes en las relaciones de TestTDO v1.0 mediante el uso de este Método de Diseño de Pruebas. Es importante señalar que esta actividad de Prueba Estática permitió verificar parcialmente el AR#7, el cual establece que "…algunos términos también deben estar relacionados con las ontologías SituationCO y ProcessCO en el nivel *core*…". Además, se llevó a cabo esta actividad para verificar únicamente las relaciones que asociaban conceptos enriquecidos de TestTDO v1.0 (es decir, con estereotipos). En otras palabras, esta actividad involucro en cierto grado la verificación de cada concepto de TestTDO estereotipado con términos de ontologías de niveles superiores de FCD-OntoArch. A medida que se inspeccionaba cada relación se comprobaba que la semántica entre el término que enriquece (es decir, el que indica el estereotipo) y el término enriquecido se corresponde adecuadamente. Sin embargo, esto se realizó de manera no sistemática y, por lo tanto, no se encuentra documentado.

Pruebas Dinámicas Funcionales

Para poder llevar a cabo las Pruebas Dinámicas sobre TestTDO se tuvo que implementar está ontología. Para alcanzar este objetivo, se utilizó el lenguaje OWL y la herramienta Protégé. Además, se utilizaron algunas guías para transformar la ontología de UML a OWL (Haasjes, 2019; Vo & Hoang, 2020), las cuales fueron muy útiles para producir la versión en OWL de TestTDO. El lector puede acceder a su implementación en http://bit.ly/TestTDO_OWL.

Una vez obtenida la implementación de TestTDO, se utilizó un Método basado en la Especificación (caja negra) para diseñar Casos de Prueba. Para producir los casos, se utilizó como Bases de Prueba la conceptualización de TestTDO y también sus requisitos funcionales relacionados al alcance (i.e., las CQs). Otra Base de Prueba utilizada proviene de las instancias del proyecto de prueba académico de una aplicación de figuras geométricas (ver Apéndice E), las cuales fueron útiles para diseñar los resultados esperados de los Casos de Prueba. Notar que este proyecto es el mismo que se utilizó para ilustrar la actividad de validación llamada Valoración.

Todos los Casos de Prueba diseñados junto a los Resultados Reales obtenidos al ejecutarlos para estas pruebas dinámicas se encuentran en el Apéndice F, aunque algunos de ellos también se ilustran en la Tabla 4-12. Tenga en cuenta que, dado que CQ3 (ver Sección 4.3.3.1) es un poco genérica para ser verificada de forma práctica, se dividió en 2 sub-CQs, a saber: CQ3.1. ¿Cuáles son los Resultados Reales producidos por una actividad de Realizar la Prueba?; y CQ3.2. ¿Cuáles son los Incidentes producidos por una actividad de Realizar la Prueba? Esto no fue necesario realizarlo para todas las CQs, como por ejemplo notar que la CQ25 no fue subdividida.

Además, se consideraron las siguientes precondiciones para los Casos de Prueba de la Tabla 4-12:

Precondición del TC#1: Que existen individuos implementados de Resultados Reales y actividades de Realizar la Prueba que los hayan producido.

Precondición de TC#2: Que existen individuos implementados de Incidentes y actividades de Realizar la Prueba que los hayan producido.

Precondición de TC#10: Que existen individuos implementados de Proyecto de Prueba, Meta de Prueba y Estrategia de Prueba, y están relacionados entre sí.

A su vez, como entradas para estos Casos de Prueba, se utilizaron las consultas implementadas en SPARQL que se muestran en la Figura 4-8. Las postcondiciones de

todos los Casos de Prueba es la misma, a saber: “el Resultado Real obtenido al ejecutar el Caso de Prueba y el resultado esperado deben coincidir”.

Tabla 4-12: Ejemplos de Casos de Prueba (TC) utilizados en la actividad de Pruebas Dinámicas Funcionales para TestTDO.

TC#	CQ	Datos de Prueba	Resultado Esperado
TC#1	CQ3.1	Datos tomados de las instancias del proyecto de prueba académico de una aplicación de figuras geométricas (ver Apéndice E).	Tres instancias de Resultado Real (AR) con los siguientes datos (nombre; valor): {(“AR#1; “Equilateral”); (“AR#2; “Scalene”); (“AR#3”; “Isosceles”)}
TC#2	CQ3.2		Una instancia de Incidente (nombre; descripción): {(“I#1”; “el Resultado Real no coincide con el resultado esperado, que es Invalid triangle”)}
TC#10	CQ25		Una instancia de Proyecto de Prueba (nombre), una de Estrategia de Prueba (nombre), una de Meta de Prueba (etiqueta; propósito), todos ellos relacionados: {(“TP#1”); (“Estrategia de Prueba Dinámica”); (“TG#1”, “verificar”)}

Una vez diseñados todos los Casos de Prueba se ejecutaron manualmente en el entorno Protégé (ver Resultados Reales en el Apéndice F). Es importante mencionar que todos los Casos de Prueba pasaron y, por lo tanto, no se produjo ningún Incidente en esta actividad de verificación dinámica.

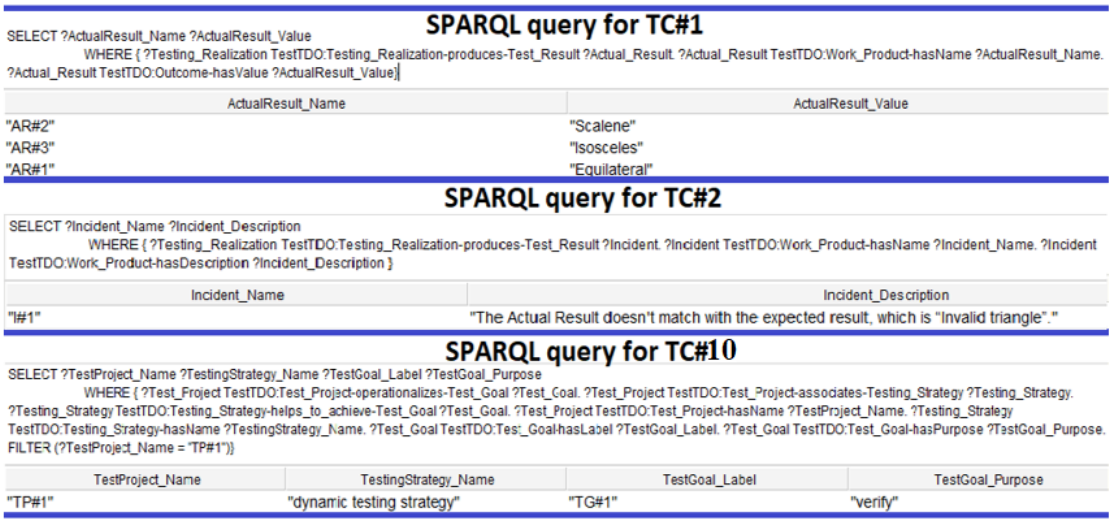


Figura 4-8: Consultas implementadas en SPARQL, las cuales son las entradas para los Casos de Prueba diseñados que se muestran en la Tabla 4-12.

Valoración

En la actividad de Valoración se validó si TestTDO era capaz de representar situaciones concretas del mundo. Para ello, se instanciaron algunos de sus términos, propiedades y relaciones en un proyecto de prueba académico de una aplicación de figuras geométricas. Este proyecto de prueba está basado en un ejemplo tomado del libro *The Art of Software Testing* de Myers *et al.* (Myers et al., 2012). En este ejemplo tenemos una aplicación que recibe tres enteros como entrada, donde cada valor representa el lado de un triángulo y, teniendo en cuenta estos valores, el programa devuelve como resultado el tipo de triángulo, es decir, isósceles, escaleno o equilátero.

En resumen, se instanciaron términos y propiedades de Entidad Comprobable, Proyecto de Prueba, Estrategia de Prueba y Metas de Prueba. También se consideraron Requisitos de Prueba asociados a Requisitos Funcionales y Bases de Prueba. Teniendo en cuenta términos con semántica extendida de ProcessCO, se instanciaron conceptos de Agentes de Prueba, Rol de Prueba, Actividades de Prueba, Casos de Prueba y Resultados de Prueba. Además, para la actividad de Diseñar la Prueba, se utilizaron dos Métodos de Diseño de Prueba para producir los Casos de Prueba, a saber: un Método basado en la Estructura (caja blanca) llamado pruebas de sentencias, y un Método basado en la Especificación (caja negra) llamado particiones de equivalencia. Todos los términos, propiedades y relaciones instanciados se encuentran en el Apéndice E. Como resultado de esta validación se concluyó que TestTDO es capaz de dar soporte semántico a Proyectos de Prueba que involucren Metas, Requisitos, y la ejecución de Actividades donde intervengan Agentes, Roles, Métodos, y se obtengan Productos de Trabajo.

Además, se realizaron entrevistas informales con tres expertos en el dominio de las pruebas de software, los cuales son externos al grupo de investigación GIDIS_Web. Ellos examinaron TestTDO y concuerdan que tiene utilidad. A su vez consideran, desde su punto de vista experto, que la terminología de la ontología es adecuada y completa.

Medición y Evaluación

Para evaluar TestTDO se utilizó como árbol de NFRs el mismo que se usó en la RSL para evaluar las 12 ontologías de pruebas de software seleccionadas (ver Tabla 3-10). Recordar que la raíz de este árbol es la característica llamada Calidad Ontológica (1). En cuanto a sus subcaracterísticas, se tiene la Calidad Estructural Ontológica (1.1), Calidad de Cobertura Terminológica específica del Dominio (1.2) y Adherencia a otros Vocabularios (1.3). Además, se utilizaron las mismas métricas e indicadores para realizar las actividades de medición y evaluación, así como también el mismo modelo de agregación, a saber, el modelo de agregación aditivo lineal. Notar que todas estas actividades y métodos forman parte de la estrategia GOCAME (Becker et al., 2015; Olsina & Becker, 2017). Se decidió utilizar el mismo árbol de NFRs y las mismas métricas e indicadores ya que el objetivo era no solo evaluar la ontología TestTDO, sino que también comparar la calidad ontológica entre ella y las 12 ontologías seleccionadas.

A continuación, en la Tabla 4-13, se muestran los resultados de la evaluación de TestTDO (en su última versión, i.e., v1.3) y se incluyen las 2 ontologías mejor clasificadas de la RSL realizada. Recuerde que se utiliza la metáfora del semáforo de tres colores para identificar el nivel de aceptabilidad alcanzado por cada atributo o característica, como se explicó en el Capítulo 3.

En cuanto a las métricas e indicadores utilizados, se diseñaron algunos más complejos y otros más sencillos. Por ejemplo, el procedimiento de la métrica utilizada para medir el atributo “Disponibilidad de relaciones no taxonómicas balanceadas” (1.1.4.1) contiene una fórmula que se utiliza para obtener el porcentaje de relaciones taxonómicas (es decir, relaciones “tipo-de/es-un” y “todo-parte/parte-de”) con respecto a las relaciones totales (recordar Figura 3-10). Además, se diseñó un indicador elemental para interpretar y evaluar este atributo (ver Figura 3-12). Estas métricas e indicadores fueron los más complejos que se utilizaron. Con respecto a los otros atributos relacionados con la subcaracterística Calidad Estructural Ontológica (1.1), las fórmulas de sus métricas y cómo interpretar sus resultados usando indicadores elementales se explicó en la Sección 3.3.3.3.

Tabla 4-13: Resultados de la evaluación de TestTDO v1.3 y las 2 ontologías mejor clasificadas de la RSL realizada. El círculo verde indica el nivel de aceptabilidad "satisfactorio" (●); el rombo naranja indica el nivel "marginal" (◆) y el cuadrado rojo "insatisfactorio" (■). Los valores de los indicadores están en [%].

Characteristics (subcharacteristics) / *Attributes*	**ROoST** (Ferreira de Souza et al., 2017)	**Asman *et al.*** (Asman & Srikanth, 2015)	**TestTDO v1.3**
1. Ontological Quality	79.54◆	66.71◆	98.11●
1.1 Ontological Structural Quality	79.08◆	61.92◆	96.22●
1.1.1 Defined Terms Availability	82.20◆	100●	100●
1.1.2 Defined Properties Availability	0■	0■	100●
1.1.3 Specified Axioms Availability	100●	0■	100●
1.1.4 Balanced Relationships Availability	87.32●	73.07◆	87.42●
1.1.4.1 Balanced Non-Taxonomic Relationships Availability	95.65●	66.34◆	84.27◆
1.1.4.2 Defined Non-Taxonomic Relationships Availability	54■	100●	100●
1.2 Domain-specific Terminological Coverage Quality	50■	100●	100●
1.2.1 Static Testing Terms Availability	0■	100●	100●
1.2.2 Dynamic Testing Terms Availability	100●	100●	100●
1.2.3 Functional Testing Terms Availability	100●	100●	100●
1.2.4 Non-Functional Testing Terms Availability	0■	100●	100●
1.3 Compliance to other Vocabularies	100●	52.50■	100●
1.3.1 Terminological use of Int'l Standard Glossaries	100●	85●	100●
1.3.2 Terminological Compliance to other Domain/Core Ontologies	100●	100●	100●
1.3.3 Terminological Compliance to Foundational Ontologies	100●	0■	100●

Por otro lado, para los atributos "Uso Terminológico de Glosarios de Estándares Internacionales" (1.3.1) y "Adherencia Terminológica con otras Ontologías de Dominio/Central" (1.3.2) se utilizaron métricas e indicadores más simples. Para poder evaluar estos atributos se investigó la documentación asociada de una ontología para encontrar alguna evidencia con respecto a si los autores habían considerado usar otras ontologías de dominio/centrales y/o glosarios de estándares internacionales para construirla. Notar que para construir TestTDO se utilizaron las ontologías centrales pertenecientes a FCD-OntoArch y los glosarios documentados en ISO 29119-1 e ISTQB. Entonces, simplemente las métricas que se usan para medir estos atributos cuentan el número de ontologías de dominio/centrales (y/o glosarios de estándares internacionales) consideradas y el indicador interpreta este valor de la siguiente manera: si no se considera ninguna ontología (o glosario), entonces el valor del indicador es 0% (■); si se considera 1 entonces el valor es 85% (●); y si se consideran 2 o más, el valor es 100% (●). Tenga en cuenta que no se realizó ningún otro análisis con respecto a estas métricas.

Asimismo, la métrica y el indicador utilizados para el atributo "Adherencia Terminológica con Ontologías Fundacionales" (1.3.3) también son simples. La métrica cuantifica únicamente el número de ontologías fundacionales en las que se basa la ontología a medir, sin analizar la calidad de la ontología fundacional utilizada. El indicador interpreta la medida de la siguiente manera: si no se considera una ontología fundacional, el valor del indicador es 0% (■); y si se consideran 1 o más, el valor es 100% (●). Recordar que TestTDO adhiere su terminología a la ontología fundacional ThingFO.

Por último, para los atributos asociados a la subcaracterística Calidad de Cobertura Terminológica específica del Dominio (1.2) también se utilizaron métricas e indicadores simples. Por ejemplo, para medir el atributo "Disponibilidad de Términos de Pruebas Estáticas" (1.2.1), se analizaban los conceptos de la ontología para decidir, de forma subjetiva, si contaba con al menos algún término de Pruebas Estáticas. En caso positivo,

la métrica e indicador daban como resultado 100% (●), sino 0% (■). Esta misma idea se aplicó para medir y evaluar los otros 3 atributos restantes (1.2.2, 1.2.3 y 1.2.4).

En resumen, observando los resultados de la Tabla 4-13 se puede afirmar que TestTDO supera en Calidad Ontológica (1) al resto de las 12 ontologías seleccionadas y es la única que alcanza un nivel satisfactorio (●). Lo mismo sucede para la subcaracterística Calidad Estructural Ontológica (1.1). No obstante, ROoST alcanza un mejor nivel de aceptabilidad para el atributo "Disponibilidad de relaciones no taxonómicas balanceadas" (1.1.4.1), lo cual es un punto a mejorar de TestTDO para alcanzar al menos un nivel de aceptabilidad satisfactorio.

Discusión sobre enfoques de VyV

Observando los estudios primarios que documentan las 12 ontologías seleccionadas durante la RSL realizada, solo tres de ellos mencionan explícitamente el uso de algún enfoque de VyV, a saber: ROoST (Ferreira de Souza et al., 2017), Asman *et al*. (Asman & Srikanth, 2015), y Vasanthapriyan *et al*. (Vasanthapriyan, Tian, Zhao, et al., 2017).

Por un lado, se utilizó un Método de Diseño de Pruebas basado en la Especificación (caja negra) para llevar a cabo Pruebas Estáticas que verifiquen el alcance de TestTDO. Los autores de ROoST también utilizaron este enfoque de verificación en (Ferreira de Souza et al., 2017), pero lo llamaron "valoración por enfoque humano" ya que ellos consideran que *testing* aplica solo para actividades dinámicas.

Teniendo en cuenta enfoques de VyV para Pruebas Dinámicas Funcionales, se diseñaron Casos de Prueba usando un Método de Diseño de Prueba de caja negra para poder verificar dinámicamente el alcance (CQs) de TestTDO. (Ferreira de Souza et al., 2017) también utilizaron este enfoque de verificación y lo llamaron "enfoque dirigido por preguntas de competencias para pruebas de ontología". Además, notar que previamente se introdujo un enfoque de pruebas dinámicas unitarias para ontologías en (Vrandečić & Gängemi, 2006).

Con respecto a la actividad llamada Valoración (ver Figura 4-6), se validó si TestTDO era capaz de representar situaciones concretas del mundo instanciando sus términos, propiedades y relaciones con un proyecto de prueba. En (Ferreira de Souza et al., 2017) los autores también aplicaron este enfoque para validar ROoST. Además, las ontologías presentadas por Asman *et al.* (Asman & Srikanth, 2015) y Vasanthapriyan *et al*. (Vasanthapriyan, Tian, Zhao, et al., 2017) fueron valoradas por expertos del dominio como se hizo también para TestTDO.

Por otro lado, para poder respaldar la actividad de Medición y Evaluación se desarrolló un árbol de NFRs (Tabla 3-10) y un conjunto de métricas e indicadores. Estos artefactos permitieron evaluar cuantitativamente característica de calidad ontológicas descritas en (d'Aquin & Gangemi, 2011) como la calidad estructural, la calidad de la cobertura terminológica y la adherencia a otros vocabularios. Un enfoque para evaluar estas características en ontologías fue presentado en (Ferreira de Souza et al., 2017), pero el mismo no era cuantitativo sino que más bien descriptivo y cualitativo.

En cuanto a estrategias de evaluación, los autores en (Brank et al., 2005) analizan diferentes enfoques de evaluación para ontologías como el enfoque de evaluación multicriterio llamado Ontometric (Lozano-Tello & Gómez-Pérez, 2004). Sin embargo, Duque-Ramos *et al*. (Duque-Ramos et al., 2011) consideró que Ontometric tiene una usabilidad limitada debido a su complejidad. Además, ellos proponen también una estrategia de evaluación. Analizando las métricas e indicadores propuestas por

Ontometric y Duque-Ramos *et al*. se notó que están débilmente especificadas y, por lo tanto, esto conlleva el riesgo que los resultados obtenidos haciendo uso de ellas sean menos reproducibles.

4.4 Conclusiones

Esta sección tiene como objetivo ilustrar las principales conclusiones de este capítulo. Con respecto a los hallazgos relacionadas con DSR, se puede confirmar que la comunidad de las Ciencias Informáticas adoptó este enfoque para el diseño y construcción de artefactos. Sin embargo, un análisis de la literatura permitió determinar que un proceso de DSR que establezca el flujo de actividades y tareas a realizar con sus correspondientes entradas y salidas aún no estaba formalmente especificado. Por lo tanto, con el objetivo de contribuir a este aspecto, en este capítulo se presentó una especificación de modelo de proceso para DSR considerando las perspectivas de modelado funcional y de comportamiento. Las especificaciones se realizaron utilizando el lenguaje SPEM. El modelo de proceso de DSR propuesto permite llevar a cabo de manera sistemática las actividades y tareas involucradas en este enfoque de investigación. Tener un proceso modelado ayuda a entenderlo y comunicarlo a la comunidad interesada, haciendo su aplicación más consistente y repetible en la ejecución de las actividades y tareas incluidas.

Es importante aclarar que se muestra un flujo recomendado para el proceso de DSR ya que se tiene consciencia de que en una instanciación particular de un proceso puede haber algunos puntos de variación, como la paralelización de algunas tareas. Además, el ciclo de vida de un determinado proyecto de DSR debe organizar actividades y tareas considerando no solo el proceso de DSR prescrito, sino también la asignación adecuada de recursos, como métodos y herramientas, entre otros, para lograr la meta propuesta. Por lo tanto, existen actividades adicionales a las especificadas en la Figura 4-1 que la planificación del proyecto también debe tener en cuenta (por ejemplo, controlar los cambios y versiones de los artefactos producidos).

El proceso de DSR propuesto también proporciona una buena base para comprender los detalles y poder discutir alternativas o personalizaciones del mismo. De hecho, el rigor proporcionado por el modelado de procesos, donde se combinan varias perspectivas pero también se pueden separar de forma independiente, proporciona una mayor riqueza de expresividad.

Por otra parte, después de analizar tanto los resultados obtenidos al realizar la RSL sobre ontologías de pruebas de software (Capítulo 3) como el estado del arte de estándares relacionados también con *testing*, se decidió desarrollar una nueva ontología de pruebas de software que se ajuste a los requisitos planteados en la Sección 4.3.3.1. Por lo tanto, usando como guía el proceso de DSR propuesto, se creó esta ontología y recibe el nombre de TestTDO. La misma se ubica en el nivel de dominio superior en el contexto de la arquitectura de cinco capas llamada FCD-OntoArch. Tener en cuenta que en este capítulo se detallan todas las actividades del proceso de DSR para la construcción de TestTDO a excepción de la última (A4 Comunicar la Investigación) ya que la misma implica la documentación/comunicación del artefacto generado y sus resultados relacionados. Notar que, para documentar/comunicar esta investigación como parte de la actividad A4 de DSR, se publicaron los artículos científicos que se muestran en la Tabla 6-2. A su vez, esta tesis de doctorado es otro documento adicional que cumple dicho propósito.

Para verificar y validar ampliamente TestTDO se llevaron a cabo varias actividades de VyV. Por un lado, para verificar TestTDO se realizaron dos tipos de actividades de

verificación estática, una para verificar el alcance de TestTDO (es decir, las CQs) y otra para inspeccionar la consistencia semántica entre las relaciones de TestTDO y las relaciones que las enriquecen de las ontologías de niveles superiores pertenecientes a FCD-OntoArch. También se verificó dinámicamente el alcance de TestTDO mediante la ejecución de un conjunto de casos de prueba. Por otro lado, se validó TestTDO instanciando sus términos en un proyecto de prueba académico y además fue examinada por tres expertos en el dominio de las pruebas de software. Finalmente, se realizaron actividades de medición y evaluación siguiendo la estrategia GOCAME para evaluar la calidad ontológica de TestTDO.

La versión actual de TestTDO (v1.3), la cual fue mejorada gracias a las actividades de VyV anteriormente mencionadas, tiene definido un total de 44 términos, 51 propiedades, 43 relaciones no taxonómicas y 17 axiomas especificados. De estos 44 términos, 32 están enriquecidos por ProcessCO usando estereotipos (ver Figura 4-2), lo cual significa que aproximadamente el 70 % de los términos de TestTDO se enriquecen con esta ontología de nivel central. Esto se debe principalmente a que el objetivo de TestTDO es nutrir terminológicamente las especificaciones de métodos y procesos de una familia de estrategias de prueba. Como se mencionó en el Capítulo 2, una especificación de proceso bien documentada debe ilustrar qué actividades deben llevarse a cabo, qué roles están involucrados y qué productos de trabajo consumen/producen las actividades (Becker et al., 2022; Curtis et al., 1992). Asimismo, una especificación de método bien documentada debe contener el procedimiento a seguir y las reglas asociadas (Becker et al., 2022). Por lo tanto, para estar alineado con este objetivo era necesario considerar la mayoría de los conceptos pertenecientes a ProcessCO. Sin embargo, no son los únicos importantes, sino que son necesarios otros términos relacionados con situaciones, proyectos, metas y entidades ya que son conceptos centrales y transversales para muchos dominios.

CAPÍTULO 5: *SAST STRATEGY* – UNA ESTRATEGIA INTEGRADA DE PRUEBAS DE SOFTWARE CONSCIENTE DE LA SITUACIÓN Y BASADAS EN ESCENARIOS

5.1 Introducción

En el capítulo anterior se describió detalladamente, entre otras cosas, la ontología de pruebas de software a nivel de dominio superior llamada TestTDO. El objetivo principal de esta ontología es enriquecer semánticamente una familia de estrategias de pruebas que ayuden a alcanzar diferentes propósitos de *testing*. Además, como se mencionó en el Capítulo 2, una estrategia bien especificada debería integrar tres pilares o capacidades, a saber: una especificación de proceso; una especificación de método; y una base conceptual que soporte semánticamente estas especificaciones (ver Figura 2-5). Por lo tanto, una vez construida la ontología TestTDO, se desarrolló una estrategia de *testing* utilizando los conceptos de TestTDO como base para enriquecer las especificaciones de su proceso y método. Esta nueva estrategia recibe el nombre de Estrategia de Prueba basada en Escenarios y consciente de la Situación (*Situation-aware Scenario-based Testing Strategy: SaST Strategy*). Recordar que en la Sección 1.1 se describió la motivación que llevó al desarrollo de esta estrategia. Además, es importante mencionar que la estrategia SaST fue diseñada para usarse solamente en actividades de Pruebas Dinámicas.

Uno de los pilares de SaST es su proceso, el cual recibe el nombre de SaSTPro (*Situation-aware Scenario-based Testing Process*). El mismo está especificado usando SPEM y considerando las vistas funcional y de comportamiento. Este proceso está asociado al objetivo específico 3 y contribución 5 (OE_3 y C_5, respectivamente, en Tabla 1-1), y se documenta en la Sección 5.2. Por otra parte, el método de la estrategia SaST se denomina SaSTMe (*Situation-aware Scenario-based Testing Method*) y se describe en la Sección 5.3. Además, este método se encuentra relacionado al objetivo específico 4 y contribución 6 (OE_4 y C_6, respectivamente, en Tabla 1-1).

Por otra parte, la Sección 5.4 contiene trabajos relacionados sobre estrategias de *testing* basadas en escenarios y/o que consideren entidades de contexto. Además, la estrategia SaST fue aplicada por dos estudiantes de grado de la carrera Ingeniería en Sistemas de la UNLPam en el marco de sus proyectos finales (o tesina de grado) recientemente defendidos. Estas aplicaciones fueron convenientes para validar la utilidad de esta estrategia dado que se utilizó en ambos casos para diseñar pruebas para un proyecto real. En la Sección 5.5 se ilustrarán estas aplicaciones y los resultados obtenidos. Notar que esto último forma parte del objetivo específico 5 y las contribuciones 7 y 8 (OE_5, C_7 y C_8, respectivamente, en Tabla 1-1) de esta tesis doctoral.

5.2 SaST Process – Especificación de Proceso para la Estrategia SaST

Considerando las perspectivas funcional y de comportamiento, la Figura 5-1 muestra la especificación de proceso de SaSTPro utilizando SPEM, el cual es uno de los pilares de la estrategia integrada SaST. A continuación, se describen las actividades de SaSTPro y los artefactos consumidos/producidos por ella. Además, se resaltarán todos los conceptos (términos, propiedades y relaciones) de este proceso que estén enriquecidos

con la ontología TestTDO. Para los términos principales se utilizarán mayúsculas, para las propiedades cursiva, y las relaciones no taxonómicas estarán subrayadas. Recordar que en el capítulo anterior se definieron todos los conceptos de TestTDO.

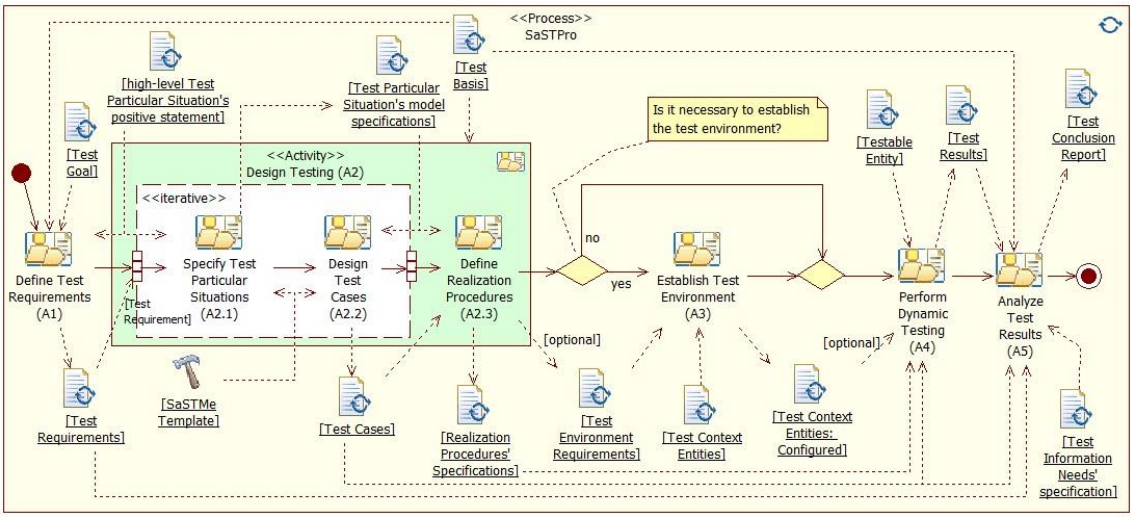


Figura 5-1: Especificación de las perspectivas funcional y de comportamiento para SaSTPro.

De acuerdo al proceso que se muestra en la Figura 5-1, la primera actividad a realizar es "Definir Requisitos de Prueba (A1)". Esta actividad requiere como entrada los siguientes documentos: la Meta de Prueba y la *declaración positiva* de la Situación Particular de Prueba, ambos documentos a nivel de Proyecto de Prueba, y también utiliza la Base de Prueba. El objetivo de A1 es producir los Requisitos de Prueba, los cuales establecen qué se debe verificar/validar de un Objeto de Prueba o Entidad Comprobable. Observando la Figura 4-3, es importante especificar en un Requisito de Prueba su *etiqueta*, *declaración*, *fase de la entidad comprobable*, *nivel de prueba* y sus *criterios de finalización*.

La siguiente actividad a realizar es "Diseñar la Prueba (A2)", la cual tiene tres subactividades. La primera es "Especificar Situaciones Particulares de Prueba (A2.1)" y produce *especificaciones de modelo* de Situaciones Particulares de Prueba, las cuales de ahora en adelante en este capítulo se nombrarán simplemente como *modelos de situación*. Ejemplos de *modelos de situación* pueden ser escenarios de casos de uso o diagramas de comunicación. Para producirlos se consume la Base de Prueba como entrada y, a su vez, estos *modelos de situación* deben estar alineados con los Requisitos de Prueba y la *declaración positiva* de la Situación Particular de Prueba (a alto nivel, i.e., a nivel de Proyecto de Prueba). En algunos casos, la Base de Prueba en sí misma puede ser un *modelo de situación*, por ejemplo, cuando este artefacto es un diagrama de casos de uso. Además, se pueden realizar pequeñas variaciones a estos modelos para producir nuevos modelos de situaciones (Cockburn, 2001).

De acuerdo a la Figura 4-3, una Situación Particular de Prueba asocia (trata con) Entidades Comprobables (test targets) y Entidades de Contexto de Prueba (test environment, si corresponden), por lo tanto, un *modelo de situación* también considera estas entidades de prueba en su especificación. Notar que en la plantilla SaSTMe (Tabla 5-1) se captura esta información.

Una vez producidos los *modelos de situación*, el siguiente paso es producir Casos de Prueba en la actividad "Diseñar Casos de Prueba (A2.2)". Para ello, A2.2 consume los *modelos de situación* y la Base de Prueba. Los Casos de Prueba se diseñan teniendo en

cuenta principalmente los *modelos de situación*, pero la Base de Prueba también podría ser útil para definir los *resultados esperados* de los casos. Además de un *resultado esperado*, un Caso de Prueba también tiene *entradas*, *precondiciones*, y *postcondiciones* (observar la Figura 4-4). En resumen, en A2.2 se especificarán estas propiedades para cada Caso de Prueba analizando los insumos de la actividad (i.e., *modelos de situación* y Base de Prueba) y se registrarán haciendo uso de la plantilla SaSTMe (Tabla 5-1). En este punto es importante señalar que las actividades A2.1 y A2.2 se realizan de forma iterativa para cada Requisito de Prueba.

Luego, se debe realizar "Definir Procedimientos de Realización (A2.3)". Esta actividad utiliza toda la información de la plantilla SaSTMe ya rellenada como entrada (es decir, los *modelos de situación* y Casos de Prueba) para producir las *especificaciones* de los Procedimientos de Realización. Por ejemplo, cuando se llevan a cabo Pruebas Dinámicas que involucran la ejecución de Casos de Prueba, la *especificación* de un Procedimiento de Realización podría ser una secuencia de Casos de Prueba en cierto orden de ejecución, y cualquier acción asociada que pueda ser requerida para establecer sus *precondiciones* y *postcondiciones*.

Además, cuando es necesario establecer un entorno de prueba, los requisitos del entorno de prueba se establecen en A2.3 considerando también la plantilla SaSTMe ya que esta contiene información de Entidades de Contexto de Prueba. Por ejemplo, los requisitos del entorno podrían estar asociados con hardware, simuladores, herramientas de software y otros elementos de soporte que sean necesarios para la actividad de "Realizar la Prueba Dinámica (A4)". Luego, si se ejecuta la Actividad de Prueba "Establecer el Entorno de Prueba (A3)" (notar que es opcional), se insumen los requisitos del entorno y se requieren como entrada las Entidades de Contexto de Prueba correspondientes. Como salida de A3, se configura el entorno de prueba y se deja listo para ser utilizado en A4.

Una vez finalizada la actividad A3 (o si no es necesario establecer el entorno de prueba), se lleva a cabo A4, la cual requiere como entrada la Entidad Comprobable y, si corresponde, las Entidades de Contexto de Prueba configuradas. Además, se utilizan las *especificaciones* de los Procedimientos de Realización. También pueden requerirse las especificaciones de los Casos de Prueba, principalmente cuando se realizan pruebas manuales. El objetivo de A4 es ejecutar los Procedimientos de Realización (notar que esto implica ejecutar los Casos de Prueba) usando la Entidad Comprobable correspondiente en su entorno de prueba previamente establecido y configurado (si existe). Como resultado, A4 produce Resultados de Prueba, i.e., Resultados Reales y/o Incidentes.

Finalmente, se realiza la actividad "Analizar los Resultados de Prueba (A5)". En A5, los Resultados de Prueba se analizan para determinar si se cumplen los Requisitos de Prueba y si se alcanzan las Necesidades de Información de Prueba. Estas últimas proporcionan información útil para definir en qué grado se alcanzaron las Metas de Prueba. Todo el resultado de este análisis se documenta en el Informe de Conclusión de Prueba.

Como comentario final de esta sección es importante señalar que SaSTPro se muestra puramente secuencial para mantener su legibilidad, excepto por las subactividades A2.1 y A2.2 que iteran por cada Requisito de Prueba. En la práctica, otras actividades también podrían iterar, por ejemplo, si algún Requisito de Prueba no se cumple debido a que es necesario contemplar más *modelos de situación* o diseñar más Casos de Prueba, entonces las actividades A2-A5 podrían volverse a ejecutar.

5.3 SaST Method – Especificación de Método para la Estrategia SaST

En esta sección se describe el método utilizado por la estrategia SaST, el cual representa uno de sus pilares. Al igual que en la sección anterior, se resaltarán los conceptos de este método que estén enriquecidos con la ontología TestTDO, utilizando mayúsculas para los términos principales, cursiva para las propiedades, y subrayado para las relaciones.

El método de la estrategia SaST recibe el nombre de SaSTMe y considera la especificación del *modelo de situación*. Este método da soporte al diseño y documentación de Casos de Prueba, capturando información relevante sobre la Situación Particular de Prueba como cuáles son las Entidades Comprobables (y Elementos de Prueba si existen) y las Entidades de Contexto de Prueba con las que trata. Es importante señalar que SaSTMe es un Método Basado en la Especificación, es decir, no utiliza la estructura interna de la Entidad Comprobable para diseñar los Casos de Prueba.

La Tabla 5-1 muestra la plantilla para SaSTMe. En primer lugar, esta plantilla pretende capturar información sobre una Situación Particular de Prueba, por lo que los diseñadores de las pruebas deben documentar su *declaración positiva* y su *modelo de situación* (o *especificación de modelo*). Ambos podrían especificarse en lenguaje natural, pero el *modelo de situación* sería más valioso si también se especificara formalmente en un diagrama estandarizado, como diagramas de secuencia UML, diagramas de comunicación UML, entre otros. Es importante señalar que la Situación Particular de Prueba podría ser diseñada desde cero en la fase de *testing* o podría ser producida en alguna fase previa del ciclo de desarrollo de software. Además, otro aspecto importante a destacar de una Situación Particular de Prueba son sus *condiciones*, las cuales representan restricciones (ya sean previas o posteriores) que deben cumplirse en la situación. Estas *condiciones* serán útiles luego para diseñar las condiciones (*precondiciones* y *postcondiciones*) de los Casos de Prueba. Es importante mencionar que no es obligatorio rellenar este campo.

Una vez que se especificó la Situación Particular de Prueba, el siguiente paso es analizarla para identificar las Entidades Comprobables (y los Elementos de Prueba, si corresponde) y las Entidades de Contexto de Prueba (si las hay). También se debe documentar cómo las Entidades de Contexto de Prueba influencian a las Entidades Comprobables ya que esta información podría ser útil para las actividades de SaSTPro en donde se diseñan e implementan los Casos de Prueba y el entorno de prueba.

Tabla 5-1: Especificación de la plantilla para el método SaSTMe.

Test Particular Situation: -*positive statement*: -*model specification*: -*conditions*: (not mandatory)	
Testable Entities (and **Test Items**, if applicable)**:** *indicate what are the test targets*	
Test Context Entities: (not mandatory) If exists Test Context Entities: -*Influences*: describe how the test context entities influence the test targets.	
Test Requirement:	
Test Basis:	
Test Cases: -*input*: -*preconditions*:	 -*expected result*: -*postconditions*:

Además, la plantilla captura el Requisito y las Bases de Prueba relacionados con la Situación Particular de Prueba. A su vez, las Bases de Prueba que se documentan aquí deberían ser útiles para el diseño de los Casos de Prueba (por ejemplo, especificaciones de requisitos funcionales o no funcionales). Notar que estos artefactos son convenientes para producir los *resultados esperados* de los Casos de Prueba.

El último paso consiste en utilizar toda la información ya incluida en la plantilla para producir Casos de Prueba. Es importante señalar que las Situaciones Particulares de Prueba y las Entidades de Contexto de Prueba asociadas pueden influir en las *entradas*, las *condiciones* o ambas de los Casos de Prueba (de Souza Doreste & Travassos, 2020). El objetivo es que los Casos de Prueba puedan reproducir o cubrir en el mayor grado posible todos los pasos especificados en las Situaciones Particulares de Prueba.

5.4 Trabajos Relacionados

Un trabajo relacionado es (de Souza Doreste & Travassos, 2020), en el cual los autores proponen CATS#, una evolución de la técnica de diseño CATS (*Context-Aware Test Suite*). Esta técnica da soporte a la especificación de casos de prueba conscientes del contexto. Mediante el uso de CATS#, los casos de prueba pueden capturar variables de contexto y también condiciones variables. Sin embargo, su enfoque no captura explícitamente cuáles son las entidades de contexto como lo hace SaSTMe, sino que solamente considera algunas propiedades de las entidades de contexto. Es importante mencionar que CATS# fue diseñada para probar CASS (*Context-Aware Software Systems*) y la estrategia SaST es más general, i.e., no solo podría usarse para probar estos tipos particulares de sistemas, sino que para otros también. Además, CATS# no está soportada semánticamente por una base conceptual robusta como una ontología.

En (Harmse et al., 2014), los autores presentan un enfoque de prueba basado en escenarios utilizado para validar formalmente diagramas de clases UML. Ellos proponen un proceso a seguir especificado como un conjunto de pasos y en lenguaje natural. Además, definen tres tipos de escenarios: consistente, inconsistente y de clasificación. En resumen, la idea propuesta es traducir el diagrama de clases UML a OWL y verificar su consistencia. Luego, si el modelo es consistente, el siguiente paso es producir y verificar escenarios que representen los requisitos del negocio. En esta investigación los autores utilizan ontologías, pero solo con fines de verificación y no para el enriquecimiento semántico de los conceptos de prueba como es el caso de SaST. Además, esta metodología solo se aplica en actividades de Pruebas Estáticas.

Otros trabajos como (Mirza & Khan, 2018; Wang et al., 2007; Yu et al., 2016) discuten enfoques para probar CASS. Todos ellos concuerdan con la siguiente definición de contexto: "*Any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves*" (Abowd et al., 1999). Esta definición es similar a la brindada por SituationCO ya que, desde el punto de vista semántico de esta ontología, Entidades Objetivo rodeadas e influenciadas por Entidades de Contexto en una Situación Particular tiene la semántica de contexto. Además, este modelo semántico captura explícitamente tipos de entidades, i.e., entidades objetivo y de contexto.

Según ISO 29119-2 (ISO, 2013b), el proceso de prueba para pruebas dinámicas implica cuatro actividades principales, a saber: diseñar e implementar las pruebas; configurar y mantener el entorno de prueba; ejecutar las pruebas; e informar incidentes

de pruebas. Por otro lado, el proceso presentado en ISTQB (ISTQB, 2018) también considera cuatro actividades principales, a saber: analizar la prueba; diseñar la prueba; implementar la prueba; y ejecutar la prueba. Notar que el proceso propuesto para la estrategia SaST (i.e., SaSTPro) que se muestra en la Figura 5-1 también considera estas actividades. Además, se agregó la subactividad "Especificar Situaciones Particulares de Prueba (A2.1)", que no está incluida en las dos propuestas anteriormente citadas.

5.5 Aplicaciones de la Estrategia SaST

Esta sección contiene dos aplicaciones diferentes de la estrategia SaST. Ambas fueron llevadas a cabo por estudiantes de grado de la carrera Ingeniería en Sistemas de la Facultad de Ingeniería de la UNLPam en el marco de sus Trabajos Finales (o tesina). Además, es importante destacar que la estrategia fue aplicada para probar aplicaciones de software en contextos laborales reales en ambos casos.

Por un lado, la Sección 5.5.1 ilustra la primera aplicación de la estrategia SaST (Eleicegui, 2022), la cual fue utilizada para diseñar casos de prueba a nivel de integración con el objetivo de probar un software de gestión de contratos y pagos. Por otro lado, la segunda aplicación de SaST (Fernandez Reale, 2023) se encuentra documentada en la Sección 5.5.2 y se utilizó para diseñar y automatizar casos de prueba a nivel de sistema para probar una aplicación web de una inmobiliaria. A continuación, en las próximas secciones, se describen estos dos casos aplicados. Notar que no se exponen aquí todos los detalles de ambos casos, sino que solamente una parte de ellos, los cuales son suficientes para ilustrar la aplicación de la estrategia. El lector puede encontrar toda la documentación de ambos casos aplicados en (Eleicegui, 2022; Fernandez Reale, 2023).

5.5.1 Caso Aplicado 1 de la Estrategia SaST

En la empresa Agripay S.A se desarrolló una plataforma de software que se utiliza para gestionar contratos inteligentes y pagos digitales. Además, la plataforma cuenta de distintos frentes, entre los cuales se encuentran una aplicación web, un conjunto de aplicaciones móviles para distintos tipos de dispositivos y una API *backend* que brinda *endpoints* para la interacción con la base de datos y para la comunicación con otros servicios externos que necesiten ser utilizados por las distintas aplicaciones. Dado que la aplicación web depende de los servicios que brinda la API *backend* como así también de otros servicios externos con los que se pueda estar comunicando, se utilizó la estrategia SaST (Eleicegui, 2022) para diseñar casos de prueba a nivel de integración que ayuden a verificar el comportamiento esperado de estas entidades operando en conjunto. SaST resultó útil para diseñar estos casos de prueba ya que considera el modelo de situación con las entidades comprobables y las entidades de contexto de prueba que las influencien.

Para este caso aplicado se llevaron a cabo solamente pruebas funcionales, es decir, pruebas para verificar requisitos funcionales. Además, solo se realizaron las actividades A1 y A2 de SaSTPro (recordar Figura 5-1). Es importante mencionar que las aplicaciones móviles anteriormente mencionadas no fueron consideradas para las pruebas. Solamente la aplicación web y la API *backend* son las entidades comprobables en este caso particular. A continuación, se describen cada una de ellas.

5.5.1.1 Entidades de Prueba

La API brinda servicios accesibles desde la aplicación web (*frontend*) mediante distintos *endpoints*. Notar que, un *endpoint* es el *path* o URL mediante la cual se hace la

petición al servicio. Estos servicios son consumidos por las aplicaciones *frontend* para consultar y/o modificar datos de una base de datos compartida. Como se mencionó anteriormente, la API es la entidad comprobable de alto nivel, pero a su vez tiene elementos de prueba asociados de más bajo nivel, los cuales son los distintos servicios disponibles. De ahora en adelante se utilizará AS ("*API Service*") para hacer mención a un servicio de la API. Por otra parte, la aplicación web cuenta con componentes web como elementos de prueba y estos serán nombrados de forma abreviada como WC ("*Web Component*").

Uno de los elementos de prueba objetivo (*test target*) asociado a la aplicación web que se seleccionó para las pruebas es el "WC listar-contratos", cuya funcionalidad consiste en mostrar por pantalla un listado de los contratos de una cierta empresa. Para poder cumplir su función depende del "AS listar-contratos", el cual genera la lista de contratos consultando una base de datos. Notar que el AS listar-contratos también es un elemento de prueba objetivo, pero es parte de la API *backend*. La interfaz del WC listar-contratos proporciona una botonera que permite realizar distintas acciones sobre un contrato, como puede ser ver el detalle del mismo, rechazarlo o firmarlo. También incluye un panel desplegable que permite utilizar filtros para obtener un conjunto de contratos en un cierto rango de fechas, contratos firmados o no firmados, entre otros. Por ejemplo, utilizando estos filtros le podemos indicar al WC que queremos obtener todos los contratos firmados que sean posteriores al 2018.

Otro elemento de prueba objetivo es el "WC firmar-contratos", el cual, como su nombre lo indica, permite firmar digitalmente un contrato. El mismo posee una interfaz de vista previa del documento del contrato (en pdf) para que el usuario pueda llevar a cabo su lectura y brinda un botón para iniciar el proceso de firma. Luego, para poder cumplir su función, consume el servicio "AS enviar-sms", el cual genera un segundo factor de autenticación al momento de realizar la firma mediante un servicio externo de envío de mensajes de texto. Además, también se necesita consumir el "AS firmar-contratos" para realizar la firma digital del contrato. Es importante mencionar que el AS firmar-contratos a su vez utiliza un servicio externo para generar un estampado *blockchain* que sirve para la trazabilidad del contrato.

Para una situación de prueba en la cual un usuario firma un contrato, el AS enviar-sms junto con el servicio externo de envío de mensajes de texto que utiliza, y el otro servicio externo que utiliza el AS firmar-contratos para generar la estampa *blockchain*, podrían considerarse todos estos servicios como entidades de contexto de prueba. Además, el AS firmar-contratos y el WC firmar-contratos serían los *test targets*.

5.5.1.2 Bases de Prueba

Esta sección tiene como objetivo describir las bases de prueba utilizadas para el diseño de las pruebas en este caso aplicado. Por un lado, como se muestra en la Tabla 5-2, se utilizó un esquema de clasificación que considera 5 niveles de impacto, los cuales tienen asociados un conjunto de respuestas de la API con ciertos códigos (por ejemplo, 200, 403) que se deben considerar o no para las pruebas según el nivel. Por ejemplo, el nivel de impacto 2 llamado "medio" considera solamente la respuesta de la API con código 200, es decir, se debería diseñar al menos una prueba para esta respuesta. En cambio, el nivel 5 ("crítico"), considera todas las respuestas de la API que se muestran en la tabla que tienen un código asociado e incluso también otros errores sin códigos que provienen de servicios externos, por lo tanto, se deben diseñar pruebas para considerar todos estos casos. Notar que, a medida que aumenta el nivel de impacto aumentan la cantidad de

pruebas a realizar. Además, este esquema se utilizó para clasificar los WCs según su grado de impacto sobre el funcionamiento final de la aplicación.

Tabla 5-2: Esquema de clasificación para los WCs que considera 5 niveles de impacto. Además, se muestran las respuestas de la API a considerar en las pruebas según el nivel de impacto. Tabla tomada de (Eleicegui, 2022).

Impacto					
#	1	2	3	4	5
Nombre	Bajo	Medio	Alto	Muy Alto	Crítico
Descripción	Componente prescindible: no compromete la experiencia de usuario.	Componente de soporte: no compromete la experiencia de usuario directamente, aunque puede comprometer el funcionamiento de otros componentes.	Componente de funcionalidad: compromete de forma directa la experiencia de usuario.	Componente de negocio: es una funcionalidad clave del producto.	Componente de aplicación: compromete directamente la utilización de la aplicación.
Respuestas de la API a considerar para las pruebas					
200	NO	SI	SI	SI	SI
403	NO	NO	NO	SI	SI
404	NO	NO	Solo si hay parámetros en la URL	Solo si hay parámetros en la URL	SI
405	NO	NO	NO	NO	SI
422	NO	NO	NO	NO	SI
Errores sin código provenientes de servicios externos	NO	NO	NO	SI	SI

La Tabla 5-3 contiene una descripción únicamente de las respuestas de la API que se consideraron para este caso aplicado. Además, se indica bajo qué condición se obtienen dichas respuestas. Notar que, para cada consulta realizada por parte de un WC a un AS, este último retorna una respuesta en formato JSON que tiene la siguiente estructura:

```
{
    code: number
    message: string
    errors: [{string: string}] (Solo presente en casos de error)
    data: JSON (Solo presente en casos 'success')
}
```

Tabla 5-3: Descripción y condiciones de algunas respuestas de los AS. Tabla tomada de (Eleicegui, 2022).

Código	**Mensaje**	**Error**	**Condición**
200	Success	Null	El AS procesó correctamente la consulta
403	Authorization Error: Permission not granted	<nombre de permiso>: 'missing'	El Usuario no posee el permiso necesario para realizar una determinada acción
404	URL not found	La URL no pudo ser procesada	El *endpoint* consultado no corresponde a un AS.
	URL param error	Param Error: <nombre del parámetro> <descripción>	Alguno de los parámetros que conforman el *endpoint* no pudo ser procesado

Por ejemplo, la respuesta con código 200 devuelve un mensaje "Success", el error es nulo, y esta respuesta se obtiene cuando el AS procesa de forma correcta la consulta por parte del WC. Por otro lado, las respuestas con códigos 403 y 404 indican casos de error. El código 403 indica un error por la falta de permisos para ejecutar una acción. En cambio, el código 404 significa que hubo un error en el procesamiento de una ruta por haber

indicado un *endpoint* inexistente o por tener algún error en los parámetros que puedan llegar a conformar el mismo.

Además, en la Tabla 5-4 se listan los WCs *targets* que se describieron en la Sección 5.5.1.1 junto a los ASs asociados, y además se muestra una breve descripción y su clasificación de nivel de impacto (recordar la Tabla 5-2). Por ejemplo, el WC listar-contratos que se relaciona con el AS listar-contratos tiene un nivel de impacto Alto (3), lo cual implica que se deben realizar pruebas para probar las respuestas con código 200 y, en caso de existir parámetros en la URL, el 404 también.

Tabla 5-4: Nivel de impacto de los WC *targets*. También se muestran los servicios de la API relacionados y una breve descripción del WC. Recordar que los niveles de impacto se definieron en la Tabla 5-2. Tabla tomada de (Eleicegui, 2022).

WC *target*	ASs relacionado	Descripción del WC	Impacto
WC listar-contratos	AS listar-contratos	Muestra una lista de los contratos de un usuario y presenta una botonera de acciones posibles a realizar sobre una instancia de contrato	Alto (3)
WC firmar-contratos	AS firmar-contratos AS enviar-sms	Realiza una vista previa del documento pdf del contrato a firmar y muestra un botón que da la opción al usuario de firmar el contrato	Muy Alto (4)

Por su parte, la Tabla 5-5 contiene información de los ASs relacionados con los WCs *targets*. Para cada uno de ellos se muestra su *endpoint* de acceso, una breve descripción del método HTTP necesario para comunicarse con él (POST o GET) y servicios externos asociados. Por ejemplo, el servicio AS listar-contratos tiene a "base-url/users/**user_id**/companies/**company_id**/contracts" como *endpoint*. Además, el método HTTP que utiliza para obtener el listado de los contratos vinculados a la empresa y al usuario indicados en los parámetros de ruta **company_id** y **user_id** es el GET, y para ello no se comunica con ningún servicio externo.

Tabla 5-5: ASs relacionados con los WCs *targets*. Se muestra sus *endpoints* marcando en negrita los parámetros en la URL, una descripción del método HTTP utilizado y los servicios externos asociados. Tabla tomada de (Eleicegui, 2022).

Nombre del AS	*endpoint* de acceso (parámetros en negrita)	Descripción del método HTTP	Servicios externos asociados
AS listar-contratos	base-url/users/**user_id**/companies/**company_id**/contracts	GET: Obtiene el listado de contratos vinculados a una empresa y un usuario en particular. Puede contener parámetros de filtrado de resultados tales como rango de fecha, empresa emisora del contrato, entre otros.	-
AS firmar-contratos	base-url/users/**user_id**/companies/**company_id**/contracts/**contract_id**/sign	POST: Marca como firmado un contrato digitalmente. Requiere como parámetro un código de verificación enviado por SMS.	Servicio de estampado *blockchain*
AS enviar-sms	base-url/users/**user_id**/send-sms	POST: Envía un código verificación SMS al número de teléfono del usuario indicado en la URL. Funcionalidad complementaria como segundo factor de autenticación.	Servicio de envío de SMS

Por último, la Figura 5-2 muestra los datos de prueba que se utilizaron para este caso aplicado. En la misma se pueden apreciar algunos datos de usuarios (tabla 'Users'); empresas ('Companies'); las relaciones entre los anteriores ('Companies x Users'); permisos ('Permissions'); los permisos asignados a los distintos usuarios ('Permissions x Users'); y datos de contratos ('Contracts').

Como último comentario de esta sección, recordar que todo lo documentado aquí forma parte de las bases de prueba, las cuales serán utilizadas para diseñar los casos de prueba. El diseño de las pruebas se ilustra en la sección siguiente.

Users				
user_id	***name***	***surname***	***email***	***phone-number***
1	Jhon	Doe	jhondoe@example.net	230251895
2	Anna	Flick	annaf@example.net	2302889512

Companies	
company_id	***business name***
1	Test S.A
2	Prueba S.R.L

Companies x Users	
company_id	***user_id***
1	1
2	2

Permissions	
permission_id	***name***
1	list-contracts
2	sign-contracts

Permissions x Users	
permission_id	***user_id***
1	1
1	2
2	1

Contracts				
contract_id	***company_id***	***contract_type***	***date***	***signed***
1	1	1	06/03/2021	0
2	1	3	04/04/2021	1
3	2	3	11/10/2021	0
4	2	2	20/02/2021	0
5	1	4	14/07/2021	1

Figura 5-2: Datos de prueba para el caso aplicado 1, tomados de (Eleicegui, 2022).

5.5.1.3 Ejecución de SaSTPro

Definir Requisitos de Prueba (A1)

Según SaSTPro (recordar Figura 5-1), la primera actividad a llevar a cabo es "Definir Requisitos de Prueba (A1)". Esta actividad requiere como entrada documentos a nivel de proyecto de prueba, a saber: la meta de prueba y la declaración positiva de la situación particular de prueba, y también las bases de prueba. El objetivo es producir los requisitos de prueba especificando su etiqueta, declaración, fase de la entidad comprobable, nivel de prueba y sus criterios de finalización (recordar Figura 4-3).

Por un lado, la meta de prueba propuesta en este caso aplicado declaraba lo siguiente: "verificar el correcto funcionamiento, de forma integrada, de los componentes de la aplicación web *frontend* (WCs) y los servicios de la API *backend* (ASs) que cada uno de ellos consume". Por otra parte, la declaración positiva de la situación particular de prueba

establecía que "los componentes Web App *frontend* de listado y firma de contrato junto con los servicios relacionados de la API *backend* deberían funcionar correctamente de forma integrada".

Observando las declaraciones anteriores, ambas establecen que se deben verificar los componentes WCs listar-contratos y firmar-contrato de manera integrada con los servicios que consumen. Por lo tanto, se decidió producir dos requisitos de prueba, uno para cada WC. El primero, enfocado en verificar la correctitud funcional del WC listar-contratos en relación a su interacción con el AS listar-contratos, se lo etiquetó como "TR-Integration-ListContracts" (ver Tabla 5-6). El segundo, con el objetivo de verificar la correctitud funcional del WC firmar-contratos en relación a su interacción con el AS firmar-contratos, se lo denominó "TR-Integration-SignContract" (ver Tabla 5-7).

Además, tanto la meta de prueba como la situación particular a nivel de proyecto indican que el nivel de prueba (*test level*) a considerar es de integración, por ello se establece este nivel en ambos requisitos. Por ejemplo, para el TR-Integration-ListContracts, el nivel de integración establece que se deben probar el WC listar-contratos y el AS listar-contratos en conjunto y sin utilizar *mocks* u otros objetos que simulen el comportamiento de ellos ya que estos se utilizan generalmente en pruebas unitarias. A su vez, es importante mencionar que todos los objetos a ser probados estaban en la fase de desarrollo del ciclo de vida del software.

Tabla 5-6: Especificación del requisito de prueba "TR-Integration-ListContracts". Tabla tomada de (Eleicegui, 2022).

Label: TR-Integration-ListContracts.
Statement: Se requiere verificar la correctitud funcional del **WC listar-contratos** en relación a su interacción con el **AS listar-contratos**.
Testable entity phase: Desarrollo.
Test level: Integración.
Completion Criteria: Se debe desarrollar al menos un caso de prueba para cada una de las siguientes situaciones:

- El servicio de la API respondió correctamente y el WC recibe un código 200 acompañado de un mensaje 'Success' y la lista de contratos filtrada y ordenada según lo esperado.
- Sucedió un error de parámetros en la URL, el WC recibe un código de error 404 acompañado de un mensaje 'Param Error.'

Refers to Testable Entities: WC listar-contratos, AS listar-contratos.
Refers to Test Context Entities: Test Database (Test DB)

Para el primer requisito (TR-Integration-ListContracts), se produjo la siguiente declaración: "Se requiere verificar la correctitud funcional del WC listar-contratos en relación a su interacción con el AS listar-contratos". Luego, para establecer los criterios de finalización, se tuvieron en cuenta las bases de prueba especificadas en la Tabla 5-2 y la Tabla 5-4 (recordar Sección 5.5.1.2). Notar que, según la Tabla 5-4, el WC listar-contratos tiene un nivel de impacto Alto (3) y, de acuerdo a la Tabla 5-2, para dicho impacto deben ser considerados las respuestas de la API con códigos 200 y 404 (este último también se considera ya que el AS listar-contratos posee parámetros en la URL).

Siguiendo con el ejemplo para el primer requisito de prueba, el mismo documenta las entidades comprobables *targets*, a saber: el WC listar-contratos y al AS listar-contratos. Además, se debe tener en cuenta una base de datos de prueba configurada, la cual tiene semántica de entidad de contexto de prueba.

Tabla 5-7: Especificación del requisito de prueba "TR-Integration-SignContract". Tabla tomada de (Eleicegui, 2022).

Label: TR-Integration-SignContract
Statement: Se requiere verificar la correctitud funcional del **WC firmar-contratos** en relación a su interacción con el **AS firmar-contratos**.
Testable entity phase: Desarrollo
Test level: Integración
Completion Criteria: Se debe desarrollar al menos un caso de prueba para cada una de las siguientes situaciones:

- El servicio de la API respondió correctamente y el WC recibe un código 200 acompañado de un mensaje 'Success' y el contrato seleccionado figura como correctamente firmado.
- Sucedió un error de parámetros en la URL, el WC recibe un código de error 404 acompañado de un mensaje 'Param Error'.
- El usuario no tiene el permiso para firmar el contrato, el WC recibe un código de error 403 acompañado de un mensaje 'Authorization Error: Permission not granted'.
- Falló la entidad de contexto que hace el envío de SMS, el WC recibe un error sin código con el mensaje 'SMS sending failed'.
- Falló la entidad de contexto que hace el estampado blockchain para la trazabilidad de la firma, el WC recibe un error sin código con el mensaje 'Stamping failed, sign cancelled'.

Refers to Testable Entities: WC firmar-contrato, AS firmar-contrato.
Refers to Test Context Entities: AS enviar-sms, Servicio externo de envío de SMS, Servicio externo de estampado blockchain, Test DB

Diseñar la Prueba (A2)

Una vez producidos los requisitos de prueba (Tabla 5-6 y Tabla 5-7), la siguiente actividad a realizar es "Diseñar la Prueba (A2)". Esta actividad contiene dos subactividades que se ejecutan iterativamente por cada requisito de prueba. En esta sección se ilustrará solamente la iteración para el requisito TR-Integration-SignContract (Tabla 5-7) aunque los detalles de la otra iteración se pueden encontrar en (Eleicegui, 2022).

La subactividad "Especificar Situaciones Particulares de Prueba (A2.1)" consume el requisito de prueba TR-Integration-SignContract, las bases de prueba (Sección 5.5.1.2) y la declaración positiva de la situación particular de prueba a nivel de proyecto anteriormente mencionada. El objetivo es producir modelos de situaciones particulares de prueba que estén alineadas con este requisito y situación de alto nivel. Estos modelos de situación se documentan haciendo uso de la plantilla SaSTMe (recordar Tabla 5-1).

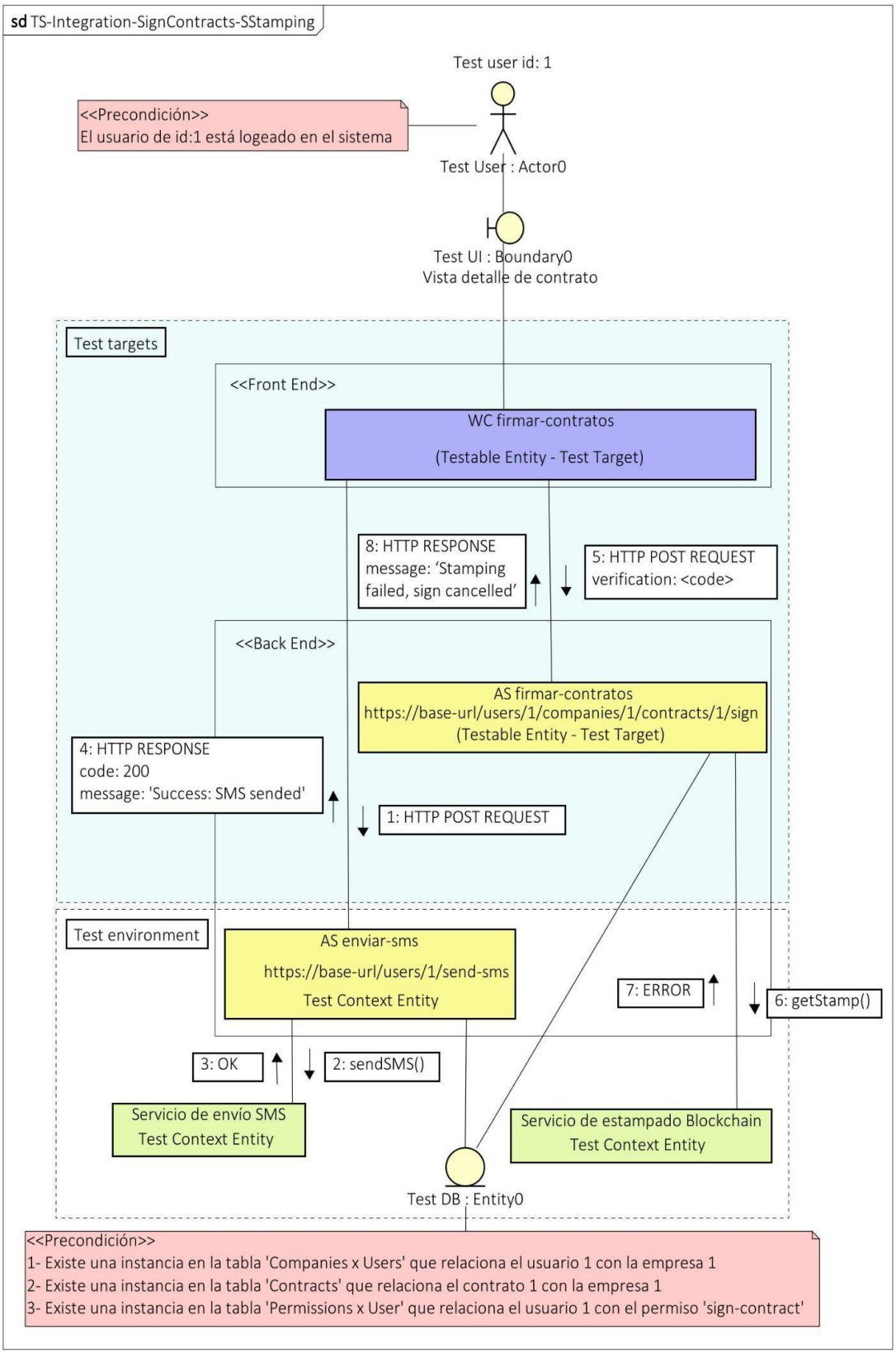
Observando los criterios de finalización del requisito TR-Integration-SignContract, los mismos presentan cinco situaciones diferentes a ser probadas para determinar el grado del cumplimiento de este requisito. Es decir, para cumplir este requisito en su totalidad se deberían producir y probar cada una de ellas. A continuación, se describe la situación producida para el último criterio, a saber: "Falló la entidad de contexto que hace el estampado *blockchain* para la trazabilidad de la firma, el WC recibe un error sin código con el mensaje Stamping failed, sign cancelled".

El escenario producido se lo llamó TS-Integration-SignContracts-SStamping (ver Tabla 5-8), y se lo especificó utilizando un diagrama de comunicación UML. Para producirlo se utilizaron las bases de prueba que se indican en la Tabla 5-8. Por ejemplo, se tuvo en cuenta la respuesta con código 200 de la Tabla 5-3; se seleccionó al usuario 1, empresa 1 y contrato 1 (notar que este contrato no está firmado) de los datos de prueba (Figura 5-2); también fue útil la información provista en la Tabla 5-4 que describe el WC

firmar-contratos en este caso y los detalles para los AS firmar-contratos y enviar-sms de la Tabla 5-5 (*endpoint*, descripción del método HTTP y servicios externos asociados). Además, para este escenario en particular las entidades de contexto son la base de datos de prueba; el AS enviar-sms; el servicio externo de envío de sms; y el servicio externo de estampado blockchain. Estas entidades se tendrían que tener en cuenta para establecer y configurar un entorno de prueba en la actividad A3 de SaSTPro.

Tabla 5-8: Plantilla SaSTMe rellenada con la situación particular de prueba llamada TS-Integration-SignContracts-SStamping. Además, la plantilla especifica las entidades, requisito y bases de prueba asociadas a esta situación. Tabla tomada de (Eleicegui, 2022).

Test Particular Situation: TS-Integration-SignContracts-SStamping -*positive statement*: Con el objetivo de comprobar que se está procesando correctamente una falla externa relacionada al estampado *blockchain*, en el WC firmar-contratos se genera un *request* HTTP con el método POST al endpoint /base-url/users/user_id/send-sms, instanciando la variable user_id con el id del usuario *logueado*. El AS enviar-sms recibe la petición, la procesa, genera un código de validación de 6 dígitos e indica al servicio externo de SMS que lo envíe al número de teléfono asociado al usuario recibido en el *request*. El servicio externo de SMS responde correctamente y notifica al AS, el cual genera una respuesta para el WC con la siguiente estructura: {code:200, message:'Success: SMS sended'}. El WC firmar-contratos procesa la respuesta de la API y en función al *success* hace visible al usuario un formulario para ingresar el código de verificación. El usuario ingresa correctamente el código recibido y presiona el botón "firmar" (se supone que el usuario va a ingresar correctamente el código recibido ya que el objeto principal de prueba está en la comunicación *frontend*/*backend* por lo que los errores humanos quedan fuera de consideración en este escenario). El WC firmar-contratos genera un *request* HTTP con método POST al *endpoint* base-url/users/**user_id**/companies/**company_id**/contracts/**contract_id**/sign, instanciando las variables user_id, company_id y contract_id con los ids del usuario *logueado*, una empresa vinculada al usuario y el id del contrato correspondiente a la empresa que va a ser firmado, respectivamente. El AS firmar-contratos recibe la petición, la procesa e indica al servicio externo de estampado *blockchain* que genere una estampa y su correspondiente QR. El servicio externo de estampado responde con algún error o no responde. El AS genera una respuesta para el WC con la siguiente estructura: response:{message: 'Stamping failed, sign cancelled'}. -*model specification*:



Testable Entities: WC firmar-contratos (*test target*) y AS firmar-contratos (*test target*)

Test Context Entities: Test DB; AS enviar-sms; Servicio externo de envío de SMS; Servicio externo de estampado *blockchain*. **-*Influences*:** *Test DB*: contiene los datos de prueba cargados para que el AS firmar-contratos y el AS enviar-sms puedan realizar con éxito las consultas correspondientes. *AS enviar-sms*: servicio que recibe una petición del WC firmar-contratos para realizar una verificación con segundo factor de autenticación para el usuario correspondiente. *Servicio externo de envío de SMS*: recibe un código de validación de 6 dígitos del AS enviar-sms el cual es enviado al usuario correspondiente y luego notifica esta operación al AS enviar-sms. Este último genera una respuesta para el WC firmar-contratos el cual hace visible al usuario un formulario para ingresar el código de verificación. *Servicio externo de estampado blockchain*: recibe peticiones del AS firmar-contratos para solicitar que genere estampas con su correspondiente QR.
Test Requirement: TR-Integration-SignContracts (ver Tabla 5-7).
Test Basis: Descripción de las respuestas de los AS (Tabla 5-3); Datos de prueba (Figura 5-2); información de los WCs provista en la Tabla 5-4; información de los ASs provista en la Tabla 5-5.

Luego, haciendo uso de toda la información especificada en la Tabla 5-8 y la plantilla SaSTMe, se diseñó un caso de prueba al realizar la subactividad "Diseñar Casos de Prueba (A2.2)". Este se encuentra documentado en la Tabla 5-9 y se lo llamo "TC-Integration-SignContracts-SStamping". Observando el diagrama que modela la situación y el caso de prueba producido se puede ver como ellos están relacionados. Por ejemplo, las precondiciones del caso de prueba se corresponden con las precondiciones de la situación.

Tabla 5-9: Plantilla SaSTMe rellenada con el caso de prueba llamado TC-Integration-SignContracts-SStamping. Tabla tomada de (Eleicegui, 2022).

Test Case: TC-Integration-SignContracts-SStamping. -*input*: { usuario id: 1, company id: 1, contract id: 1 } -*expected result*: message: 'Stamping failed, sign cancelled. -*preconditions*: El usuario de id = 1 debe estar correctamente logueado, poseer el permiso para firmar contratos, estar vinculado a la empresa de id = 1 y estar posicionado en la vista de firma de contrato para el contrato de id 1 de la aplicación Web.

En este caso aplicado solo se ejecutó el proceso SaSTPro hasta esta actividad. Además, en esta sección solo se mostraron algunos productos de trabajo producidos con fines ilustrativos. Notar que, como ya se mencionó anteriormente, el resto de las especificaciones de los demás artefactos producidos (i.e., el resto de las situaciones y casos de prueba) se encuentran en (Eleicegui, 2022).

5.5.2 Caso Aplicado 2 de la Estrategia SaST

La estrategia SaST se utilizó en otro caso aplicado (Fernandez Reale, 2023) para verificar una aplicación web de una inmobiliaria de la localidad de Santa Rosa. El objetivo era probar la aplicación desde un enfoque de pruebas de sistema para comprobar su correcto funcionamiento de acuerdo a los requisitos funcionales. Notar que, la aplicación web de la inmobiliaria como un todo es la entidad comprobable en este caso aplicado ya que se llevaran a cabo pruebas a nivel de sistema. Además, esta aplicación se encontraba en la etapa de desarrollo en ese momento.

Se probaron cuatro funcionalidades de la aplicación web haciendo uso de la estrategia SaST, a saber: iniciar sesión; nueva publicación; modificar publicación; y contactar a la inmobiliaria por correo electrónico (ver Figura 5-3). En esta sección se mostrarán los detalles de la aplicación de la estrategia solamente para la funcionalidad "contactar a la inmobiliaria por correo electrónico". No obstante, la ejecución de las actividades de SaSTPro para el resto de las funcionalidades y sus artefactos de prueba asociados se describen en (Fernandez Reale, 2023).

5.5.2.1 Bases de Prueba

Esta sección tiene como propósito ilustrar las bases de prueba utilizadas para el diseño de las pruebas de la funcionalidad "contactar a la inmobiliaria por correo electrónico". Una de ellas es el diagrama de casos de uso que se muestra en la Figura 5-3. Tener en cuenta que, de ahora en adelante, la entidad comprobable se la nombrará como "Inmobiliaria X".

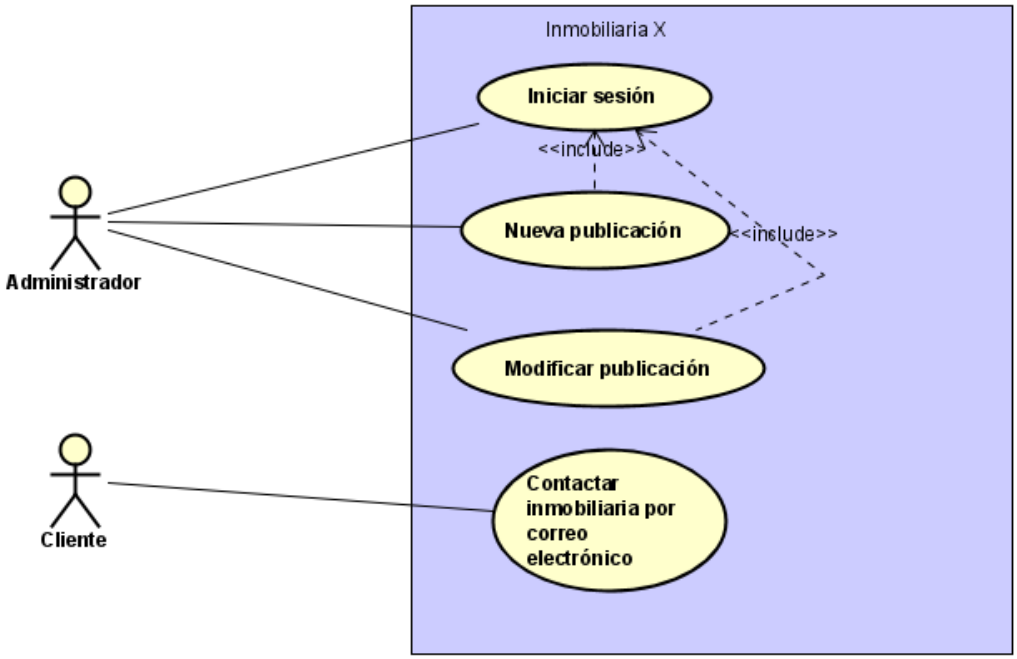


Figura 5-3: Diagrama de casos de uso de las funcionalidades a probar en el caso aplicado 2. Figura tomada de (Fernandez Reale, 2023).

También se consideró la historia de usuario que se especifica en la Tabla 5-10. Según la historia de usuario, se espera que la funcionalidad permita enviar un correo electrónico al administrador de la inmobiliaria. Sin embargo, deben cumplirse ciertas validaciones, por ejemplo, al observar la Tabla 5-11 se puede ver que los tres primeros campos de la inmobiliaria tienen validaciones como campos obligatorios y deben cumplir con ciertos formatos.

Finalmente, la Figura 5-4 muestra el esquema de la vista del formulario para poder contactarse con la inmobiliaria. Recordar que todo lo documentado aquí forma parte de las bases de prueba, las cuales serán consumidas por las diferentes actividades de SaSTPro para poder producir, por ejemplo, los requisitos de prueba y los casos de prueba. A continuación, en la próxima subsección, se describen las actividades de SaSTPro ejecutadas en este caso aplicado para verificar la funcionalidad "contactar a la inmobiliaria por correo electrónico".

Tabla 5-10: Historia de usuario de la funcionalidad “contactar inmobiliaria por correo electrónico”. Tabla tomada de (Fernandez Reale, 2023).

Historia de usuario	
Número: 5	**Usuario:** cliente
Nombre de historia: contactar inmobiliaria por correo electrónico	
Prioridad en negocio: alta	**Riesgo en desarrollo:** bajo
Puntos estimados: 6	**Iteración asignada:** 2
Programador responsable: Leonel Fernandez	
Descripción: quiero poder enviar un correo electrónico a la inmobiliaria con la información/consulta.	
Observaciones: para poder realizar correctamente la funcionalidad deben cumplirse ciertas validaciones en el formulario (ver Tabla 5-11).	
Criterios de aceptación: ● Solo es posible comunicarse con la inmobiliaria por medio de un correo electrónico cuando se cumplen las validaciones de los formularios. En caso contrario, el sistema debe emitir un mensaje de error y no debe permitir enviar el correo electrónico.	

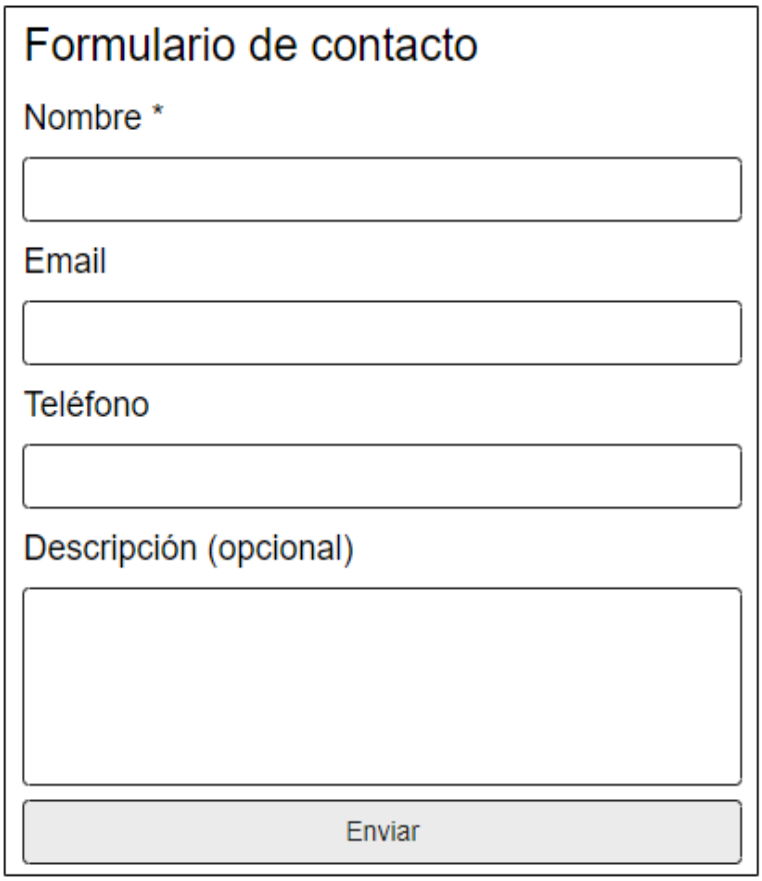


Figura 5-4: Wireframe del formulario de contacto. Figura tomada de (Fernandez Reale, 2023).

Tabla 5-11: Tabla de validaciones del formulario de contacto que se muestra en la Figura 5-4. Tabla tomada de (Fernandez Reale, 2023).

Formulario	Campo
Contacto	● Nombre ○ Campo obligatorio. ● E-mail ○ Debe tener formato de correo electrónico ○ Obligatorio en caso que el usuario no ingrese el número de teléfono. ● Teléfono ○ Debe tener formato de número de teléfono. ○ Obligatorio en caso que el usuario no ingrese el correo electrónico.

5.5.2.2 Ejecución de SaSTPro

Definir Requisitos de Prueba (A1)

Recordar que, según SaSTPro (ver Figura 5-1), la primera actividad a realizar es "Definir Requisitos de Prueba (A1)". A1 requiere como entrada documentos a nivel de proyecto de prueba, a saber: la meta de prueba y la declaración positiva de la situación particular de prueba, y también las bases de prueba. El objetivo es producir los requisitos de prueba, para los cuales se debe especificar su etiqueta, declaración, fase de la entidad comprobable, nivel de prueba y sus criterios de finalización (recordar Figura 4-3).

Se especificó la siguiente declaración de meta de prueba en este caso aplicado: "probar la aplicación Inmobiliaria X desde un enfoque de pruebas de sistema para verificar su correcto funcionamiento de acuerdo a los requisitos funcionales". Por otro lado, la declaración positiva de la situación particular de prueba de alto nivel establecía que "un usuario administrador tiene acceso a las funcionalidades de iniciar sesión, nueva publicación, y modificar publicación. Respecto a los usuarios clientes, pueden contactar a la inmobiliaria por medio de la funcionalidad de enviar un correo electrónico. En todos los casos se espera que las funcionalidades estén libres de error".

Observando la declaración positiva de la situación particular de prueba de alto nivel se distinguen cuatro funcionalidades que se espera que estén libres de error. Notar que estas funcionalidades coinciden con las que se muestran en la base de prueba especificada en el diagrama de casos de uso de la Figura 5-3. Teniendo en cuenta esto, se decidió producir cuatro requisitos de prueba en este caso aplicado, uno por cada funcionalidad. La Tabla 5-12 especifica el requisito de prueba llamado "TR-System-ContactarInmobiliaria", mientras que los otros requisitos se pueden encontrar en (Fernandez Reale, 2023). Para todos los requisitos producidos la aplicación a ser probada (*test target*) es la Inmobiliaria X y el nivel de prueba es de sistema, tal como se establece en la meta de prueba. Además, recordar que anteriormente se mencionó que la aplicación se encontraba en la etapa de desarrollo.

La Tabla 5-11 contiene las validaciones que deben cumplir algunos campos del formulario de la Figura 5-4 para la funcionalidad de contactar a la inmobiliaria por correo electrónico. Haciendo uso de esta información se propusieron los criterios de finalización para el requisito TR-System-ContactarInmobiliaria. Por ejemplo, dado que hay campos que deben ingresarse obligatoriamente para poder usar esta funcionalidad, una posible situación podría ser cuando el usuario completa solamente los campos obligatorios. También podría darse el escenario que se olvide de completar algún campo obligatorio y

quiera hacer uso de la funcionalidad. En esta última situación, el sistema debe emitir un mensaje de error (ver criterios de aceptación de la historia de usuario 5 en la Tabla 5-10).

Tabla 5-12: Especificación del requisito de prueba "TR-System-ContactarInmobiliaria". Tabla tomada de (Fernandez Reale, 2023).

Label: TR-System-ContactarInmobiliaria.
Statement: se requiere verificar la correctitud de la funcionalidad que permite al cliente contactar a la inmobiliaria por medio de un correo electrónico.
Testable Entity Phase: desarrollo.
Test level: sistema.
Completion Criteria: se debe desarrollar al menos un caso de prueba para cada una de las siguientes situaciones:

- El usuario cliente contacta a la inmobiliaria satisfactoriamente sólo considerando los campos obligatorios (ver Tabla 5-11).
- El usuario cliente contacta a la inmobiliaria satisfactoriamente rellenando todos los campos disponibles en el formulario.
- El usuario no cumple con la validación de un campo y no puede realizar su consulta. El sistema muestra un mensaje de error al respecto (ver Tabla 5-11).
- El usuario no cumple con la validación de un campo, corrige y logra contactarse con la inmobiliaria exitosamente (ver Tabla 5-11).
- El usuario intenta contactar a la inmobiliaria y no lo logra debido a que el servidor de correos no funciona correctamente. El sistema debe advertir al usuario que ocurrió un error.

Refers to Testable Entity: Inmobiliaria X.
Refers to Test Context Entities: servidor de correos electrónicos y correos de pruebas.

Dado que esta funcionalidad consiste en enviar un correo electrónico al administrador de la inmobiliaria, se requiere entonces un servidor de correo electrónico configurado para que la misma funcione correctamente. Este servidor se considera como entidad de contexto de prueba y también se diseñó una situación en el que esta entidad falle. A continuación, se ilustrará esta situación junto al caso de prueba producido usando la plantilla SaSTMe.

Diseñar la Prueba (A2)

Una vez producidos los requisitos en A1, la siguiente actividad a llevar a cabo es "Diseñar la Prueba (A2)". Recordar que esta actividad implica otras dos subactividades que se ejecutan iterativamente por cada requisito de prueba. En esta sección se ilustrará solamente la iteración para el requisito TR-System-ContactarInmobiliaria (Tabla 5-12) aunque los detalles de las otras iteraciones se pueden encontrar en (Fernandez Reale, 2023).

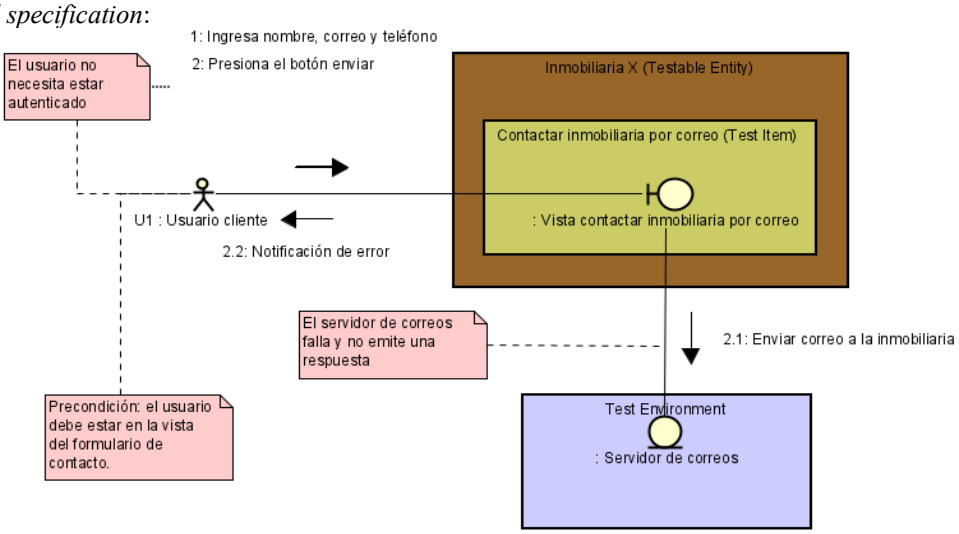
Primero, en la subactividad "Especificar Situaciones Particulares de Prueba (A2.1)" se consume el requisito de prueba TR-System-ContactarInmobiliaria, las bases de prueba (Sección 5.5.2.1) y la declaración positiva de la situación particular de prueba a nivel de proyecto anteriormente mencionada. Como resultado, se produce la situación particular de prueba que se muestra en la Tabla 5-13, la cual se encuentra documentada haciendo uso de la plantilla SaSTMe.

Por un lado, se tuvo en cuenta el último criterio de finalización del requisito TR-System-ContactarInmobiliaria ("el usuario intenta contactar a la inmobiliaria y no lo logra debido a que el servidor de correos no funciona correctamente. El sistema debe advertir al usuario que ocurrió un error") para diseñar la situación que se especificó en un diagrama de comunicación UML. También fue útil contar con el Wireframe de contactar inmobiliaria por correo (Figura 5-4) y las reglas de validación asociadas del formulario (Tabla 5-11) para poder diseñar una consulta valida desde el usuario cliente al sistema (ver mensaje 1 en el diagrama UML).

Como se puede ver en la Tabla 5-13, la entidad comprobable es el sistema como un todo (Inmobiliaria X) ya que el nivel de prueba es de sistema. Sin embargo, en este punto del proceso SaSTPro es importante mencionar cuál es el *test target* de la situación, tal como lo indica la plantilla SaSTMe (Tabla 5-1). En este caso, que el nivel de prueba sea de sistema indica que las pruebas se van a ejecutar sobre una instancia del sistema completa y no sobre un módulo en particular de forma aislada (nivel unitario), o dos o más módulos en conjunto (nivel de integración como el caso aplicado anterior). Además, es importante mencionar cuál es el foco a ser probado de la situación, que en este caso particular es la funcionalidad contactar inmobiliaria por correo y todo lo que ella implica, a saber: una interfaz de usuario con un formulario y sus validaciones, y un componente que realizará una petición a un servidor de correo electrónico usando los datos brindados en dicho formulario.

Tabla 5-13: Plantilla SaSTMe rellenada con la situación particular de prueba llamada TPS#5-ContactarInmobiliaria. Además, la plantilla especifica las entidades, requisito y bases de prueba asociadas a esta situación. Tabla tomada de (Fernandez Reale, 2023).

Test Particular Situation: TPS#5-ContactarInmobiliaria
-positive statement: El servidor de correos falla en el momento que el usuario intenta contactarse con la inmobiliaria por medio de un correo electrónico.
-model specification:



-conditions:
-precondiciones: el usuario cliente debe estar en la vista del formulario de contacto. Además, tiene que haber un servidor de correos electrónicos configurado con un correo emisor y otro receptor. Este no responde a las solicitudes que realiza el sistema debido a un problema propio del servidor, ajeno a la aplicación.
-postcondiciones: el usuario cliente no logra contactarse con la inmobiliaria y puede identificar una notificación advirtiendo que la operación no pudo realizarse.

Testable Entity: Inmobiliaria X
Test Item: funcionalidad contactar inmobiliaria por correo.
Test Target: funcionalidad contactar inmobiliaria por correo.

Test Context Entities: servidor de correos electrónicos.

-Influences:
Servidor de correos electrónicos: el servidor de correo envía un correo electrónico al administrador de la inmobiliaria y notifica al sistema de la inmobiliaria que este correo fue enviado. Tener en cuenta que este servidor debe tener configurado un correo emisor y otro receptor para realizar las pruebas.

Test Requirement: TR-System-ContactarInmobiliaria (Tabla 5-12).

Test Basis: Diagrama de casos de uso (Figura 5-3); Wireframe de contactar inmobiliaria por correo (Figura 5-4); Historia de usuario de la funcionalidad contactar inmobiliaria por correo electrónico (Tabla 5-10); Tabla de validaciones del formulario de contacto (Tabla 5-11).

La siguiente subactividad es "Diseñar Casos de Prueba (A2.2)", la cual utiliza toda la información ya especificada en la Tabla 5-13 y la plantilla SaSTMe. Como resultado de esta actividad se produjo el caso de prueba TC#5.1-ContactarInmobiliaria que se muestra en la Tabla 5-14. Como se puede ver en la entrada del caso de prueba, se tienen en cuenta los campos nombre, correo y teléfono, tal como se indica en el mensaje 1 del modelo de situación de TPS#5-ContactarInmobiliaria. Además, se considera el "clic" en el botón "enviar" (ver mensaje 2 del modelo de situación).

Tabla 5-14: Plantilla SaSTMe rellenada con el caso de prueba llamado TC#5.1-ContactarInmobiliaria. Tabla tomada de (Fernandez Reale, 2023).

Test Case: TC#5.1-ContactarInmobiliaria.
-*input*:
nombre: "Test FR#4 TPS#5 TC#1",
correo: "test@gmail.com",
telefono: "12345678"
Presiona el botón enviar.
-*expected result*: el sistema no logra enviar el correo electrónico con los datos suministrados por el cliente debido a que el servidor de correos falló.
-*preconditions*: el usuario debe estar en la vista del formulario de contacto. El servidor de correos no responde a las solicitudes que realiza el sistema y tiene que estar configurado con los siguientes correos: contactotest@test.com e inmobiliariatest@test.com.
-*postconditions*: el usuario cliente puede identificar una notificación advirtiendo que la operación no pudo realizarse.

A su vez, las condiciones de la situación fueron utilizadas para diseñar las precondiciones y postcondiciones del caso de prueba. Notar que aquí se establecieron un correo electrónico emisor y otro receptor específicos, a saber: contactotest@test.com e inmobiliariatest@test.com. En este punto finaliza A2.2 para la iteración que considera el requisito TR-System-ContactarInmobiliaria. Recordar que en esta sección únicamente se ilustra la iteración de dicho requisito y las restante se pueden encontrar en (Fernandez Reale, 2023).

Luego de que termine el ciclo A2.1-A2.2 para todos los requisitos, se debe ejecutar "Definir Procedimientos de Realización (A2.3)". Esta actividad utiliza toda la información ya capturada en la plantilla SaSTMe, es decir, la situación particular de prueba (Tabla 5-13) y el caso de prueba (Tabla 5-14) ya producidos. Como resultado, se produce la automatización de esta situación que considera dicho caso de prueba (ver Figura 5-5) y los requisitos del entorno de prueba (ver Tabla 5-15). Es importante mencionar que se utilizó Cypress como marco de trabajo para la automatización de las pruebas.

En la implementación de la situación particular de prueba de la Figura 5-5 se puede ver que en la porción de código "*Arrange*" se obtienen inicialmente los datos de entradas correspondientes (observar la línea que contiene el *dataSet*). Posteriormente, se ejecuta la función "*intercept*". Notar la importancia de esta función ya que permite simular una respuesta del servidor de correos. En este caso, se crea un interceptor que al identificar una solicitud de tipo "POST" y con la URL "/api/ContactoMail/SendMail" debe responder "statusCode: 500". Además, al interceptor se le asigna el alias "reqError".

```
describe('TPS#5', () => {

  beforeEach(function () {
    //Arrange (Preparar)
    cy.disableUncaughtException();
    cy.fixture('contactar-inmobiliaria').then((dataSet) => this.dataSet = dataSet);
    cy.intercept(
      'POST',
      '/api/ContactoMail/SendMail',
      { statusCode: 500 }
    ).as('reqError')
    /* Precondicion: el usuario debe estar en la vista del formulario de contacto.
     El servidor de correos no responde a las solicitudes que realiza el sistema
     y tiene que estar configurado con los siguientes correos: contactotest@test.com y
    inmobiliariatest@test.com */
    cy.visit('/contacto');
    cy.get('#titulo-contacto').should('contain', 'Formulario de Contacto');
  })

  it('Usuario contacta a la inmobiliaria y el servidor de correos falla', function() {
    //Act (Actuar)
    cargarFormularioContacto(this.dataSet[6]);
    cy.get("#enviar").click()

    cy.wait('@reqError')
      .then(interception => {
        /* Resultado esperado: el sistema no logra enviar el correo electrónico con los
        datos suministrados por el cliente debido a que el servidor de correos falló. */
        expect(interception.response.statusCode).to.equal(500);
      });

    //Assert (Afirmar)
    /* Postcondicion: el usuario cliente puede identificar una notificación advirtiendo
     que la operación no pudo realizarse. */
    cy.get(".swal2-error").should("exist");
  })
})
```

Figura 5-5: Especificación del procedimiento de realización para la TPS#5-ContactarInmobiliaria. Figura tomada de (Fernandez Reale, 2023).

Tabla 5-15: Requisitos del entorno de prueba para la funcionalidad contactar inmobiliaria por correo electrónico. Tabla tomada de (Fernandez Reale, 2023).

Entidad de contexto de prueba	**Requisitos del entorno de prueba**
Servidor de correo electrónico	Debe tener configurado dos correos electrónicos: contactotest@test.com e inmobiliariatest@test.com.

Antes de comenzar a interactuar con la funcionalidad, se debe cumplir con la precondición “el usuario debe estar en la vista del formulario de contacto. El servidor de correos no responde a las solicitudes que realiza el sistema y tiene que estar configurado con los siguientes correos: contactotest@test.com e inmobiliariatest@test.com”. Por un lado, para cumplir con la condición que el usuario se encuentre en la vista del formulario de contacto, en el código se realiza una aserción con la URL “/contacto” y, además, debe existir el título “Formulario de contacto”. Respecto a la precondición relacionada al servidor de correos, se debe configurar la entidad de contexto de prueba con los requisitos mencionados en la Tabla 5-15.

Una vez finalizada la etapa de preparación (*Arrange*), se puede avanzar a la porción “*Act*”. En esta sección solo se ejecutan dos acciones. Primero se realiza la carga del formulario con los datos de entrada correspondiente (ver *input* en Tabla 5-14) y luego se presiona el botón enviar.

En la etapa “*Assert*” se verifica que el resultado esperado y la postcondición se cumplan. El caso de prueba menciona como resultado esperado lo siguiente: “el sistema

no logra enviar el correo electrónico con los datos suministrados por el cliente debido a que el servidor de correos falló". Por lo tanto, se debe hacer uso del interceptor creado al comienzo de la automatización. Para ello se ejecuta "*cy.wait('@reqError')*" que tiene la finalidad de escuchar las solicitudes que realiza la aplicación y al encontrar que se envió el correo electrónico entonces verifica que el resultado sea un error con código 500.

Luego, la implementación comprueba que sea visible una notificación de error. De esta manera logra verificar la postcondición "el usuario cliente puede identificar una notificación advirtiendo que la operación no pudo realizarse" y finaliza el caso de prueba. En este punto finaliza A2.3.

Establecer el Entorno de Prueba (A3)

La siguiente actividad a llevar a cabo es A3 ya que se debe establecer un entorno de prueba. En la actividad anterior se definió, en la Tabla 5-15, cual es la entidad de contexto y sus requisitos asociados para la funcionalidad contactar inmobiliaria por correo electrónico.

En resumen, se debe configurar un servidor de correo electrónico con dos correos, a saber: contactotest@test.com e inmobiliariatest@test.com. El correo contactotest@test.com tiene como objetivo enviar correos electrónicos con los datos suministrados por el cliente al administrador de la inmobiliaria. Por su parte, inmobiliariatest@test.com es el receptor de los correos enviados con los datos de los clientes. Una vez que se implementó esta configuración, se dio por finalizada A3.

Realizar la Prueba Dinámica (A4)

En A4 se llevan a cabo las pruebas ya diseñadas e implementadas ejecutándolas sobre la entidad comprobable en los entornos previstos. Siguiendo con el ejemplo anterior, en esta actividad se ejecutó el caso de prueba TC#5.1-ContactarInmobiliaria (Tabla 5-14) sobre la entidad comprobable que es la aplicación web de la inmobiliaria. Recodar que este caso de prueba fue automatizado usando Cypress (Figura 5-5).

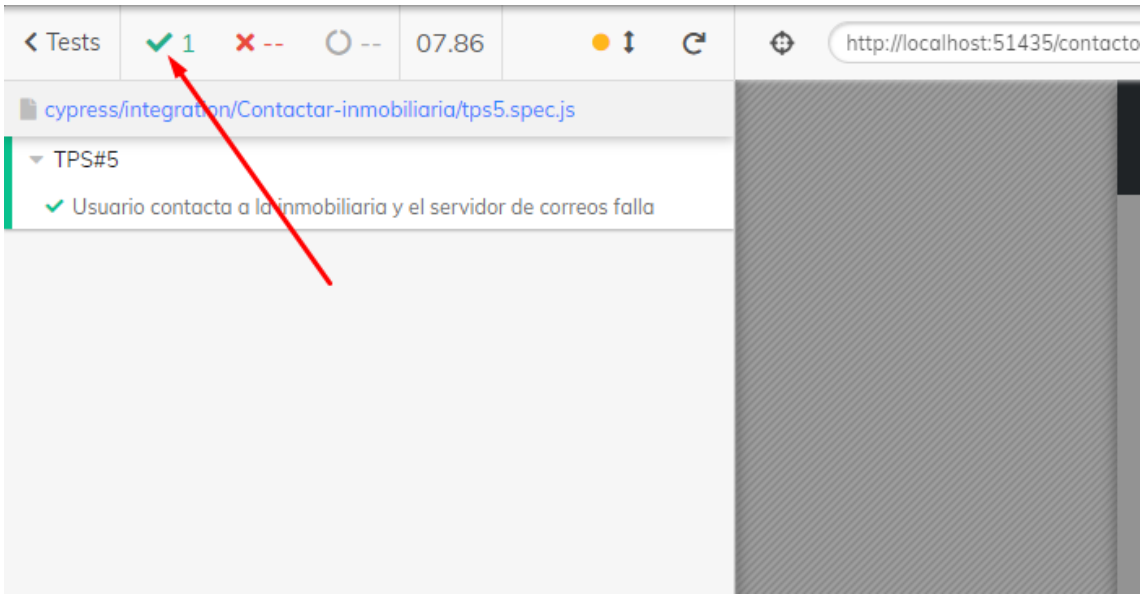


Figura 5-6: Captura de pantalla de la ejecución de TPS#5-ContactarInmobiliaria (y el caso de prueba asociado TC#5.1-ContactarInmobiliaria) para la funcionalidad contactar inmobiliaria por correo electrónico. Figura tomada de (Fernandez Reale, 2023).

Se puede observar en la Figura 5-6 que el caso de prueba TC#5.1-ContactarInmobiliaria para la situación TPS#5 no falló al ser ejecutado. Esta información

representa el resultado de prueba producido en A4. Luego de ejecutar el resto de los casos de prueba y obtener todos los resultados, finaliza A4.

Analizar los Resultados de Prueba (A5)

En esta última actividad se utilizan muchos de los artefactos producidos/consumidos a lo largo del proceso para analizar los resultados obtenidos de todas las pruebas y generar un informe de conclusión de prueba. En resumen, se consumieron en A5 todas las bases de prueba utilizadas, todos los requisitos de prueba producidos, todos los casos de prueba diseñados, todos los resultados de prueba obtenidos en A4 y, además, se utilizó una especificación de necesidad de información de prueba (ver Tabla 5-16). Tener en cuenta que en esta tesis solo se muestran algunos de los artefactos consumidos por A5 correspondientes a una sola situación para el caso de contactar inmobiliaria por correo. El resto de los artefactos insumidos se pueden encontrar documentados en (Fernandez Reale, 2023).

Tabla 5-16: Especificación de necesidad de información de prueba para el caso aplicado de la inmobiliaria. Tabla tomada de (Fernandez Reale, 2023).

Label: test-inmobiliaria **Statement**: analizar la correctitud de las funcionalidades seleccionadas de la aplicación web de la inmobiliaria. **Purpose:** analizar

De acuerdo a la declaración de la meta de necesidad de información de prueba (Tabla 5-16) se debe "analizar la correctitud de las funcionalidades seleccionadas de la aplicación web de la inmobiliaria". Recordar que las funcionalidades seleccionadas son cuatro, a saber: iniciar sesión; nueva publicación; modificar publicación; y contactar inmobiliaria por correo electrónico (Figura 5-3). Analizando todas las pruebas ejecutadas se verificó la correctitud de las cuatro funcionalidades, alcanzando una cobertura de correctitud funcional del 100% en este caso aplicado (Fernandez Reale, 2023). Notar que en esta tesis solo se mostró un ejemplo para la funcionalidad contactar inmobiliaria por correo electrónico. Si solo se hubieran ejecutado pruebas para esta funcionalidad, se hubiera alcanzado solamente una cobertura funcional del 25%.

Tabla 5-17: Informe de conclusión de prueba para el caso aplicado de la inmobiliaria. Tabla tomada de (Fernandez Reale, 2023).

Name: RCP#1
Version: 1.0
Description: ● Funcionalidad iniciar sesión: TPS#1, TPS#2 y TPS#3 pasaron satisfactoriamente las pruebas. ● Funcionalidad nueva publicación: TPS#2, TPS#3 y TPS#4 pasaron satisfactoriamente las pruebas. Por otro lado, TPS#1 y TPS#5 no lograron pasar debido a un error en la lógica de la aplicación. ● Funcionalidad contactar inmobiliaria por correo electrónico: TPS#1, TPS#2, TPS#3, TPS#4 y TPS#5 pasaron satisfactoriamente. ● Funcionalidad modificar publicación: TPS#1, TPS#2, TPS#3, TPS#4 y TPS#5 pasaron satisfactoriamente las pruebas. Como conclusión, se verificó la correctitud de todas las funcionalidades seleccionadas. Además, se realizó la cobertura del 100% de los criterios de finalización de los requisitos de prueba. Respecto a los resultados de prueba, el 88% de las situaciones particulares de prueba (16 de 18) pasaron de forma exitosa.

Además, también es importante considerar los criterios de finalización de los requisitos de prueba. En este caso aplicado (Fernandez Reale, 2023), se diseñaron situaciones y

casos de prueba para todos los escenarios propuestos de todos los requisitos producidos. En esta tesis se ilustró solamente el cumplimiento de uno de los cinco criterios de finalización para el requisito TR-System-ContactarInmobiliaria (Tabla 5-12).

Respecto a los resultados de prueba obtenidos en A4 (es decir, la comparación entre los resultados esperados de los casos de pruebas y los resultados reales), se detectó que el sistema se comportó de manera inesperada solamente en dos situaciones de prueba para la funcionalidad nueva publicación. Por lo tanto, el 88% (16 de 18) de todas las situaciones particulares de prueba diseñadas y ejecutadas fueron satisfactorias. La Tabla 5-17 contiene el informe de conclusión de prueba que resumen toda esta información.

5.6 Conclusiones

Para concluir este capítulo, aquí se ilustraron dos aplicaciones de la estrategia SaST mostrando algunos de los artefactos producidos. Por un lado, la primera aplicación se realizó con el objetivo de diseñar escenarios y casos de prueba a nivel de integración. En este primer caso las entidades comprobables objetivo (*test targets*) de las situaciones producidas fueron una aplicación web y una API *backend*. Esta plataforma web está destinada a la gestión de contratos inteligentes y pagos digitales. Una limitación de este primer caso aplicado de SaST es que solo se llevaron a cabo las actividades A1, A2.1 y A2.2.

Por otra parte, en la segunda aplicación de SaST se ejecutó el proceso completo, incluyendo la actividad opcional A3. A su vez, a diferencia del caso anterior, se diseñaron pruebas a nivel de sistema para una aplicación web de una inmobiliaria. Dicha aplicación permite gestionar publicaciones para que los clientes puedan visualizar los inmuebles disponibles para alquilar y comprar.

Como valoraciones personales respecto de la estrategia integrada SaST, ambos alumnos (Eleicegui, 2022; Fernandez Reale, 2023) coincidieron en que su proceso (SaSTPro) fue fácil de entender y seguir. Además, la ontología TestTDO brindó soporte conceptual al momento en que se les presentaron dudas con respecto a la semántica de algunos conceptos. Si bien SaST no contiene una plantilla para especificar algunos artefactos como por ejemplo los requisitos de prueba, los alumnos tomaron el modelo de TestTDO como referencia para especificar estos artefactos. Notar que el modelo conceptual de TestTDO (Figura 4-2) indica cuáles son los atributos/propiedades de cada artefacto de prueba (es decir, su estructura) y, además, estas propiedades están definidas.

Con respecto al método SaSTMe y su plantilla provista, ambos tesistas de grado acuerdan que les fue muy útil para el relevamiento de la información necesaria y suficiente a la hora de diseñar las pruebas. No obstante, en el primer caso aplicado (Eleicegui, 2022) el alumno sugirió agregar el atributo “condiciones” como una propiedad adicional de las situaciones de prueba ya que consideraba importante la especificación de precondiciones o postcondiciones en un escenario. Como beneficio de modelar dichas condiciones es que aportan más claridad a la hora de entender una situación particular y, a su vez, pueden tenerse en cuenta luego a la hora de diseñar los casos de prueba asociados. Esta sugerencia fue adoptada luego de la finalización del primer caso aplicado, por eso en la Tabla 5-8 la plantilla no tiene condiciones en la situación particular, sino que forman parte del modelo. En cambio, si se observa la Tabla 5-13 para el segundo caso aplicado, se puede ver en la plantilla SaSTMe que se especificaron precondiciones y postcondiciones como propiedades de la situación particular de prueba.

Como último comentario de esta sección, se notó una dificultad por el tesista de grado del segundo caso aplicado (Fernandez Reale, 2023) a la hora de diseñar y diferenciar el resultado esperado y la postcondición de algunos casos de prueba. TestTDO define ambas propiedades y la diferencia radica principalmente en que la postcondición es una restricción que debe evaluarse como verdadera y, en cambio, el resultado esperado es un valor o comportamiento concreto, es decir, no es una condición a ser evaluada.

Sim embargo, en algunos casos, el alumno planteaba que el resultado esperado puede redefinirse para ser expresado como una condición o viceversa. Por ejemplo, la postcondición del caso de prueba TC#5.1-ContactarInmobiliaria (Tabla 5-14) establece que "el usuario cliente puede identificar una notificación advirtiendo que la operación no pudo realizarse". Esto también puede definirse como un resultado esperado de la siguiente manera: "el sistema muestra una notificación advirtiendo que la operación no pudo realizarse". La conclusión es que ambas opciones son correctas ya que es una cuestión de diseño a la hora de producir los casos de prueba. Además, estos problemas pueden estar vinculados al diseño de casos de prueba a nivel de sistema por ser de más alto nivel, y no en casos de prueba a nivel unitario o de integración ya que el alumno anterior no tuvo dificultades en este punto.

CAPÍTULO 6: CONCLUSIONES

En este último capítulo se presentan las principales conclusiones de esta tesis. En la Sección 6.1 se resumen las oportunidades de mejora que se detectaron en el estado del arte, las cuales motivaron a la realización de este trabajo de investigación. Luego, en la Sección 6.2, se presentan los objetivos específicos descriptos en la Introducción y la correspondencia con las contribuciones realizadas que abordaron cada una de las oportunidades de mejoras detectadas. Además, se incluyó la Sección 6.3 que resume las publicaciones más relevantes asociadas a cada contribución, aunque también menciona otras publicaciones relacionadas que fueron previas y se publicaron en congresos nacionales o internacionales. Finalmente, en la Sección 6.4, se exponen algunas futuras líneas de investigación afines a la temática aquí presentada.

6.1 Oportunidades de Mejora de Aspectos Observados en el Estado en el Arte

Al llevar a cabo la investigación para el presente trabajo doctoral se detectaron oportunidades de mejora relacionadas a estrategias integradas de *testing*. Específicamente, algunas limitaciones observadas fueron las siguientes:

- Desde lo mejor del conocimiento del autor de esta tesis, actualmente no existen metodologías o estrategias integradas de *testing* que utilicen una ontología de pruebas de software como base conceptual. Aunque en la literatura se pueden encontrar estrategias como ISO 29119 o ISTQB, que consideran las tres capacidades, estas utilizan un glosario como base conceptual. No obstante, es importante destacar que una ontología es una estructuración semántica más rica que un glosario o taxonomía, ya que no solo define explícitamente los términos principales, sino también sus propiedades, relaciones y axiomas.
- Tras realizar una Revisión Sistemática de Literatura (RSL) sobre ontologías de pruebas de software se concluyó que no existe una ontología de *testing* adecuada que dé soporte a una familia de estrategias de pruebas de software. Al evaluar las soluciones actuales se detectaron algunas limitaciones como, por ejemplo, no tienen una buena cobertura del dominio considerando términos de pruebas funcionales, no funcionales, estáticas y/o dinámicas; no están directamente relacionadas con conceptos de requisitos no funcionales y/o requisitos funcionales; no tienen calidad estructural satisfactoria; no reutilizan una ontología fundacional; entre otros aspectos.
- A pesar de que existen muchos métodos o técnicas para las pruebas de software, no todos ellos son lo suficientemente adecuados para probar el software en diferentes situaciones en las cuales es importante modelar las entidades de contexto que influencian el objeto de prueba. Además, como mencionan los autores en (Amalfitano et al., 2019), hay un escaso número de tecnologías (es decir, métodos, herramientas, estrategias) que son útiles para probar situaciones en las que es importante considerar las entidades de contexto que interactúan con el objeto de prueba.

6.2 Contribuciones Realizadas

Con el fin de mejorar los aspectos observados en el estado del arte que se mencionaron en la sección anterior, a lo largo de este trabajo se han realizado aportes o contribuciones

que se listan en la Tabla 6-1. Notar que esta tabla tiene como objetivo que el lector pueda visualizar fácilmente en qué capítulo se abordó cada contribución de la presente tesis. Además, los objetivos específicos que se listaron en la Sección 1.2 fueron replicados en la Tabla 6-1 para facilitar la lectura.

Tabla 6-1: Capítulos de esta tesis asociados a cada objetivo y contribución.

Objetivos específicos (OE)	Contribuciones (C)	Capítulo de esta tesis que abordó la contribución asociada
OE_1: dado que la estrategia a desarrollar debe estar semánticamente soportada por una ontología, se debe investigar cuáles son las ontologías existentes para el dominio de las pruebas de software.	C_1: Una especificación de proceso para revisiones/mapeos sistemáticos de literatura.	**Capítulo 3: Estado del Arte de Ontologías de Pruebas de Software**
	C_2: Una Revisión Sistemática de Literatura sobre ontologías de pruebas de software.	
OE_2: Una vez conocido el estado del arte de las ontologías de pruebas de software (i.e., una vez alcanzado el OE_1), se debe adoptar/adaptar alguna existente si cumple con los requisitos establecidos. En caso de que ninguna ontología existente cumpla con los requisitos establecidos, se deberá desarrollar una nueva.	C_3: Una especificación de proceso para la metodología de investigación de *Design Science Research.*	**Capítulo 4: Construyendo una Ontología de Alto Nivel para el Dominio de las Pruebas de Software**
	C_4: Una nueva ontología de dominio de nivel superior para pruebas de software llamada TestTDO.	
OE_3: Haciendo uso de los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de proceso que considere actividades de *testing* de diseño, ejecución y análisis, y además especifique cuáles son sus entradas/salidas correspondientes. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.	C_5: Una especificación de proceso para la estrategia SaST haciendo uso del lenguaje SPEM, el cual considera perspectivas de proceso.	**Capítulo 5: SaST Strategy – una estrategia integrada de pruebas de software consciente de la situación y basadas en escenarios**
OE_4: Usando los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de método que este basado en la técnica de diseño de pruebas basada en escenarios. Además, debe dar soporte a situaciones que consideren entidades de contexto. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.	C_6: Una especificación de método para la estrategia SaST, la cual es una plantilla que captura información relacionada a la situación/escenario de prueba, entre otras cosas.	
OE_5: Una vez confeccionada la estrategia integrada, se debe validar la misma para inicialmente evaluar su utilidad. Notar que se deben haber alcanzado primero los objetivos OE_3 y OE_4 para poder lograr este objetivo.	C_7: la ilustración de la aplicación de la estrategia en un contexto real solo para el diseño de los casos de pruebas y escenarios.	
	C_8: la presentación de otra aplicación de la estrategia, también en un contexto real, la cual incluyo el diseño, implementación, automatización y ejecución de las pruebas.	

Como se mencionó en la sección anterior, no existen estrategias integradas de *testing* que consideren una ontología de pruebas de software como base conceptual. Por lo tanto,

las contribuciones C_4, C_5 y C_6 acortan esta brecha ofreciendo una primera estrategia de pruebas de software integrada y con una base conceptual ontológica como soporte semántico. También es importante mencionar que la estrategia SaST propuesta puede resultar adecuada para probar el software en diferentes situaciones en las cuales es importante modelar las entidades de contexto que influencian el objeto de prueba, acortando otra brecha adicional detectada en la literatura.

Además, como resultado de este trabajo, se construyó una nueva ontología de *testing* llamada TestTDO que se integró a la arquitectura ontológica FCD-OntoArch (contribución C_4), la cual tiene una amplia cobertura terminológica considerando conceptos de pruebas estáticas, dinámicas, funcionales y no funcionales, así como también conceptos relacionados a procesos, métodos, agentes, situaciones, entidades, requisitos, entre otros. A su vez, de acuerdo a los resultados de la evaluación documentada en la Sección 4.3.3.3, TestTDO tiene una calidad ontológica satisfactoria (98,11% ●) con respecto a las otras soluciones propuestas.

Otra contribución importante de esta tesis es C_2, la RSL sobre ontologías de pruebas de software. La misma permitió encontrar de una manera sistemática y rigurosa las ontologías de *testing* existentes en la literatura. Además, en este estudio secundario se realizó la evaluación de las ontologías encontradas, brindando información para tomar la decisión si se debería adoptar, adaptar o crear una nueva ontología de pruebas de software que de soporte semántico a la familia de estrategias de *testing*.

Por último, pero no menos importante, como contribuciones no directamente relacionadas con el objetivo general de este trabajo (recordar Sección 1.2), se mejoraron las especificaciones de procesos existentes para llevar a cabo estudios secundarios (contribución C_1) y para la metodología de investigación *Design Science Research* (C_3). Estas especificaciones fueron modeladas considerando diferentes perspectivas o vistas de modelado de proceso.

6.3 Publicaciones Relevantes

Como soporte a las contribuciones listadas en la Tabla 6-1, en esta sección se ilustran las publicaciones más relevantes que fueron producto de este trabajo de investigación. Las mismas se encuentran listadas en la 3er columna de la Tabla 6-2. Además, en dicha tabla, se relaciona cada publicación con cada objetivo y contribución previamente establecidos. Al igual que se hizo con la Tabla 6-1 y con la misma intención de facilitar la lectura, los objetivos específicos y las contribuciones fueron replicadas en la Tabla 6-2.

Es importante mencionar que en la Tabla 6-2 solo se muestran las publicaciones más relevantes, aunque también hubo otras publicaciones previas en congresos nacionales/internacionales relacionadas a ciertas contribuciones que fueron luego extendidas en artículos de revista. Empezando por la primera contribución (C_1: Una especificación de proceso para revisiones/mapeos sistemáticos de literatura), el proceso fue primero presentado en: **Tebes, G.**, Peppino, D., Becker, P., & Olsina, L. (2019). Especificación del Modelo de Proceso para una Revisión Sistemática de Literatura. In *Proceedings of the XXII Iberoamerican Conference on Software Engineering, CIbSE 2019, La Habana, Cuba, April 22-26* (pp. 391–404).

Tabla 6-2: Publicaciones relevantes asociadas a cada objetivo y contribución.

Objetivos específicos (OE)	Contribuciones (C)	Publicaciones relacionadas
OE_1: dado que la estrategia a desarrollar debe estar semánticamente soportada por una ontología, se debe investigar cuáles son las ontologías existentes para el dominio de las pruebas de software.	C_1: Una especificación de proceso para revisiones/mapeos sistemáticos de literatura.	Olsina, L., Becker, P., Peppino, D., & **Tebes, G.** (2019). Specifying the Process Model for Systematic Reviews: An Augmented Proposal. *Journal of Software Engineering Research and Development*, *7*, 1–23. https://doi.org/10.5753/jserd.2019.460
	C_2: Una Revisión Sistemática de Literatura sobre ontologías de pruebas de software.	**Tebes, G.**, Peppino, D., Becker, P., Matturro, G., Solari, M., & Olsina, L. (2020). Analyzing and documenting the systematic review results of software testing ontologies. *Information and Software Technology*, *123*, 1–23. https://doi.org/10.1016/j.infsof.2020.106298
OE_2: Una vez conocido el estado del arte de las ontologías de pruebas de software (i.e., una vez alcanzado el OE_1), se debe adoptar/adaptar alguna existente si cumple con los requisitos establecidos. En caso de que ninguna ontología existente cumpla con los requisitos establecidos, se deberá desarrollar una nueva.	C_3: Una especificación de proceso para la metodología de investigación de *Design Science Research.*	**Tebes, G.**, Rivera, B., Becker, P., Papa, M. F., Peppino, D., & Olsina, L. (2020). Specifying the design science research process: an applied case of building a software testing ontology. In *Proceedings of the XXIII Iberoamerican Conference on Software Engineering, CIbSE 2020* (pp. 378–391). Curran Associates.
	C_4: Una nueva ontología de dominio de nivel superior para pruebas de software llamada TestTDO.	**Tebes, G.**, Olsina, L., Peppino, D., & Becker, P. (2021). Specifying and Analyzing a Software Testing Ontology at the Top-Domain Ontological Level. *Journal of Computer Science & Technology (JCS&T)*, *21*(2), 126–145. https://doi.org/10.24215/16666038.21.e12
OE_3: Haciendo uso de los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de proceso que considere actividades de *testing* de diseño, ejecución y análisis, y además especifique cuáles son sus entradas/salidas correspondientes. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.	C_5: Una especificación de proceso para la estrategia SaST haciendo uso del lenguaje SPEM, el cual considera perspectivas de proceso.	**Tebes, G.**, Peppino, D., Becker, P., & Olsina, L. (2021). A Situation-aware Scenario-based Testing Strategy. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)* (pp. 1–8). IEEE.
OE_4: Usando los conceptos de pruebas de software de la ontología establecida, se debe desarrollar una especificación de método que este basado en la técnica de diseño de pruebas basada en escenarios. Además, debe dar soporte a situaciones que consideren entidades de contexto. Notar que se necesita alcanzar previamente el OE_2 para poder lograr este objetivo.	C_6: Una especificación de método para la estrategia SaST, la cual es una plantilla que captura información relacionada a la situación/escenario de prueba, entre otras cosas.	
OE_5: Una vez confeccionada la estrategia integrada, se debe validar la misma para inicialmente evaluar su utilidad. Notar que se deben haber alcanzado	C_7: la ilustración de la aplicación de la estrategia en un contexto real solo para el diseño de los casos de pruebas y escenarios.	Eleicegui, M. (2022). *Aplicación práctica de SaST, una estrategia de prueba basada en escenarios y consciente del contexto. Tesina de Ingeniería en Sistemas*. Facultad de Ingeniería, Universidad Nacional de La Pampa. https://repo.unlpam.edu.ar/handle/unlpam/8285

primero los objetivos OE_3 y OE_4 para poder lograr este objetivo.	C_8: la presentación de otra aplicación de la estrategia, también en un contexto real, la cual incluyo el diseño, implementación, automatización y ejecución de las pruebas.	Fernandez Reale, L. C. A. (2023). *Diseño y automatización de pruebas de sistema utilizando una estrategia de pruebas de software basada en escenarios y consciente de la situación. Tesina de Ingeniería en Sistemas*. Facultad de Ingeniería, Universidad Nacional de La Pampa. https://repo.unlpam.edu.ar/handle/unlpam/8325

Por otra parte, para la RSL sobre ontologías de pruebas de software (C_2), primero se realizó una prueba piloto la cual se documentó en: **Tebes, G.**, Peppino, D., Dameno, J., Becker, P., & Olsina, L. (2018). Diseñando una Revisión Sistemática de Literatura sobre Ontologías de Testing de Software. In *6to Congreso Nacional de Ingeniería en Informática/Sistemas De Información (CoNaIISI 2018)* (pp. 1-13). Luego, se llevó a cabo la RSL en su totalidad y se publicó inicialmente en: **Tebes, G.**, Peppino, D., Becker, P., Matturro, G., Solari, M., & Olsina, L. (2019). A Systematic Review on Software Testing Ontologies. In *Quality of Information and Communications Technology: 12th International Conference, QUATIC 2019, Ciudad Real, Spain, September 11–13, 2019, Proceedings 12* (pp. 144-160). Springer International Publishing.

Con respecto a la cuarta contribución (C_4: Una nueva ontología de dominio de nivel superior para pruebas de software llamada TestTDO), inicialmente fue publicada en: **Tebes, G.**, Olsina, L., Peppino, D., & Becker, P. (2020). TestTDO: A Top-Domain Software Testing Ontology. In *Proceedings of the XXIII Iberoamerican Conference on Software Engineering, CIbSE 2020* (pp. 364-377). Curran Associates. Además, aspectos de verificacion y validación de TestTDO fueron documentados en: Olsina, L., **Tebes, G.**, Peppino, D., & Becker, P. (2020). Approaches used to Verify and Validate a Software Testing Ontology as an Artifact. In *2020 IEEE Congreso Bienal de Argentina (ARGENCON)* (pp. 1-8). IEEE. A su vez, como TestTDO esta principalmente enriquecida con conceptos de la ontología ProcessCO, tambien es importante mencionar otro trabajo relacionado publicado en: Becker, P., Papa, M. F., **Tebes, G.**, & Olsina, L. (2022). Discussing the applicability of a process core ontology and aspects of its internal quality. *Software Quality Journal*, *30*(4), 1003-1038.

Finalmente, una primera aplicación de la estrategia SaST fue publicada en: Peppino, D., **Tebes, G.**, Becker, P., & Olsina, L. (2020). Designing Context-Aware Test Cases for Particular Situations. In *Congreso Nacional de Ingeniería Informática/Sistemas de Información (CoNaIISI)* (pp. 49-62). Tambien es importante aclarar en este punto que, aunque el autor de esta tesis doctoral no pudo participar como director o codirector de las tesinas de grado de Ingeniería en Sistemas mencionadas en la Tabla 6-2 (ver C_7 y C_8) por una cuestión de los requisitos necesarios para poder ocupar ese rol, participó activamente dando soporte a los estudiantes en el uso de la estrategia SaST y controlando su ejecución con el objetivo de retroalimentar y validar dicha estrategia.

6.4 Trabajos Futuros

A partir de este trabajo son varias las líneas de investigación en las que se pretende avanzar. Por un lado, se pretende seguir contribuyendo a la familia de estrategias de *testing* construyendo nuevas estrategias para diferentes propósitos y que consideren e integren las tres capacidades ya mencionadas a lo largo de este trabajo, a saber, una especificación de proceso, una especificación de métodos, y una base conceptual robusta como una ontología. La motivación principal de esta línea de avance futuro es que es muy probable que SaST sea la primera estrategia integrada de pruebas de software que considera una ontología de *testing* como base conceptual. Por lo tanto, esta familia de

estrategias integradas de *testing* por ahora solo cuenta con una estrategia y podría seguir creciendo incorporando nuevas de ellas.

Relacionado con el trabajo futuro anterior, si se desarrollaran nuevas estrategias integradas de *testing* más específicas y dado que la ontología TestTDO se encuentra a nivel de dominio superior, se podrían adoptar/adaptar o desarrollar nuevas ontologías de pruebas de software a nivel de dominio inferior para poder cubrir conceptos más específicos. Estas otras ontologías de más bajo nivel se podrían enriquecer con la semántica de dominio brindada por los conceptos de alto nivel de TestTDO y, a su vez, se podrían integrar a la arquitectura ontológica FCD-OntoArch. Por ejemplo, se podría analizar la posibilidad de adaptar e integrar la ontología denominada PTOntology (*Performance Testing*) (Freitas & Vieira, 2014) que cubre conceptos de pruebas no funcionales para pruebas de rendimiento. Usando esta ontología como base conceptual, se podría construir una nueva estrategia integrada de *testing* para llevar a cabo pruebas de rendimiento que tenga especificaciones de proceso y método particulares.

En este trabajo se realizó una validación inicial de la estrategia SaST al ser aplicada en dos empresas por dos estudiantes de grado para sus proyectos finales de ingeniería. Con la idea de seguir validando y mejorando esta estrategia se pretende aplicarla en otro contexto real para proyectos de software del dominio de las transacciones financieras.

REFERENCIAS

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a Better Understanding of Context and Context-Awareness. In *Handheld and Ubiquitous Computing: First International Symposium, HUC'99 Karlsruhe, Germany, September 27–29, 1999 Proceedings 1* (pp. 304–307). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48157-5_29

Amalfitano, D., Matalonga, S., Doreste, A. C. S., Fasolino, A. R., & Travassos, G. H. (2019). *A Rapid Review on Testing of Context-Aware Contemporary Software Systems. Technical Report, RT-ES-760, Systems Engineering and Computer Science Department*.

Arnicans, G., Romans, D., & Straujums, U. (2013). Semi-automatic generation of a software testing lightweight ontology from a glossary based on the ONTO6 methodology. *Frontiers in Artificial Intelligence and Applications*, *249*, 263–276.

Asman, A., & Srikanth, R. M. (2015). *A Top Domain Ontology For Software Testing. Master Thesis*. Jönköping University.

Bai, X., Lee, S., Tsai, W. T., & Chen, Y. (2008). Ontology-based test modeling and partition testing of web services. In *2008 IEEE International Conference on Web Services* (pp. 465–472). IEEE. https://doi.org/10.1109/ICWS.2008.111

Barbosa, E. F., Nakagawa, E. Y., Riekstin, A. C., & Maldonado, J. C. (2008). Ontology-based development of testing related tools. *20th International Conference on Software Engineering and Knowledge Engineering, SEKE 2008*, 697–702.

Becker, P., Lew, P., & Olsina, L. (2012). Specifying Process Views for a Measurement, Evaluation, and Improvement Strategy. *Advances in Software Engineering*, *2012*. https://doi.org/10.1155/2012/949746

Becker, P., Papa, F., & Olsina, L. (2015). Process Ontology Specification for Enhancing the Process Compliance of a Measurement and Evaluation Strategy. *CLEI Electronic Journal*, *18*(1), 1–26. https://doi.org/10.19153/cleiej.18.1.2

Becker, P., Papa, M. F., Tebes, G., & Olsina, L. (2022). Discussing the applicability of a process core ontology and aspects of its internal quality. *Software Quality Journal*, *30*(4), 1003–1038. https://doi.org/10.1007/s11219-022-09592-3

Becker, P., Tebes, G., Peppino, D., & Olsina, L. (2019). Applying an Improving Strategy that embeds Functional and Non-Functional Requirements Concepts. *Journal of Computer Science and Technology*, *19*(2), 153–175. https://doi.org/10.24215/16666038.19.e15

Benjamins, V. R., & Gomez-Perez, A. (2000). Knowledge-System Technology: Ontologies and Problem-Solving Methods. *Department of Social Science Informatics, University of Amsterdam, The Netherlands*.

Bezerra, D., Costa, A., & Okada, K. (2009). SwTOI (Software Test Ontology Integrated) and its application in Linux test. In *International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science* (Vol. 460, pp. 25–36). Amsterdam, The Netherlands: Elsevier. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84887054589&partnerID=40&md5=72b9d6f38c21b1ef553f502db354ea84

Bhattacharjee, V., Suri, D., & Mahanti, P. (2007). Software testing: A graph theoretic approach. *International Journal of Information and Communication Technology*, *1*(1), 14–25. https://doi.org/10.1504/IJICT.2007.013274

Biolchini, J., Mian, P. G., Natali, A. C. C., & Travassos, G. H. (2005). Systematic review in software engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, *679*(05), 45.

Borst, W. N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse. PhD Thesis*. Centre for Telematics and Information Technology (CTIT).

Bourque, P., & Fairley, R. E. (2014). *SWEBOK: guide to the software engineering body of knowledge*. IEEE Computer Society.

Brank, J., Grobelnik, M., & Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)* (pp. 166–170). Slovenia: Citeseer Ljubljana.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571–583. https://doi.org/https://doi.org/10.1016/j.jss.2006.07.009

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, *8*, 203–236. https://doi.org/10.1023/B:AGNT.0000018806.20944.ef

Bürger, T., & Simperl, E. (2008). Measuring the Benefits of Ontologies. In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops: OTM Confederated International Workshops and Posters, ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent+ QSI, ORM, PerSys, RDDS, SEMELS, and SWWS 2008, Monterrey, Mexico* (pp. 584–594). Springer Berlin Heidelberg.

Cai, L., Tong, W., Liu, Z., & Zhang, J. (2009). Test Case Reuse Based on Ontology. In *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing* (pp. 103–108). IEE. https://doi.org/10.1109/PRDC.2009.25

Chen, Y., & Jones, B. (2007). Software Testing Strategies for Model-Based Chassis Control Systems. *SAE Technical Papers*.

CMMI. (2010). *CMMI® for Development, Version 1.3, Carnegie Mellon University*. https://doi.org/10.1184/R1/6572342.v1

Cockburn, A. (2001). *Writing Effective Use Cases*. Pearson Education India.

Curtis, B., Kellner, M., & Over, J. (1992). Process Modeling. *Communications of the ACM*, *35*(9), 75–90. https://doi.org/10.1145/130994.130998

d'Aquin, M., & Gangemi, A. (2011). Is There Beauty in Ontologies? *Applied Ontology*, *6*(3), 165–175.

Daraio, C., Lenzerini, M., Leporelli, C., Naggar, P., Bonaccorsi, A., & Bartolucci, A. (2016). The advantages of an Ontology-Based Data Management approach: openness, interoperability and data quality. *Scientometrics*, *108*, 441–455. https://doi.org/10.1007/s11192-016-1913-6

de S. Campos Junior, H., de Paiva, C. A., Braga, R., Araújo, M. A. P., David, J. M. N., &

Campos, F. (2017). Regression Tests Provenance Data in the Continuous Software Engineering Context. In *Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing* (pp. 1–6). https://doi.org/10.1145/3128473.3128483

de Souza Doreste, A. C., & Travassos, G. H. (2020). Towards Supporting the Specification of Context-Aware Software System Test Cases. In *Proceedings of the XXIII Iberoamerican Conference on Software Engineering, CIbSE 2020* (pp. 356–363).

Dujmovic, J. J. (1996). A Method For Evaluation And Selection Of Complex Hardware And Software Systems. In *The 1996 22 nd International Conference for the Resource Management & Performance Evaluation of Enterprise Computing Systems, CMG. Part 1(of 2)* (pp. 368–378).

Duque-Ramos, A., Fernández-Breis, J. T., Stevens, R., & Aussenac-Gilles, N. (2011). OQuaRE: A square-based approach for evaluating the quality of ontologies. *Journal of Research and Practice in Information Technology*, *43*(2), 159–176.

Eleicegui, M. (2022). *Aplicación práctica de SaST, una estrategia de prueba basada en escenarios y consciente del contexto. Tesina de Ingeniería en Sistemas*. Facultad de Ingeniería, Universidad Nacional de La Pampa. https://repo.unlpam.edu.ar/handle/unlpam/8285

Fayen, E. G. (2007). Guidelines for the construction, format, and management of monolingual controlled vocabularies: A revision of ANSI/NISO Z39.19 for the 21st century. *Information Wissenschaft Und Praxis*, *58*(8), 445.

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. In *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series* (pp. 33–40).

Fernandez Reale, L. C. A. (2023). *Diseño y automatización de pruebas de sistema utilizando una estrategia de pruebas de software basada en escenarios y consciente de la situación. Tesina de Ingeniería en Sistemas*. Facultad de Ingeniería, Universidad Nacional de La Pampa. https://repo.unlpam.edu.ar/handle/unlpam/8325

Ferreira de Souza, É., de Almeida Falbo, R., & Vijaykumar, N. L. (2013). Ontologies in software testing: A Systematic Literature Review. In *VI Seminar on Ontology Research in Brazil* (p. 71).

Ferreira de Souza, É., de Almeida Falbo, R., & Vijaykumar, N. L. (2017). ROoST: Reference ontology on software testing. *Applied Ontology*, *12*(1), 59–90.

Freitas, A., & Vieira, R. (2014). An ontology for guiding performance testing. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* (Vol. 1, pp. 400–407). IEEE. https://doi.org/10.1109/WI-IAT.2014.62

Garousi, V., & Mäntylä, M. V. (2016). A systematic literature review of literature reviews in software testing. *Information and Software Technology*, *80*, 195–216.

Geerts, G. L. (2011). A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems*, *12*(2), 142–151.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge

sharing. *International Journal of Human - Computer Studies*, *43*(5–6), 907–928. https://doi.org/10.1006/ijhc.1995.1081

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

Haasjes, R. E. Y. (2019). *Metamodel transformations between UML and OWL, Master Thesis*. University of Twente.

Harmse, H. F., Britz, K., Gerber, A., & Moodley, D. (2014). Scenario testing using formal ontologies. *CEUR Workshop Proceedings*, *1301*. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84916230683&partnerID=40&md5=729d79762557cd6ad4bf631688206579

Hevner, A., & Chatterjee, S. (2010). *Design research in information systems theory and practice*. Springer.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, *19*(2), 4.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly: Management Information Systems*, *28*(1), 75–105. https://doi.org/10.2307/25148625

IEEE. (1998). *829-1998 IEEE Standard for Software Test Documentation*.

IEEE. (2008a). *1028–2008 IEEE standard for software reviews and audits*.

IEEE. (2008b). *829-2008 IEEE Standard for Software and System Test Documentation*.

Irshad, M., Petersen, K., & Poulding, S. (2018). A systematic literature review of software requirements reuse approaches. *Information and Software Technology*, *93*, 223–245.

ISO. (2001). ISO/IEC 9126-1: Software engineering-product quality-part 1: Quality model. *International Organization for Standardization*.

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

ISO. (2013b). ISO/IEC/IEEE 29119-2: Software and systems engineering - Software Testing – Part 2: Test processes. *International Organization for Standardization*.

ISO. (2013c). ISO/IEC/IEEE 29119-3: Software and systems engineering – Software Testing – Part 3: Test documentation. *International Organization for Standardization*.

ISO. (2015). ISO/IEC/IEEE 29119-4: Software and systems engineering - Software Testing – Part 4: Test techniques. *International Organization for Standardization*.

ISO. (2016). ISO/IEC/IEEE 29119-5: Software and systems engineering – Software Testing – Part 5: Keyword-Driven Testing. *International Organization for Standardization*.

ISTQB. (2018). *Certified Tester – Foundation Level Syllabus, version 3.1*. https://www.istqb.org/downloads/category/2-foundation-level-documents.html

ISTQB. (2021a). *Certified Tester – Advanced Level Technical Test Analyst (CTAL-TTA) Syllabus, version 4.0*. https://www.istqb.org/downloads/category/2-foundation-level-documents.html

ISTQB. (2021b). *Standard Glossary of Terms used in Software Testing, version 3.5*. https://www.istqb.org/

ISTQB. (2022). *Certifying Software Testers Worldwide*. https://www.istqb.org/

Kitchenham, B. (2004a). Procedures for Performing Systematic Reviews. *Keele, UK, Keele University*, *33*(2004), 1–26.

Kitchenham, B. (2004b). Procedures for Undertaking Systematic Reviews: Joint technical report. *Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd.(0400011T. 1)*.

Kitchenham, B. A., Budgen, D., & Brereton, O. P. (2010). The value of mapping studies--A participant-observer case study. In *14th international conference on evaluation and assessment in software engineering (ease)* (pp. 1–9).

Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). *Evidence-based software engineering and systematic reviews* (Vol. 4). CRC press.

Kitchenham, B., & Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and Software Technology*, *55*(12), 2049–2075.

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*.

Kitchenham, B., Pretorius, R., Budgen, D., Brereton, P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering – A tertiary study. *Information and Software Technology*, *52*(8), 792–805.

Lam, R. W., & Kennedy, S. H. (2005). Using metaanalysis to evaluate evidence: practical tips and traps. *The Canadian Journal of Psychiatry*, *50*(3), 167–174.

Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2007). IPOG: A General Strategy for T-Way Software Testing. In *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)* (pp. 549–556). IEEE. https://doi.org/10.1109/ECBS.2007.47

Lozano-Tello, A., & Gómez-Pérez, A. (2004). ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, *15*(2), 1–18. https://doi.org/10.4018/jdm.2004040101

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266.

Marshall, C., & Brereton, P. (2013). Tools to support systematic literature reviews in software engineering: A mapping study. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 296–299). IEEE.

McKay, J., & Marshall, P. (2005). A review of design science in information systems. In *16th Australasian Conference on Information Systems* (pp. 1–11).

Meriem, A., & Abdelaziz, M. (2019). Combining Model-Based Testing and Failure Modes and Effects Analysis for Test Case Prioritization: A Software Testing Approach. *Journal of Computer Science*, *15*(4), 435–449. https://doi.org/10.3844/jcssp.2019.435.449

Merrell, E., Kelly, R. M., Kasmier, D., Smith, B., Brittain, M., Ankner, R., Maki, E.,

Heisey, C. W., & Bush, K. (2021). Benefits of Realist Ontologies to Systems Engineering. In *8th International Workshop on Ontologies and Conceptual Modelling (OntoCom)* (pp. 1–10).

Milton, S. K., Keen, C. D., & Kurnia, S. (2010). Understanding the benefits of ontology use for Australian industry: A conceptual study. *ACIS 2010 Proceedings - 21st Australasian Conference on Information Systems*.

Mirza, A. M., & Khan, M. N. A. (2018). An Automated Functional Testing Framework for Context-Aware Applications. *IEEE Access*, *6*, 46568–46583. https://doi.org/10.1109/ACCESS.2018.2865213

Myers, G. J., Badgett, T., & Sandler, C. (2012). *The Art of Software Testing* (3rd ed.). John Wiley & Sons, Inc. https://doi.org/10.1002/9781119202486

Napoleão, B., Felizardo, K. R., de Souza, É. F., & Vijaykumar, N. L. (2017). Practical similarities and differences between Systematic Literature Reviews and Systematic Mappings: a tertiary study. In *SEKE* (Vol. 2017, pp. 85–90).

Nguyen, D. C., Perini, A., & Tonella, P. (2008). A Goal-Oriented Software Testing Methodology. In *Agent-Oriented Software Engineering VIII: 8th International Workshop, AOSE 2007, Honolulu, HI, USA, May 14, 2007, Revised Selected Papers* (pp. 58–72). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-79488-2_5

Oberle, D. (2014). How ontologies benefit enterprise applications. *Semantic Web*, *5*(6), 473–491. https://doi.org/10.3233/SW-130114

Offermann, P., Levina, O., Schönherr, M., & Bub, U. (2009). Outline of a design science research process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology* (pp. 1–11).

Olsina, L. (2020). Analyzing the Usefulness of ThingFO as a Foundational Ontology for Sciences. In *proceedings of Argentine Symposium on Software Engineering, ASSE'20, 49 JAIIO* (pp. 172–191).

Olsina, L. (2021). Applicability of a Foundational Ontology to Semantically Enrich the Core and Domain Ontologies. In *13th International Conference on Knowledge Engineering and Ontology Development (KEOD), IC3K* (pp. 111–119).

Olsina, L. (2023). The Foundational Ontology ThingFO: Architectural Aspects, Concepts, and Applicability. In *Fred, A., Aveiro, D., Dietz, J., Bernardino, J., Masciari, E., Filipe, J. (eds.) Knowledge Discovery, Knowledge Engineering and Knowledge Management. IC3K 2021. Communications in Computer and Information Science* (Vol. 1718, pp. 73–99). Springer.

Olsina, L., & Becker, P. (2017). Family of strategies for different evaluation purposes. In *XX CIbSE' 17* (pp. 221–234).

Olsina, L., & Becker, P. (2018). Linking Business and Information Need Goals with Functional and Non-functional Requirements. In *Proceedings of the XXI Conferencia Iberoamericana en Software Engineering (CIbSE'18)* (pp. 381–394).

Olsina, L., Becker, P., Peppino, D., & Tebes, G. (2019). Specifying the Process Model for Systematic Reviews: An Augmented Proposal. *Journal of Software Engineering Research and Development*, *7*, 1–23. https://doi.org/10.5753/jserd.2019.460

Olsina, L., Dieser, A., & Covella, G. (2014). Metrics and indicators as key organizational

assets for ICT security assessment. In *Emerging Trends in ICT Security* (pp. 25–44). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-411474-6.00002-5

Olsina, L., Papa, F., & Becker, P. (2023). NFRsTDO v1.2's Terms, Properties and Relationships -- A Top-Domain Non-Functional Requirements Ontology. *ArXiv Preprint*. https://arxiv.org/abs/2302.01096

Olsina, L., Papa, F., & Molina, H. (2008). How to Measure and Evaluate Web Applications in a Consistent Way. In *Springer Book: Web Engineering: Modeling and Implementing Web Applications* (pp. 385–420). https://doi.org/10.1007/978-1-84628-923-1_13

Olsina, L., Tebes, G., & Becker, P. (2021). *SituationCO v1.2's Terms, Properties and Relationships - A Core Ontology for Particular and Generic Situations. Preprint in Research Gate*. https://doi.org/10.13140/RG.2.2.23646.56644

OMG. (2008). Software & Systems Process Engineering Meta-Model Specification. *OMG Std., Rev, 2*, 18–71.

OMG. (2017). *Unified Modeling Language (UML) Specification, Version 2.5.1*.

OMG. (2019). *UML Testing Profile 2 (UTP 2), Version 2.1*. https://www.omg.org/spec/UTP2

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

Pei, H., Yin, B., Xie, M., & Cai, K.-Y. (2017). A cloud-based dynamic random software testing strategy. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 509–513). IEEE. https://doi.org/10.1109/IEEM.2017.8289943

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1–18.

Pfeffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006). The design science research process: A model for producing and presenting information systems research. In *First International Conference on Design Science Research in Information Systems and Technology Proceedings. Claremont, CA* (pp. 83–106).

Portela, C., Vasconcelos, A., Sinimbú, A., Silva, E., Ronny, M., Lira, W., Oliveira, S., & others. (2012). A comparative analysis between BPMN and SPEM modeling standards in the software processes context. *Journal of Software Engineering and Applications*, *5*(5), 330–339. http://dx.doi.org/10.4236/jsea.2012.55039

Ramler, R., Felderer, M., & Leitner, M. (2017). A Lightweight Approach for Estimating Probability in Risk-Based Software Testing. In *Risk Assessment and Risk-Driven Quality Assurance: 4th International Workshop, RISK 2016, Held in Conjunction with ICTSS 2016, Graz, Austria, October 18, 2016, Revised Selected Papers 4* (pp. 115–128). Springer International Publishing. https://doi.org/10.1007/978-3-319-57858-3_9

Rivera, B., Becker, P., Papa, M. F., & Olsina, L. (2016). A Holistic Quality Evaluation,

Selection and Improvement Approach driven by Multilevel Goals and Strategies. *CLEI Electronic Journal*, *19*(3), 54–104.

Russell, N., der Aalst, W., Ter Hofstede, A., & Wohed, P. (2006). On the suitability of UML 2.0 activity diagrams for business process modelling. *Conceptual Modelling 2006: Proceedings of APCCM2006*, 95–104.

Sackett, D. L., Rosenberg, W. M. C., Gray, J. A. M., Haynes, R. B., & Richardson, W. S. (1996). Evidence based medicine: What it is and what it isn't. *BMJ (Clinical Research Ed.)*, *312*(7023), 71–72. https://doi.org/10.1136/bmj.313.7050.170c

Sapna, P. G., & Mohanty, H. (2011). An Ontology Based Approach for Test Scenario Management. In *Information Intelligence, Systems, Technology and Management: 5th International Conference, ICISTM 2011, Gurgaon, India, March 10-12, 2011. Proceedings 5* (pp. 91–100). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19423-8_10

Schneider, K. (2009). *Experience and Knowledge Management in Software Engineering* (Vol. 235). Berlin: Springer. https://doi.org/10.1007/978-3-540-95880-2

Sepúlveda, S., Cravero, A., & Cachero, C. (2016). Requirements modeling languages for software product lines: A systematic literature review. *Information and Software Technology*, *69*, 16–36. https://doi.org/https://doi.org/10.1016/j.infsof.2015.08.007

Sommerville, I. (2011). *Software Engineering* (9th ed.). Pearson Education India.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, *25*(1), 161–197. https://doi.org/https://doi.org/10.1016/S0169-023X(97)00056-6

Swartout, W., & Tate, A. (1999). Guest Editors' Introduction: Ontologies. *IEEE Intelligent Systems*, *14*(1), 18–19. https://doi.org/10.1109/MIS.1999.747901

Tahir, T., Rasool, G., & Gencel, C. (2016). A systematic literature review on software measurement programs. *Information and Software Technology*, *73*, 101–121. https://doi.org/https://doi.org/10.1016/j.infsof.2016.01.014

Tebes, G., Olsina, L., Peppino, D., & Becker, P. (2020a). FRsTDO v1.1's Terms, Properties and Relationships -- A Top-Domain Functional Requirements Ontology. *Preprint in Research Gate*. https://doi.org/10.13140/RG.2.2.31659.26400

Tebes, G., Olsina, L., Peppino, D., & Becker, P. (2020b). TestTDO: A Top-Domain Software Testing Ontology. In *Proceedings of the XXIII Iberoamerican Conference on Software Engineering, CIbSE 2020* (pp. 364–377). Curran Associates.

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

Tebes, G., Peppino, D., Becker, P., Matturro, G., Solari, M., & Olsina, L. (2019). A Systematic Review on Software Testing Ontologies. In *Quality of Information and Communications Technology: 12th International Conference, QUATIC 2019, Ciudad Real, Spain, September 11–13, 2019, Proceedings 12* (pp. 144–160). Springer International Publishing.

Tebes, G., Peppino, D., Becker, P., Matturro, G., Solari, M., & Olsina, L. (2020). Analyzing and documenting the systematic review results of software testing

ontologies. *Information and Software Technology*, *123*, 1–23. https://doi.org/10.1016/j.infsof.2020.106298

Tebes, G., Peppino, D., Becker, P., & Olsina, L. (2021). A Situation-aware Scenario-based Testing Strategy. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)* (pp. 1–8). IEEE. https://doi.org/10.1109/SCCC54552.2021.9650399

Tebes, G., Peppino, D., Becker, P., Papa, M. F., Belen, R., & Olsina, L. (2018). Family of Evaluation Strategies: A Practical Case for Comparing and Adopting Strengths. *Journal of Computer Science & Technology*, *18*, 48–60. https://doi.org/10.24215/16666038.18.e06

Tebes, G., Peppino, D., Dameno, J., Becker, P., & Olsina, L. (2018). Diseñando una Revisión Sistemática de Literatura sobre Ontologías de Testing de Software. In *6to CONGRESO NACIONAL DE INGENIERÍA EN INFORMÁTICA/SISTEMAS DE INFORMACIÓN (CoNaIISI 2018)* (pp. 1–13).

Tebes, G., Rivera, B., Becker, P., Papa, M. F., Peppino, D., & Olsina, L. (2020). Specifying the design science research process: an applied case of building a software testing ontology. In *Proceedings of the XXIII Iberoamerican Conference on Software Engineering, CIbSE 2020* (pp. 378–391). Curran Associates.

Thuan, N. H., Drechsler, A., & Antunes, P. (2019). Construction of design science research questions. *Communications of the Association for Information Systems*, *44*(1), 20.

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

Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2016). Agile, Web Engineering and Capability Maturity Model Integration: A systematic literature review. *Information and Software Technology*, *71*, 92–107. https://doi.org/https://doi.org/10.1016/j.infsof.2015.11.002

Traoré, I. (2003). A Transition-Based Strategy for Object-Oriented Software Testing. In *Proceedings of the 2003 ACM Symposium on Applied Computing* (pp. 1055–1062). Association for Computing Machinery. https://doi.org/10.1145/952532.952739

Vaishnavi, V., & Kuechler, W. (2004). *Design research in information systems*.

Vasanthapriyan, S., Tian, J., & Xiang, J. (2017). An ontology-based knowledge framework for software testing. In *Knowledge and Systems Sciences. KSS 2017. Communications in Computer and Information Science* (Vol. 780, pp. 212–226). https://doi.org/10.1007/978-981-10-6989-5_18

Vasanthapriyan, S., Tian, J., Zhao, D., Xiong, S., & Xiang, J. (2017). An Ontology-Based Knowledge Sharing Portal for Software Testing. In *2017 IEEE international conference on software quality, reliability and security companion (QRS-C)* (pp. 472–479). IEEE. https://doi.org/10.1109/QRS-C.2017.82

Vo, M. H. L., & Hoang, Q. (2020). Transformation of UML class diagram into OWL Ontology. *Journal of Information and Telecommunication*, *4*(1), 1–16. https://doi.org/10.1080/24751839.2019.1686681

Vrandečić, D., & Gängemi, A. (2006). Unit tests for ontologies. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. OTM 2006. Lecture*

Notes in Computer Science (Vol. 4278, pp. 1012–1020). Springer Berlin Heidelberg. https://doi.org/10.1007/11915072_2

Wang, Z., Elbaum, S., & Rosenblum, D. S. (2007). Automated Generation of Context-Aware Tests. In *29th International Conference on Software Engineering (ICSE'07)* (pp. 406–415). IEEE. https://doi.org/10.1109/ICSE.2007.18

White, S. A., & others. (2004). Process modeling notations and workflow patterns. *Workflow Handbook*, *2004*, 265–294.

Wieringa, R. (2014). Design Science Methodology for Information Systems and Software Engineering. *Springer Berlin Heidelberg*.

Yu, L., Tsai, W.-T., & Perrone, G. (2016). Testing Context-Aware Applications Based on Bigraphical Modeling. *IEEE Transactions on Reliability*, *65*(3), 1584–1611. https://doi.org/10.1109/TR.2016.2575444

Zhu, H., & Huo, Q. (2005). Developing Software Testing Ontology in UML for a Software Growth Environment of Web-Based Applications. In *Software Evolution with UML and XML* (pp. 263–295). IGI Global. https://doi.org/10.4018/978-1-59140-462-0.ch009

APÉNDICE A: VISTA ORGANIZACIONAL PARA EL PROCESO DE RSL PROPUESTO

Teniendo en cuenta que una RSL requiere mucho esfuerzo y que difícilmente puede ser abordado por una sola persona, por lo general, varios investigadores están involucrados y desempeñan diferentes roles. Las RSLs de mayor calidad deberán contar con aportes de expertos en el tema a revisar, en los diferentes métodos de búsqueda y recolección, en métodos de análisis cualitativo y cuantitativo, entre muchos otros aspectos. Por lo tanto, en la Figura A-1, se muestra la vista organizacional con los diferentes roles involucrados en un proceso de RSL.

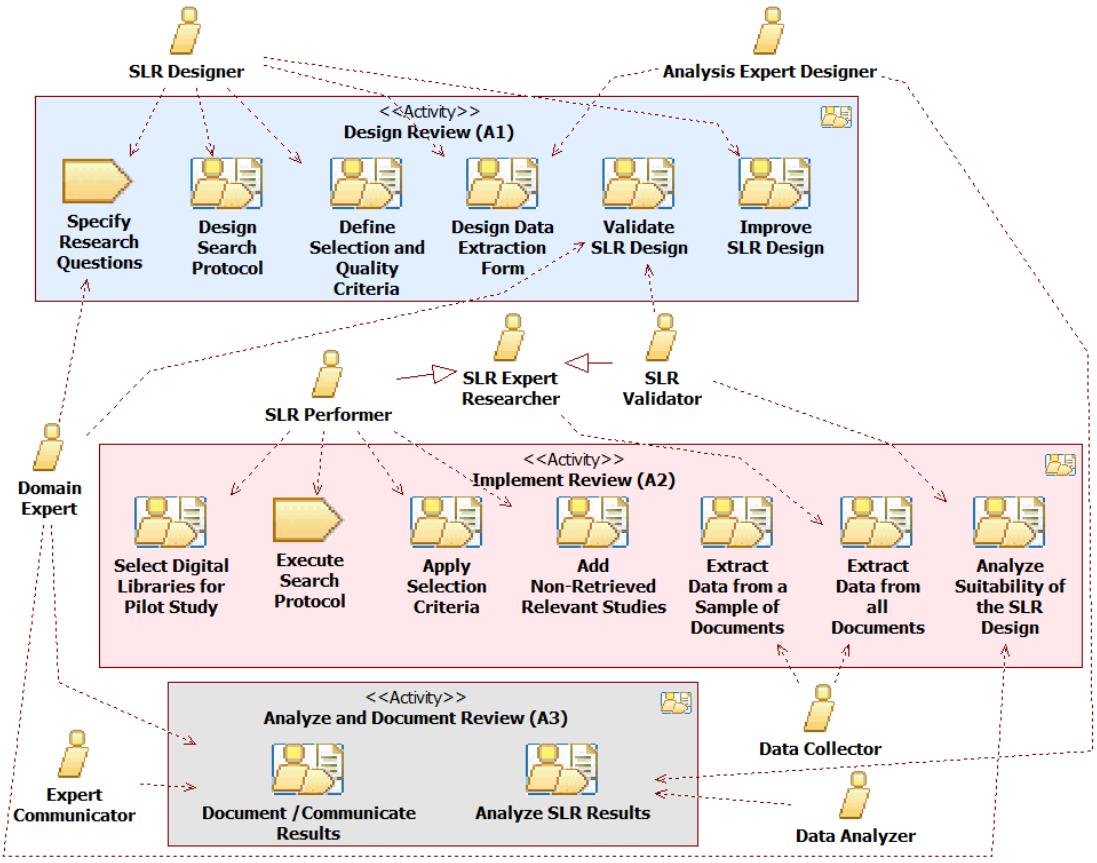


Figura A-1: Vista organizacional del proceso de RSL propuesto. Figura tomada de (Olsina et al., 2019).

Entre los roles involucrados, A1 incluye al Diseñador de la RSL cuyo agente debe poder llevar a cabo estrategias de búsqueda e identificar las fuentes o bibliotecas apropiadas. Además, este rol y el Diseñador Experto en Análisis son necesarios para diseñar el formulario de extracción de datos, así como para la definición de potenciales métodos de análisis. El rol de Validador de la RSL debería ser jugado por un agente con experiencia en la realización de revisiones sistemáticas. Con respecto al rol de Experto en el Dominio, este debería ser desempeñado por un agente destinado a validar el protocolo y aclarar cuestiones relacionadas con el tema de investigación.

La Tabla A-1 describe las responsabilidades y/o capacidades requeridas para los diferentes roles. Tenga en cuenta que un agente puede desempeñar diferentes roles y, a su vez, un rol puede ser desempeñado por uno o más agentes (o incluso por un equipo). Por ejemplo, en una RSL particular, varios investigadores desempeñan con frecuencia la función de Recolector de Datos, ya que las subactividades Extraer Datos de una Muestra

de Documentos y Extraer Datos de Todos los Documentos consumen mucho tiempo y requieren mucho esfuerzo.

Tabla A-1: Definiciones de los roles del proceso de RSL propuesto. Tabla tomada y ligeramente adaptada de (Olsina et al., 2019).

Rol	Definición (en términos de responsabilidades/capacidades)
Diseñador Experto en Análisis (*Analysis Expert Designer*)	Un investigador responsable de identificar los métodos y técnicas adecuadas que se utilizarán para el análisis de datos cualitativos/cuantitativos. El agente que desempeñe este rol también debe ser capaz de manejar técnicas de documentación y visualización.
Analizador de Datos (*Data Analyzer*)	Responsable de realizar el análisis de datos.
Recolector de Datos (*Data Collector*)	Responsable de extraer datos de estudios primarios o secundarios.
Experto en el Dominio (*Domain Expert*)	Un investigador o profesional con conocimientos, habilidades y experiencia en un tema particular o dominio de interés.
Comunicador Experto (*Expert Communicator*)	Investigador con habilidades de retórica y oratoria que comunica los resultados de una RSL a una comunidad/audiencia prevista.
Diseñador de la RSL (*SLR Designer*)	Investigador con conocimientos y habilidades para diseñar y especificar protocolos de RSLs.
Investigador Experto de la RSL (*SLR Expert Researcher*)	Un investigador con conocimiento y experiencia en la realización de RSLs.
Realizador de la RSL (*SLR Performer*)	Un investigador con conocimientos y habilidades para la recuperación de documentos. El agente que desempeñe este rol debe ser experto en el uso de motores de búsqueda y, métodos y técnicas de recuperación de documentos.
Validador de la RSL (*SLR Validator*)	Un investigador con experiencia en RSLs para verificar la adecuación y validez de un diseño de RSL.

APÉNDICE B: TRABAJOS RELACIONADOS A DSR

En (Offermann et al., 2009) los autores presentan un proceso para DSR estructurado en tres fases principales, a saber: 1) identificación del problema; 2) diseño de la solución; y 3) evaluación, que pueden interactuar entre ellas. Cada fase implica ciertos pasos a seguir. La fase de identificación del problema se divide en: identificar el problema, revisar la literatura, realizar entrevistas con expertos y una evaluación de la relevancia del problema. Cuando el problema se identifica, se realiza una evaluación de su relevancia para decidir si será abordado como un problema a resolver con DSR. Esto implica determinar una hipótesis, la cual se irá ajustando conforme el proceso de investigación avance. La evaluación de la relevancia se lleva a cabo interrogando a distintos investigadores con el fin de determinar si acuerdan con las hipótesis planteadas para el problema detectado.

La fase de diseño de la solución se divide en los siguientes pasos: diseño del artefacto e investigación de la literatura. Luego de que el problema se identificó y su relevancia es admitida para ser abordada mediante DSR, es preciso determinar una solución a dicho problema en la forma de un artefacto.

Finalmente, la fase de evaluación implica refinar la hipótesis (si es necesario) y realizar sobre ella casos de estudio que permitan ir ajustando la misma. Adicionalmente, sugieren utilizar la técnica de entrevistas con expertos para demostrar que hay interés general en la solución. La entrevista puede considerar la siguiente pregunta: "¿considera que el artefacto provee una solución viable para el problema?". Y podría también incluirse una pregunta respecto a la relevancia, aun cuando ésta ya fue analizada durante la etapa de identificación del problema.

Para concluir con el proceso de investigación, los resultados se sintetizan y se publican en tesis doctorales, revistas y/o eventos científicos. De esta manera, se espera una retroalimentación de la comunidad científica interesada sobre los resultados expuestos obtenidos. El trabajo en (Offermann et al., 2009) es relevante y las fases han sido consideradas para la construcción del proceso sugerido en este trabajo. Sin embargo, el proceso no está modelizado sino más bien presentado en forma de esquema, sin tener en cuenta productos de trabajos producidos y consumidos por las distintas fases. Además, solo considera la perspectiva de modelado de proceso de comportamiento.

Otros trabajos de interés son (A. R. Hevner, 2007; A. R. Hevner et al., 2004), en los cuales los autores presentan un marco de investigación que permite comprender el alcance de DSR para ser aplicado en investigaciones relacionadas a Sistemas de la Información. Además, contiene siete guías que ayudan a entender los requisitos para utilizar de manera efectiva DSR. Las guías son las siguientes: 1) el diseño como un artefacto; 2) relevancia del problema; 3) evaluación del diseño; 4) contribuciones de la investigación; 5) rigor de la investigación; 6) el diseño como un proceso de búsqueda; y 7) comunicación de la investigación. En síntesis, estas guías sostienen que DSR debe producir un artefacto viable que constituya una solución a un problema de negocio relevante. Para determinar la viabilidad del artefacto se debe evaluar su utilidad, calidad y eficacia por medio de métodos rigurosos de evaluación (casos de estudio, pruebas, simulaciones). Esto permitirá conocer las contribuciones en el dominio de investigación que el desarrollo del artefacto brindará a la comunidad en sí, y finalmente, contribuir a dicha comunidad por medio de la comunicación de los resultados. Al igual que sucede con (Offermann et al., 2009), en (A. R. Hevner, 2007; A. R. Hevner et al., 2004) tampoco se presenta un modelo

de proceso que indique formalmente las actividades y tareas a desarrollar, teniendo en consideración estas guías. Incluso, tampoco hace mención de los productos de trabajo que se consumen y producen a lo largo del proceso de DSR.

A su vez, en (A. R. Hevner, 2007), la aplicación del enfoque de DSR en Sistemas de la Información implica tres ciclos de investigación denominados ciclo de Relevancia, de Diseño y de Rigor, como se ilustran en la Figura B-1.

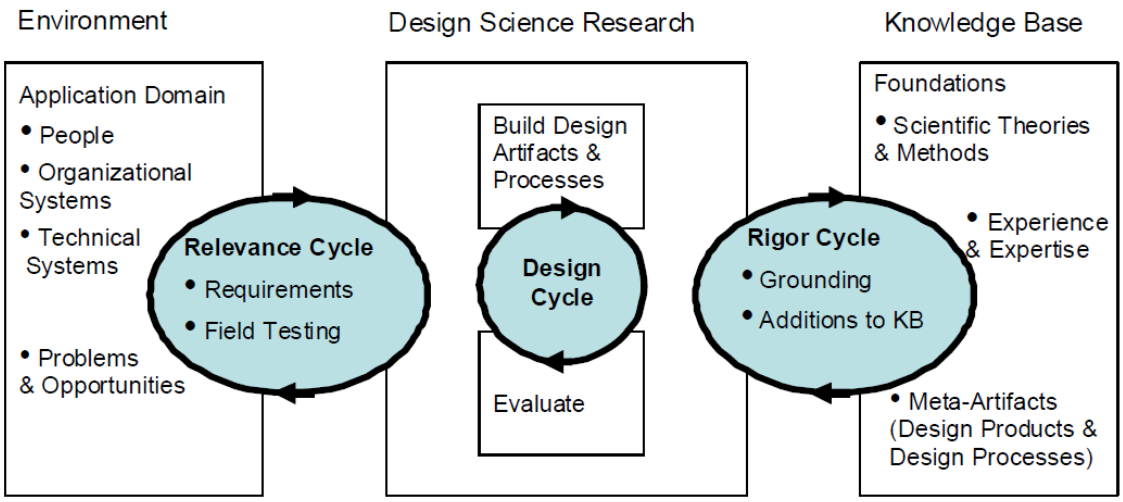


Figura B-1: Los tres ciclos en DSR. Figura tomada de (A. R. Hevner, 2007).

En el ciclo de Relevancia se identifica el dominio del problema, detectando problemas/metas organizacionales y oportunidades de mejora. Además, se llevan a cabo las investigaciones del estado del arte para determinar la relevancia de abordar la solución del problema mediante DSR. También se formulan las preguntas de investigación y los requisitos del artefacto que guiarán el diseño y construcción del mismo.

Por otra parte, en el ciclo de Diseño se itera entre las principales actividades de diseño y construcción del artefacto. A su vez, se identifican los procesos de diseño, métodos o heurísticas que se utilizarán para construir el artefacto, así como también las evaluaciones necesarias para evaluar el diseño. También se identifican mejoras en el diseño a través de la retroalimentación obtenida de las validaciones durante el ciclo de Relevancia.

Finalmente, en el ciclo de Rigor se asegura que el diseño esté basado en teorías científicas y métodos bien establecidos. En caso de ser un artefacto útil para resolver el problema en cuestión, se podrá incluir en la base de conocimiento para contribuir al acervo de la comunidad.

Los autores de (Pfeffers et al., 2006) presentan lo que denominan un modelo de proceso para DSR, pero no formalizado en un lenguaje de modelado sino simplemente con una figura de cuadros, el cual ilustra la secuencia de actividades (es decir, la vista de comportamiento) que proponen seguir para implementar el enfoque de investigación. El trabajo también es relevante y sirvió como material para formalizar el proceso que se propone en esta tesis doctoral. Las actividades planteadas en (Pfeffers et al., 2006) inician con la identificación del problema y la motivación, detectando la relevancia del mismo. A seguir, se identifican los objetivos de la solución. La solución es el artefacto que debe ser diseñado y construido. Posteriormente se demuestra su eficacia como solución al problema identificado. La demostración puede ser llevada a cabo por medio de casos de estudio, experimentación, simulación, prueba de conceptos, entre otros. Luego se evalúa y, de ser necesario, se vuelve a la etapa de diseño para realizar una nueva iteración.

Finalmente, concluyen con la actividad de comunicar los resultados para diseminar el conocimiento obtenido a la comunidad interesada.

En (Vaishnavi & Kuechler, 2004) se describe un modelo de proceso que se sigue en la aplicación de DSR. En términos generales, esta propuesta propone como inicio el descubrimiento del problema, generando como salida una propuesta del mismo. Luego se continúa con una fase denominada creativa, donde se detectan las funcionalidades y requisitos que deberá tener el artefacto para dar luego lugar a la etapa de desarrollo. El proceso culmina con la evaluación del artefacto de acuerdo a los requisitos establecidos y la exposición de los resultados dentro de la etapa de conclusiones. Este trabajo, al igual que los anteriores analizados, realiza más bien una descripción de las tareas enmarcadas como “pasos del proceso” (como los autores prefieren denominar), pero carece de una especificación formal. Es importante destacar que en (Vaishnavi & Kuechler, 2004) se tuvieron en cuenta las vistas de comportamiento y funcional. No obstante, dentro de la perspectiva funcional, solo se ilustran los artefactos producidos por las actividades, sin modelar las entradas de las mismas.

APÉNDICE C: MATRIZ DE VERIFICACIÓN ESTÁTICA PARA LAS CQS DE TESTTDO

Este apéndice muestra la Tabla C-1 que contiene la matriz de verificación estática completa que se produjo como resultado de la actividad Realizar Pruebas Estáticas aplicada a TestTDO. Recordar que esta actividad esta ilustrada en la Sección 4.3.3.3 y el objetivo era verificar los requisitos del alcance, es decir, las CQs.

Tabla C-1. Matriz de verificación estática del alcance de TestTDO. Los códigos que identifican los axiomas se corresponden con los presentados en la Sección 4.3.3.2.

Preguntas de Competencia (CQs)	**Términos, relaciones y *propiedades***	**Axiomas**
CQ1. ¿Cuáles son los productos de trabajo producidos por una actividad de Diseñar la Prueba?	Diseñar la Prueba es-una Actividad de Prueba Diseñar la Prueba produce Especificación de Prueba Caso de Prueba es-una Especificación de Prueba Lista de Verificación es-una Especificación de Prueba Conjunto de Casos de Prueba es-una Especificación de Prueba Diseñar la Prueba produce la ***especificación*** de un Procedimiento de Realización	A_XV, A_XVI
CQ2. ¿Cuáles son los productos de trabajo consumidos por una actividad de Diseñar la Prueba?	Diseñar la Prueba es-una Actividad de Prueba Diseñar la Prueba consume Base de Prueba	A_XI, A_XII, A_XV, A_XVI
CQ3. ¿Cuáles son los productos de trabajo producidos por una actividad de Realizar la Prueba?	Realizar la Prueba es-una Actividad de Prueba Realizar la Prueba produce Resultado de Prueba Resultado Real es-un Resultado de Prueba Incidente es-un Resultado de Prueba	A_VI, A_VII, A_VIII
CQ4. ¿Cuáles son los productos de trabajo consumidos por una actividad de Realizar la Prueba?	Realizar la Prueba es-una Actividad de Prueba Realizar la Prueba consume Especificación de Prueba Caso de Prueba es-una Especificación de Prueba Lista de Verificación es-una Especificación de Prueba Conjunto de Casos de Prueba es-una Especificación de Prueba	A_VII, A_XI, A_XII
CQ5. ¿Cuáles son los productos de trabajo producidos por una actividad de Analizar Resultados de Prueba?	Analizar Resultados de Prueba es-una Actividad de Prueba Analizar Resultados de Prueba produce Informe de Conclusión de Prueba	-
CQ6. ¿Cuáles son los productos de trabajo consumidos por una actividad de Analizar Resultados de Prueba?	Analizar Resultados de Prueba es-una Actividad de Prueba Analizar Resultados de Prueba consume Resultado de Prueba Resultado Real es-un Resultado de Prueba Incidente es-un Resultado de Prueba Analizar Resultados de Prueba consume Especificación de Prueba Caso de Prueba es-una Especificación de Prueba Lista de Verificación es-una Especificación de Prueba Conjunto de Casos de Prueba es-una Especificación de Prueba Analizar Resultados de Prueba toma en cuenta la ***declaración*** de una Necesidad de Información de Prueba	A_XI, A_XII
	Realizar Pruebas Dinámicas es-una actividad de Realizar la Prueba	A_XIII, A_XIV

CQ7. ¿Cuáles son los tipos de Actividades de Prueba dinámicas?	Realizar la Prueba es-una Actividad de Prueba Realizar Pruebas Dinámicas Funcionales es-una actividad de Realizar Pruebas Dinámicas Realizar Pruebas Dinámicas No Funcionales es-una actividad de Realizar Pruebas Dinámicas	
CQ8. ¿Cuál es el conjunto mínimo de Actividades de Prueba incluidas en un Proceso de Prueba?	Actividad de Prueba es-parte-de Proceso de Prueba Diseñar la Prueba es-una Actividad de Prueba Realizar la Prueba es-una Actividad de Prueba Analizar Resultados de Prueba es-una Actividad de Prueba	A_V
CQ9. ¿Cuáles son los tipos de actividades de Realizar la Prueba que consumen Casos de Prueba?	Realizar la Prueba es-una Actividad de Prueba Realizar Pruebas Estáticas es-una actividad de Realizar la Prueba Realizar Pruebas Dinámicas es-una actividad de Realizar la Prueba Realizar Pruebas Dinámicas Funcionales es-una actividad de Realizar Pruebas Dinámicas Realizar Pruebas Dinámicas No Funcionales es-una actividad de Realizar Pruebas Dinámicas Realizar la Prueba consume Especificación de Prueba Caso de Prueba es-una Especificación de Prueba	A_VIII
CQ10. ¿Cuáles son los tipos de actividades de Realizar la Prueba que consumen Listas de Verificación?	Realizar la Prueba es-una Actividad de Prueba Realizar Pruebas Estáticas es-una actividad de Realizar la Prueba Realizar Pruebas Dinámicas es-una actividad de Realizar la Prueba Realizar Pruebas Dinámicas Funcionales es-una actividad de Realizar Pruebas Dinámicas Realizar Pruebas Dinámicas No Funcionales es-una actividad de Realizar Pruebas Dinámicas Realizar la Prueba consume Especificación de Prueba Listas de Verificación es-una Especificación de Prueba	-
CQ11. ¿Cuáles son los tipos de entidades de prueba requeridas por un Proceso de Prueba como entrada?	Proceso de Prueba requiere como entrada Entidad Comprobable Proceso de Prueba requiere como entrada Entidad de Contexto de Prueba	A_IX, A_X
CQ12. ¿Cuáles son los tipos de Métodos de Prueba asignados a una Actividad de Prueba dinámica?	Realizar Pruebas Dinámicas es-una actividad de Realizar la Prueba Realizar la Prueba es-una Actividad de Prueba Método de Prueba Dinámica se asigna a una actividad de Realizar Pruebas Dinámicas Método de Prueba Dinámica es-un Método de Realización de Pruebas Método de Realización de Pruebas es-un Método de Pruebas	-
CQ13. ¿Cuáles son los tipos de Métodos de Prueba asignados a una Actividad de Prueba estática?	Realizar Pruebas Estáticas es-una actividad de Realizar la Prueba Realizar la Prueba es-una Actividad de Prueba Método de Prueba Estática se asigna a una actividad de Realizar Pruebas Estáticas Método de Prueba Estática es-un Método de Realización de Pruebas Método de Realización de Pruebas es-un Método de Pruebas	-
CQ14. ¿Cuáles son los tipos de Métodos de Prueba asignados a	Diseñar la Prueba es-una Actividad de Prueba Método de Diseño de Prueba se asigna a una actividad de Diseñar la Prueba	A_XV, A_XVI

una actividad de Diseñar la Prueba?	Método de Diseño de Prueba es-un Método de Pruebas	
	Método basado en la Especificación es-un Método de Diseño de Prueba	
	Método basado en la Estructura es-un Método de Diseño de Prueba	
	Método basado en la Experiencia es-un Método de Diseño de Prueba	
CQ15. ¿Cuáles son los tipos de Agentes de Prueba asignados a una Actividad de Prueba?	Agente de Prueba se asigna a una Actividad de Prueba	-
	Agente Humano de Prueba es-un Agente de Prueba	
	Agente Automatizado de Prueba es-un Agente de Prueba	
CQ16. ¿Un Proceso de Prueba implica roles?	Proceso de Prueba involucra Rol de Prueba	A_XVII
CQ17. ¿Puede un Agentes de Prueba usar herramientas?	Agente de Prueba usa Herramienta de Prueba	-
CQ18. ¿En qué Situación Particular una Entidad Comprobable se considera una Entidad Evaluable?	Requisito de Prueba hace referencia a una Entidad Comprobable	A_I
	Requisito de Prueba se basa en una Base de Prueba	
	Base de Prueba esta enlazada a Requisito No Funcional	
CQ19. ¿En qué Situación Particular una Entidad Comprobable se considera una Entidad Desarrollable?	Requisito de Prueba hace referencia a una Entidad Comprobable	A_II
	Requisito de Prueba se basa en una Base de Prueba	
	Base de Prueba esta enlazada a Requisito Funcional	
CQ20. En una Situación Particular, ¿puede una Entidad Comprobable estar rodeada de Entidades de Contexto de Prueba?	Entidad Comprobable está rodeada de Entidad de Contexto de Prueba	-
	Entidad de Contexto de Prueba influencia Entidad Comprobable	
	Situación Particular de Prueba trata con el objetivo de prueba Entidad Comprobable	
	Situación Particular de Prueba trata con el entorno de prueba Entidad de Contexto de Prueba	
CQ21. ¿Se puede derivar una Meta de Prueba en uno o más Requisitos de Prueba?	Meta de Prueba se deriva en Requisito de Prueba	A_III
CQ22. ¿Existe una Actividad de Prueba que considere una Meta de Necesidad de Información de Prueba?	Analizar Resultados de Prueba es-una Actividad de Prueba	-
	Analizar Resultados de Prueba toma en cuenta una Necesidad de Información de Prueba	
CQ23. Para un Requisito de Prueba que se refiere a un Objeto de Prueba, ¿cuál es el nivel de prueba?	Requisito de Prueba hace referencia a una Entidad Comprobable	-
	Requisito de Prueba hace referencia a una Entidad de Contexto de Prueba	
	Requisito de Prueba tiene la propiedad llamada ***nivel de prueba***	
CQ24. ¿Un Requisito de Prueba está relacionado con Requisitos Funcionales y No Funcionales?	Requisito de Prueba se basa en una Base de Prueba	A_I, A_II
	Base de Prueba esta enlazada a Requisito No Funcional	
	Base de Prueba esta enlazada a Requisito Funcional	
CQ25. Para un Proyecto de Prueba que operacionaliza una Meta de Prueba, ¿tiene este proyecto una Estrategia de Prueba asociada que ayuda a alcanzar el propósito de la Meta de Prueba?	Proyecto de Prueba operacionaliza Meta de Prueba	A_IV
	Proyecto de Prueba asocia Estrategia de Prueba	
	Estrategia de Prueba ayuda a alcanzar Meta de Prueba	
	Meta de Prueba tiene la propiedad llamada *propósito*	

APÉNDICE D: LISTA DE VERIFICACIÓN ESTÁTICA PARA LAS RELACIONES DE TESTTDO

Este apéndice contiene, en la Tabla D-1, la Lista de Verificación estática completa usada para la verificación de la consistencia semántica de las relaciones de TestTDO en su versión 1.0. Recordar que se produjo como resultado de la actividad Realizar Pruebas Estáticas y la misma esta ilustrada en la Sección 4.3.3.3.

Tabla D-1. Lista de Verificación utilizada para inspeccionar la consistencia semántica de las relaciones de TestTDO v1.0 frente a las relaciones que las enriquecen de las ontologías de niveles superiores (por ejemplo, la ontología ProcessCO).

Relación de TestTDO	**Término 1 de TestTDO <<Estereotipo>>**	**Término 2 de TestTDO <<Estereotipo>>**	**Inspeccionar**	**Resultado Real (paso/fallo)**	**Descripción del Incidente (si hay un fallo)**
adopts	Gestión de las Pruebas <<Gestión del Proyecto de ProjectCO>>	Ciclo de Vida de las Pruebas << Ciclo de Vida del Proyecto de ProjectCO>>	¿Existe una coincidencia semántica entre la relación denominada "*adopts*" de TestTDO con alguna relación de los conceptos de Gestión del Proyecto y Ciclo de Vida del Proyecto (de ProjectCO)?	**paso**	
aggregates	Proyecto de Prueba <<Proyecto de ProjectCO>>	Proyecto de Prueba <<Proyecto de ProjectCO>>	¿Existe una relación de agregación del concepto Proyecto (de ProjectCO) consigo mismo?	**paso**	
aggregates	Proceso de Prueba <<Proceso de Trabajo de ProcessCO>>	Actividad de Prueba <<Actividad de ProcessCO>>	¿Existe una relación de agregación entre los conceptos de Proceso de Trabajo y Actividad (de ProcessCO)?	**paso**	
aggregates	Conjunto de Casos de Prueba <<Artefacto de ProcessCO>>	Caso de Prueba <<Artefacto de ProcessCO>>	¿Existe una relación de agregación del concepto Artefacto (de ProcessCO) consigo mismo?	**paso**	

associates	Proyecto de Prueba <<Proyecto de ProjectCO>>	Estrategia de Prueba <<Estrategia de ProjectCO>>	¿Existe una coincidencia semántica entre la relación denominada "*associates*" de TestTDO con alguna relación de los conceptos de Proyecto y Estrategia (de ProjectCO)?	**fallo**	No hay relación entre Proyecto y Estrategia en ProjectCO. Sin embargo, la relación de "*associates*" es adecuada entre el Proyecto de Prueba y la Estrategia de Prueba. Por lo tanto, se recomienda que esta relación se agregue en ProjectCO.
consumes	Analizar Resultados de Prueba <<Actividad de ProcessCO>>	Resultado de Prueba <<Producto de Trabajo de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*consumes*" de TestTDO con alguna relación de los conceptos de Actividad y Producto de Trabajo (de ProcessCO)?	**paso**	
consumes	Diseñar la Prueba <<Actividad de ProcessCO>>	Base de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*consumes*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
consumes	Realizar la Prueba <<Actividad de ProcessCO>>	Especificación de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*consumes*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
consumes	Analizar Resultados de Prueba <<Actividad de ProcessCO>>	Especificación de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*consumes*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
defines	Proyecto de Prueba <<Proyecto de ProjectCO>>	Entidad de Contexto de Prueba <<Entidad de Contexto de SituationCO>>	¿Existe una coincidencia semántica entre la relación denominada "*defines*" de TestTDO con alguna relación de los conceptos de Proyecto (de ProjectCO) y Entidad de Contexto (de SituationCO)?	**fallo**	No existe una relación entre Proyecto y Entidad de Contexto. La relación "*defines*" existe solo entre Proyecto y Situación Particular.

has assigned	Actividad de Prueba <<Actividad de ProcessCO>>	Agente de Prueba <<Agente de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*has assigned*" de TestTDO con alguna relación de los conceptos de Actividad y Agente (de ProcessCO)?	**paso**	
has assigned	Diseñar la Prueba <<Actividad de ProcessCO>>	Método de Diseño de Prueba <<Método de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*has assigned*" de TestTDO con alguna relación de los conceptos de Actividad y Método (de ProcessCO)?	**paso**	
has assigned	Realizar Pruebas Estáticas <<Actividad de ProcessCO>>	Método de Prueba Estática <<Método de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*has assigned*" de TestTDO con alguna relación de los conceptos de Actividad y Método (de ProcessCO)?	**paso**	
has assigned	Realizar Pruebas Dinámicas <<Actividad de ProcessCO>>	Método de Prueba Dinámica <<Método de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*has assigned*" de TestTDO con alguna relación de los conceptos de Actividad y Método (de ProcessCO)?	**paso**	
helps to achieve	Estrategia de Prueba <<Estrategia de ProjectCO>>	Meta de Prueba <<Meta Específica de GoalCO>>	¿Existe una coincidencia semántica entre la relación denominada "*helps to achieve*" de TestTDO con alguna relación de los conceptos de Estrategia (de ProjectCO) y Meta Específica (de GoalCO)?	**paso**	
involves	Proceso de Prueba <<Proceso de Trabajo de ProcessCO>>	Rol de Prueba <<Rol de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*involves*" de TestTDO con alguna relación de los conceptos de Proceso de Trabajo y Rol (de ProcessCO)?	**paso**	

is based on	Requisito de Prueba <<Aserción de ThingFO>>	Base de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*is based on*" de TestTDO con alguna relación de los conceptos de Aserción sobre Particulares (de ThingFO) y Artefacto (de ProcessCO)?	**fallo**	No existe una relación entre Aserciones y Artefactos que coincida con "*is based on*". Por lo tanto, el término Requisito de Prueba debería ser estereotipado como <<Aserción sobre Particulares de ThingFO>> especializando la relación "*deals with particulars*".
is based on	Procedimiento de Realización como Artefacto	Especificación de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*is based on*" de TestTDO con alguna relación del concepto de Artefacto (de ProcessCO) consigo mismo?	**paso**	
is derived in	Meta de Prueba <<Meta Específica de GoalCO>>	Requisito de Prueba <<Aserción de ThingFO>>	¿Existe una coincidencia semántica entre la relación denominada "*is derived in*" de TestTDO con alguna relación de los conceptos de Meta Específica (de GoalCO) y Aserción sobre Particulares (de ThingFO)?	**paso**	
is in a particular situation with	Entidad Comprobable <<Entidad Evaluable / Desarrollable de NFRsTDO / FRsTDO>>	Entidad de Contexto de Prueba <<Entidad de Contexto de SituationCO>>	¿Existe una coincidencia semántica entre la relación denominada "*is in a particular situation with*" de TestTDO con alguna relación de los conceptos de Entidad Evaluable / Desarrollable (de NFRsTDO / FRsTDO) y Entidad de Contexto (de SituationCO)?	**fallo**	Existe una no conformidad semántica de la relación "*is in a particular situation with*" considerando la relación entre Entidad Evaluable / Desarrollable y Entidad de Contexto en SituationCO. La relación entre estos términos se denomina "*is surrounded by*". Además, existe el término Situación Particular que relaciona ambos términos en SituaciónCO. Este término tiene la semántica que la relación "*is in a particular situation with*" quiere transmitir. También falta la relación "*influences*".
is managed by	Proyecto de Prueba <<Proyecto de ProjectCO>>	Gestión de las Pruebas <<Gestión del Proyecto de ProjectCO>>	¿Existe una coincidencia semántica entre la relación denominada "*is managed by*" de TestTDO con alguna relación de los conceptos de Proyecto y Gestión del Proyecto (de ProjectCO)?	**paso**	

is supported by	Meta de Prueba <<Meta Específica de GoalCO>>	Meta de Necesidad de Información de Prueba << Meta de Necesidad de Información de GoalCO>>	¿Existe una coincidencia semántica entre la relación denominada "*is supported by*" de TestTDO con alguna relación de los conceptos de Meta Específica y Meta de Necesidad de Información (de GoalCO)?	**paso**	
kind of	Agente Humano de Prueba <<Agente Humano de ProcessCO>>	Agente de Prueba <<Agente de ProcessCO>>	¿Existe una relación "*kind of*" entre Agente Humano y Agente (de ProcessCO)?	**paso**	
kind of	Agente Automatizado de Prueba <<Agente Automatizado de ProcessCO>>	Agente de Prueba <<Agente de ProcessCO>>	¿Existe una relación "*kind of*" entre Agente Automatizado y Agente (de ProcessCO)?	**paso**	
kind of	Resultado de Prueba <<Producto de Trabajo de ProcessCO>>	Resultado Real <<Artefacto de ProcessCO>>	¿Existe una relación "*kind of*" entre Producto de Trabajo y Artefacto (de ProcessCO)?	**paso**	
kind of	Resultado de Prueba <<Producto de Trabajo de ProcessCO>>	Incidente <<Artefacto de ProcessCO>>	¿Existe una relación "*kind of*" entre Producto de Trabajo y Artefacto (de ProcessCO)?	**paso**	
operationalizes	Proyecto de Prueba <<Proyecto de ProjectCO>>	Meta de Prueba <<Meta Específica de GoalCO>>	¿Existe una coincidencia semántica entre la relación denominada "operationalizes" de TestTDO con alguna relación de los conceptos de Proyecto (de ProjectCO) y Meta Específica (de GoalCO)?	**paso**	
plays	Agente de Prueba <<Agente de ProcessCO>>	Rol de Prueba <<Rol de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*plays*" de TestTDO con alguna relación de los conceptos de Agente y Rol (de ProcessCO)?	**paso**	
produces	Gestión de las Pruebas <<Gestión del Proyecto de ProjectCO>>	Plan de Prueba <<Plan de ProjectCO>>	¿Existe una coincidencia semántica entre la relación denominada "*produces*" de TestTDO con alguna relación de los conceptos de Gestión del	**paso**	

			Proyecto y Plan (de ProjectCO)?		
produces	Realizar la Prueba <<Actividad de ProcessCO>>	Resultado de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*produces*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
produces	Analizar Resultados de Prueba <<Actividad de ProcessCO>>	Informe de Conclusión de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*produces*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
produces	Diseñar la Prueba <<Actividad de ProcessCO>>	Especificación de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*produces*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
produces	Diseñar la Prueba <<Actividad de ProcessCO>>	Procedimiento de Realización como Artefacto	¿Existe una coincidencia semántica entre la relación denominada "*produces*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
refers to	Requisito de Prueba <<Aserción de ThingFO>>	Entidad Comprobable <<Entidad Evaluable / Desarrollable de NFRsTDO / FRsTDO>>	¿Existe una coincidencia semántica entre la relación denominada "*refers to*" de TestTDO con alguna relación de los conceptos de Aserción (de ThingFO) y Entidad Evaluable / Desarrollable (de NFRsTDO / FRsTDO)?	**fallo**	No existe una relación entre Aserciones y Entidades que coincida con "*refers to*". Por lo tanto, el término Requisito de Prueba debería ser estereotipado como <<Aserción sobre Particulares de ThingFO>> especializando la relación "*deals with particulars*".
relies on	Incidente <<Artefacto de ProcessCO>>	Resultado Real <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*relies on*" de	**paso**	

			TestTDO con alguna relación del concepto de Artefacto (de ProcessCO) consigo mismo?		
requires as input	Proceso de Prueba <<Proceso de Trabajo de ProcessCO>>	Entidad Comprobable <<Entidad Evaluable / Desarrollable de NFRsTDO / FRsTDO>>	¿Existe una coincidencia semántica entre la relación denominada "*requires as input*" de TestTDO con alguna relación de los conceptos de Proceso de Trabajo (de ProcessCO) y Entidad Evaluable / Desarrollable (de NFRsTDO / FRsTDO)?	**paso**	
takes into account	Analizar Resultados de Prueba <<Actividad de ProcessCO>>	Meta de Necesidad de Información de Prueba <<Meta de Necesidad de Información de GoalCO>> (como Artefacto)	¿Existe una coincidencia semántica entre la relación denominada "*takes into account*" de TestTDO con alguna relación de los conceptos de Actividad y Artefacto (de ProcessCO)?	**paso**	
uses	Agente de Prueba <<Agente de ProcessCO>>	Herramienta de Prueba <<Herramienta de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*uses*" de TestTDO con alguna relación de los conceptos de Agente y Herramienta (de ProcessCO)?	**paso**	
uses	Ciclo de Vida de las Pruebas <<Ciclo de Vida del Proyecto de ProjectCO>>	Estrategia de Prueba <<Estrategia de ProjectCO>>	¿Existe una coincidencia semántica entre la relación denominada "*uses*" de TestTDO con alguna relación de los conceptos de Ciclo de Vida del Proyecto y Estrategia (de ProjectCO)?	**paso**	
uses	Método de Diseño de Prueba <<Método de ProcessCO>>	Base de Prueba <<Artefacto de ProcessCO>>	¿Existe una coincidencia semántica entre la relación denominada "*uses*" de TestTDO con alguna relación de los conceptos de Método y Artefacto (de ProcessCO)?	**fallo**	No existe relación entre Método y Artefacto en ProcessCO. Se recomienda eliminar esta relación.

verifies / validates	Especificación de Prueba <<Artefacto de ProcessCO / Cosa de ThingFO>>	Entidad Comprobable <<Entidad Evaluable / Desarrollable de NFRsTDO / FRsTDO / Cosa de ThingFO>>	¿Existe una coincidencia semántica entre la relación denominada "*verifies / validates*" de TestTDO con alguna relación de los conceptos de Artefacto / Thing (de ProcessCO / ThingFO) y Entidad Evaluable / Desarrollable / Thing (de NFRsTDO / FRsTDO / ThingFO)?	**paso**

APÉNDICE E: PRUEBA DE CONCEPTO DE TESTTDO

Este apéndice contiene una prueba de concepto para TestTDO, la cual está basada en un ejemplo tomado de (Myers et al., 2012). En este ejemplo se propone probar una aplicación que tiene como entrada tres enteros los cuales representan los lados de un triángulo. Además, el programa debe brindar como salida el tipo de triángulo correspondiente según los valores de las entradas (i.e., escaleno, isósceles o equilátero).

En esta prueba de concepto se brindó la semántica, usando los términos y propiedades de TestTDO, a cada elemento de prueba del ejemplo anteriormente mencionado. Recordar que esta validación se llevó a cabo como parte de la actividad Valoración, ilustrada en la Sección 4.3.3.3. A continuación se listan dichas instancias con su semántica correspondiente asignada:

Requisito Funcional:

-*nombre*: FR#1

-*declaración*: Se requiere un programa que reciba tres enteros, donde cada valor represente el lado de un triángulo, y teniendo en cuenta estos valores de entrada devuelva como resultado el tipo de triángulo (ya sea isósceles, o escaleno o equilátero).

- *Solicitante*: Profesor de un curso de pregrado sobre *testing*.

Entidad Comprobable:

-*nombre*: sistema de figuras geométricas

-*descripción*: es una aplicación de software didáctica para representar diferentes figuras geométricas y sus características, ingresando sus parámetros.

Elemento de Prueba:

- *nombre*: tipo_triangulo

-*descripción*: es la función que implementa el FR#1.

Proyecto de Prueba:

-*nombre*: TP#1

Estrategia de Prueba:

-*nombre*: Estrategia de Prueba Dinámica

Meta de Prueba:

-*etiqueta*: TG#1

-*declaración*: verificar que todas las unidades funcionales del sistema de figuras geométricas funcionen correctamente, de acuerdo a los requisitos funcionales.

-*propósito*: verificar

-*criterio de éxito*: probar el 100% de las unidades funcionales del sistema.

Metas de Necesidad de Información de Prueba:

1)

-*etiqueta*: TIN#1

- *declaración*: analizar si se han verificado todas las unidades funcionales del sistema de figuras geométricas.

- *propósito*: analizar

2)

-*etiqueta*: TIN#2

- *declaración*: analizar si los Resultados de Pruebas de cada unidad funcional han alcanzado los criterios de finalización.

- *propósito*: analizar

Requisito de Prueba:

-*etiqueta*: TR#1

-*declaración*: verificar que la Entidad Comprobable denominada "tipo_triangulo" se ajusta al Requisito Funcional FR#1.

-*fase de la entidad comprobable*: desarrollo

-*nivel de prueba*: unitario

-*criterio de finalización*: alcanzar el 100% de cobertura según el método utilizado.

Agente Humano de Prueba:

-*nombre*: THA#1

-*capacidades*: tiene experiencia llevando a cabo procesos de pruebas.

Rol de Prueba:

-*nombre*: *tester*

-*habilidades*: diseñar y ejecutar Casos de Prueba

Para llevar a cabo la actividad de Diseñar la Prueba, el agente THA#1 utilizó dos Métodos de Diseño de Pruebas para probar el Elemento de Prueba "tipo_triangulo":

Método de Diseño de Prueba 1 – Método basado en la Estructura (caja blanca):

-*nombre*: pruebas de sentencias

-*regla*: los Casos de Prueba deben cubrir al menos tantas sentencias del código fuente como lo requieran los criterios de finalización del TR#1.

-*procedimiento de diseño*: desarrollar el grafo de flujo de control (ver Figura E-1); luego, producir Casos de Prueba para cubrir tanto nodos del grafo como porcentaje de cobertura se requiera.

Bases de Prueba (Artefacto a ser consumido por la actividad Diseñar la Prueba): el Requisito Funcional FR#1 y el código fuente del Elemento de Prueba "tipo_triangulo". Supongamos que el siguiente código fuente, escrito en lenguaje C, representa la implementación de la función "tipo_triangulo":

```
char* triangle_type (int sideA, int sideB, int sideC){
    char *result; (1)
    strcpy (result, "Invalid triangle"); (2)
    if(sideA==sideB && sideB==sideC){ //if it is equilateral  (3)
            strcpy(result, "Equilateral"); (4)
```

```
    } else if(sideA!=sideB && sideA!=sideC && sideB!=sideC){ // if it is scalene (5)
            strcpy(result, “Scalene”); (6)
    } else{ // it is isosceles
            strcpy(result, “Isosceles”); (7)
    }
    return result; (8)
}
```

Además, como Base de Prueba, la Figura E-1 ilustra el grafo de flujo de control correspondiente al código fuente para la función “tipo_triangulo”. Notar que cada número de cada nodo se corresponde con el número de sentencia especificado en el código fuente (ver los números en rojos y entre paréntesis en el código).

Caso de Prueba: (notar que faltan al menos 2 Casos de Prueba más para alcanzar el 100% de cobertura de sentencias, es decir, el criterio de finalización)

-*nombre*: TC#1_PS

-*versión*: 1.0

-*precondición*: ninguna

-*entrada*: (1, 1, 1)

-*resultado esperado*: “Equilateral”

-*postcondición*: la función finaliza correctamente, sin arrojar errores en tiempo de ejecución.

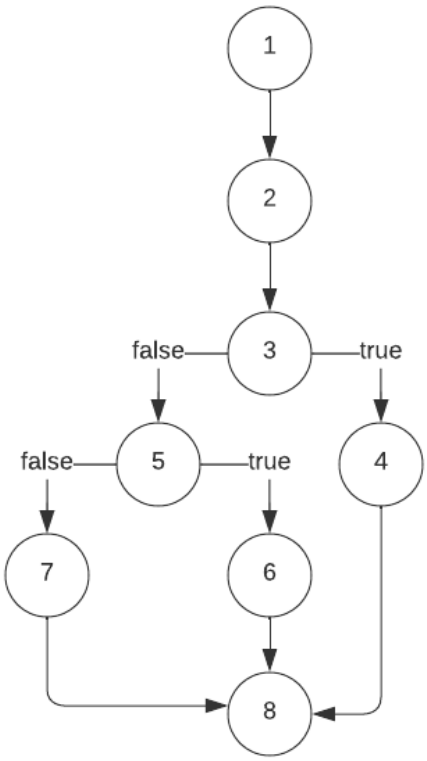


Figura E-1. Grafo de flujo de control correspondiente al código fuente de la función denominada “tipo_triangulo”.

Método de Diseño de Prueba 2 – Método basado en la Especificación (caja negra):

-*nombre*: particiones de equivalencia

-*regla*: a) Los Casos de Prueba deben cubrir tantas particiones de equivalencia como lo requieran los criterios de finalización del TR#1; b) tener en cuenta particiones para entradas válidas e inválidas.

-*procedimiento de diseño*: analizar las Bases de Prueba para determinar las particiones de equivalencia y producir los Casos de Prueba correspondientes.

Bases de Prueba (Artefacto a ser consumido por la actividad Diseñar la Prueba): el Requisito Funcional FR#1 y las siguientes particiones de equivalencia (PE):

PE#1: para triángulos válidos → X = {(sideA, sideB, sideC) donde (sideA + sideB > sideC) y (sideB + sideC > sideA) y (sideA + sideC > sideB)}

PE#2: para triángulos no válidos → Y = ¬X

Caso de Prueba:

1)

-*nombre*: TC#1_PE

-*versión*: 1.0

-*precondición*: ninguna

-*entrada*: (2, 3, 4)

-*resultado esperado*: “Scalene”

-*postcondición*: la función finaliza correctamente, sin arrojar errores en tiempo de ejecución.

2)

-*nombre*: TC#2_PE

-*versión*: 1.0

-*precondición*: ninguna

-*entrada*: (1, 1, 2)

-*resultado esperado*: “Invalid triangle”

-*postcondición*: la función finaliza correctamente, sin arrojar errores en tiempo de ejecución.

Conjunto de Casos de Prueba: incluye TC#1_PS, TC#1_PE, y TC#2_PE.

-*nombre*: TS#1

-*versión*: 1.0

Procedimiento de Realización:

-*especificación*: A continuación, se muestra el código fuente en C para ejecutar el Conjunto de Casos de Prueba llamado TS#1:

```
#include <assert.h>
void testSuiteTS1(){ // Test Suite named TS#1
        assert(triangle_type(1,1,1), “Equilateral”); //TC#1_PS
        assert(triangle_type(2,3,4), “Scalene”); //TC#1_ PE
```

```
        assert(triangle_type(1,1,2), “Invalid triangle”);  //TC#2_ PE
}
```

Para llevar a cabo la actividad de Realizar la Prueba Dinámica Funcional, se utilizó el Procedimiento de Realización anterior producido en la actividad de Diseñar la Prueba. Notar que este Procedimiento de Realización forma parte del Método de Realización Dinámico asignado a la actividad de Realizar la Prueba Dinámica. El agente THA#1 ejecutó el código fuente especificado en el Procedimiento de Realización produciendo como producto de trabajo los siguientes Resultados de Prueba:

1) Para el Caso de Prueba TC#1_PS

Resultado Real:

-*nombre*: AR#1

-*valor*: “Equilateral”

2) Para TC#1_ PE

Resultado actual:

-*nombre*: AR#2

-*valor*: “Scalene”

3) Para TC#2_ PE

Resultado actual:

-*nombre*: AR#3

-*valor*: “Isosceles”

Incidente:

-*nombre*: I#1

-*descripción*: el Resultado Real no coincide con el resultado esperado, que es “Invalid triangle”.

El agente THA#1, teniendo en cuenta las dos Metas de Necesidad de Información de Prueba especificadas (TIN#1 y TIN#2), analizó los Resultados de Prueba y produjo el siguiente Informe de Conclusión de Prueba:

-*nombre*: TCR#1

-*versión*: 1.0.

-*descripción*: a) los Casos de Prueba TC#1_PS y TC#1_ PE pasaron con éxito; b) el Caso de Prueba denominado TC#2_ PE falló de acuerdo al Incidente I#1 registrado; c) se recomienda que se lleven a cabo acciones de cambio para que la función “tipo_triangulo” se ajuste adecuadamente al requisito funcional FR#1; d) las coberturas de particiones de equivalencia y de sentencias fueron del 100. Por lo tanto, el criterio de finalización del requisito de prueba TR#1 se cumplió para la función “tipo_triangulo”.

APÉNDICE F: PRUEBAS DINÁMICAS FUNCIONALES PARA TESTTDO

Este apéndice contiene todos los artefactos producidos al realizar pruebas dinámicas sobre la versión de OWL de la ontología TestTDO. Recordar que todos estos artefactos se produjeron como resultado de la actividad de verificación Pruebas Dinámicas Funcionales, ilustrada en la Sección 4.3.3.3.

Declaración del Requisitos de Prueba: "Verificar dinámicamente que TestTDO cumple con el alcance previsto en las CQs".

Entidad Comprobable: implementación en OWL de TestTDO

Método de Diseño de Pruebas: Método basado en la Especificación (método de caja negra)

Requisitos Funcionales: Preguntas de Competencia (CQs).

Bases de Prueba: CQs; conceptualización de TestTDO (además de su relación con los términos de Requisitos Funcionales y Requisitos No Funcionales); y datos de prueba tomados de las instancias del proyecto de prueba académico de una aplicación de figuras geométricas (ver Apéndice E).

Recordar que para esta actividad de verificación se desglosaron algunas CQs que eran muy genéricas en otras más específicas. A continuación, se muestran todas las **Preguntas de Competencia (CQs) seleccionadas y desglosadas**:

CQs relacionadas con Productos de Trabajo de Prueba:

CQ3. ¿Cuáles son los Productos de Trabajo producidos por una actividad de Realizar la Prueba?

CQ3 se dividió en 2 sub-CQs, a saber:

CQ3.1. ¿Cuáles son los Resultados Reales producidos por una actividad de Realizar la Prueba?

CQ3.2. ¿Cuáles son los Incidentes producidos por una actividad de Realizar la Prueba?

CQs relacionadas con Actividades de Prueba:

CQ11. ¿Cuáles son los tipos de entidades de prueba requeridas por un Proceso de Prueba como entrada?

CQ11 se dividió en 2 sub-CQs, a saber:

CQ11.1. ¿Cuál es la Entidad Comprobable requerida por un Proceso de Prueba como entrada?

CQ11.2. ¿Cuáles son las Entidades de Contexto de Prueba requeridas por un Proceso de Prueba como entrada?

CQs relacionadas con Métodos de Prueba:

CQ14. ¿Cuáles son los tipos de Métodos de Prueba asignados a una actividad de Diseñar la Prueba?

CQs relacionadas con Agentes de Prueba:

CQ16. ¿Un Proceso de Prueba implica roles?

CQs relacionadas con entidades de prueba:

CQ19. ¿En qué Situación Particular una Entidad Comprobable se considera una Entidad Desarrollable?

CQs relacionadas con Metas de Prueba:

CQ21. ¿Se puede derivar una Meta de Prueba en uno o más Requisitos de Prueba?

CQs relacionadas con Requisitos de Prueba:

CQ23. Para un Requisito de Prueba que se refiere a un Objeto de Prueba, ¿cuál es el nivel de prueba?

CQs relacionadas con Proyectos de Prueba:

CQ25. Para un proyecto de prueba que operacionaliza una meta de prueba, ¿tiene este proyecto una estrategia de testing asociada que ayuda a alcanzar el propósito de la meta de prueba?

A continuación se ilustra, para cada CQ, el Caso de Prueba diseñado y el Resultado Real obtenido al ejecutarlo. La postcondición para todos los Casos de Prueba diseñados es que su Resultado Real obtenido y su resultado esperado coincidan. Además, notar que se utilizaron consultas en SPARQL como entradas de los Casos de Prueba. Para todas las consultas en SPARQL se utilizaron los siguientes prefijos:

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 TestTDO: <http://www.semanticweb.org/ontologies/2020/TestTDO.owl#>

CQ3.1. ¿Cuáles son los Resultados Reales producidos por una actividad de Realizar la Prueba?

Caso de Prueba TC#1:

Precondición: Que existen individuos implementados de Resultados Reales y actividades de Realizar la Prueba que los hayan producido.

Entrada:

```
SELECT ?ActualResult_Name ?ActualResult_Value WHERE { ?Testing_Realization TestTDO:Testing_Realization-produces-Test_Result ?Actual_Result. ?Actual_Result TestTDO:Work_Product-hasName ?ActualResult_Name. ?Actual_Result TestTDO:Outcome hasValue ?ActualResult_Value}
```

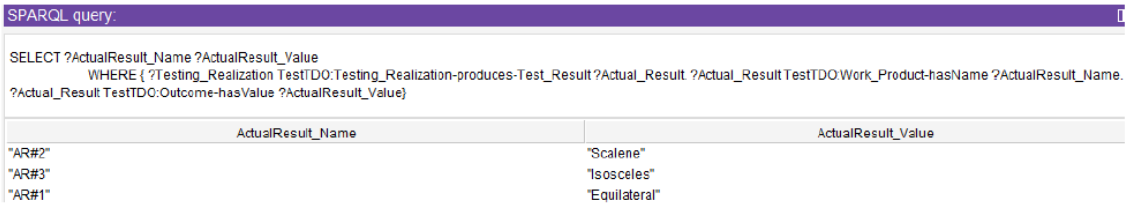
Resultado esperado:

3 individuos de Resultados Reales con los siguientes datos:

Actual Results (AR)	*name*	*value*
AR1	AR#1	Equilateral

AR2	AR#2	Scalene
AR3	AR#3	Isosceles

Resultado Real:


SPARQL query:

SELECT ?ActualResult_Name ?ActualResult_Value
WHERE { ?Testing_Realization TestTDO:Testing_Realization-produces-Test_Result ?Actual_Result. ?Actual_Result TestTDO:Work_Product-hasName ?ActualResult_Name. ?Actual_Result TestTDO:Outcome-hasValue ?ActualResult_Value}

ActualResult_Name	ActualResult_Value
"AR#2"	"Scalene"
"AR#3"	"Isosceles"
"AR#1"	"Equilateral"

CQ3.2. ¿Cuáles son los Incidentes producidos por una actividad de Realizar la Prueba?

Caso de Prueba TC#2:

Precondición: Que existen individuos implementados de Incidentes y actividades de Realizar la Prueba que los hayan producido.

Entrada:

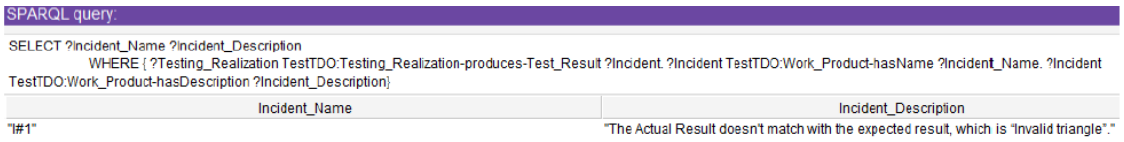
SELECT ?Incident_Name ?Incident_Description WHERE { ?Testing_Realization TestTDO:Testing_Realization-produces-Test_Result ?Incident. ?Incident TestTDO:Work_Product-hasName ?Incident_Name. ?Incident TestTDO:Work_Product hasDescription ?Incident_Description }

Resultado esperado:

1 individuo de Incidente con los siguientes datos:

Incident (I)	*name*	*description*
I1	I#1	The Actual Result doesn't match with the expected result, which is “Invalid triangle”.

Resultado Real:


SPARQL query:

SELECT ?Incident_Name ?Incident_Description
WHERE { ?Testing_Realization TestTDO:Testing_Realization-produces-Test_Result ?Incident. ?Incident TestTDO:Work_Product-hasName ?Incident_Name. ?Incident TestTDO:Work_Product-hasDescription ?Incident_Description}

Incident_Name	Incident_Description
"I#1"	"The Actual Result doesn't match with the expected result, which is "Invalid triangle"."

CQ11.1. ¿Cuál es la Entidad Comprobable requerida por un Proceso de Prueba como entrada?

Caso de Prueba TC#3:

Precondición: Que existen individuos implementados de Entidad Comprobable y Proceso de Prueba que la haya requerido como entrada.

Entrada:

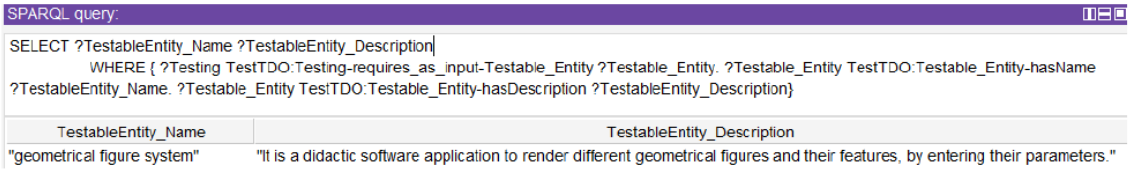
```
SELECT ?TestableEntity_Name ?TestableEntity_Description WHERE { ?Testing TestTDO:Testing-requires_as_input-Testable_Entity ?Testable_Entity. ?Testable_Entity TestTDO:Testable_Entity-hasName ?TestableEntity_Name. ?Testable_Entity TestTDO:Testable_Entity-hasDescription ?TestableEntity_Description}
```

Resultado esperado:

1 individuo de Entidad Comprobable con los siguientes datos:

Testable Entity (TE)	*name*	*description*
TE1	geometrical figure system	It is a didactic software application to render different geometrical figures and their features, by entering their parameters.

Resultado Real:



CQ11.2. ¿Cuáles son las Entidades de Contexto de Prueba requeridas por un Proceso de Prueba como entrada?

Caso de Prueba TC#4:

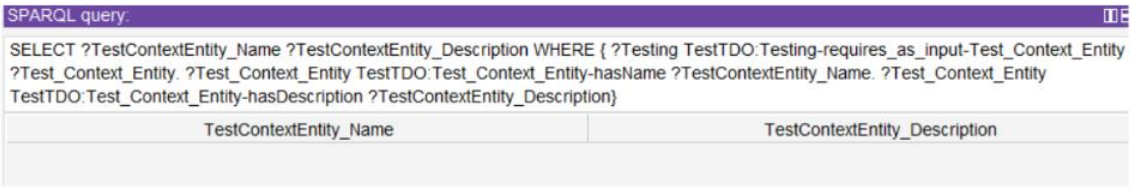
Precondición: Que no existen individuos implementados de Entidades de Contexto y tampoco de Proceso de Prueba que las haya requerido como entrada.

Entrada:

```
SELECT ?TestContextEntity_Name ?TestContextEntity_Description WHERE { ?Testing TestTDO:Testing-requires_as_input-Test_Context_Entity ?Test_Context_Entity. ?Test_Context_Entity TestTDO:Test_Context_Entity-hasName ?TestContextEntity_Name. ?Test_Context_Entity TestTDO:Test_Context_Entity-hasDescription ?TestContextEntity_Description}
```

Resultado esperado: ningún individuo de Entidad de Contexto.

Resultado Real:



CQ14. ¿Cuáles son los tipos de Métodos de Prueba asignados a una actividad de Diseñar la Prueba?

Caso de Prueba TC#5:

Precondición: Que existen individuos implementados de actividades de Diseñar la Prueba y Métodos de Diseño de Prueba asignados a ellas.

Entrada:

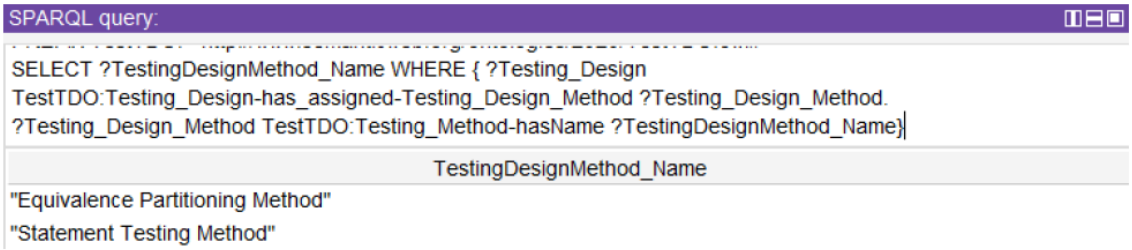
SELECT ?TestingDesignMethod_Name WHERE { ?Testing_Design TestTDO:Testing_Design-has_assigned-Testing_Design_Method Testing_Design_Method. ?Testing_Design_Method TestTDO:Testing_Method-hasName ?TestingDesignMethod_Name}

Resultado esperado:

2 individuos de Métodos de Diseño de Prueba con los siguientes datos:

Testing Design Method (TDM)	*name*
TDM1	Statement Testing Method
TDM2	Equivalence Partitioning Method

Resultado Real:



CQ16. ¿Un Proceso de Prueba implica roles?

Caso de Prueba TC#6:

Precondición: Que existen individuos implementados de Rol de Prueba y Proceso de Prueba que lo haya involucrado.

Entrada:

SELECT ?TestingRole_Name WHERE { ?Testing TestTDO:Testing-involves-Testing_Role ?Testing_Role. ?Testing_Role TestTDO:Testing_Role-hasName ?TestingRole_Name}

Resultado esperado:

1 individuo de Rol de Prueba con los siguientes datos:

Testing Role (TR)	*name*
TR1	Tester

Resultado Real:

SPARQL query:

```
SELECT ?TestingRole_Name WHERE { ?Testing TestTDO:Testing-involves-Testing_Role
?Testing_Role. ?Testing_Role TestTDO:Testing_Role-hasName ?TestingRole_Name}
```

TestingRole_Name
"Tester"

CQ19. ¿En qué Situación Particular una Entidad Comprobable se considera una Entidad Desarrollable?

Caso de Prueba TC#7:

Precondición: Que existen individuos implementados de Requisito de Prueba, Base de Prueba, Entidad Comprobable y Requisito Funcional, y exista alguna relación entre ellos.

Entrada:

SELECT ?TestableEntity_Name ?TestRequirement_Label ?TestBasis_Name ?FunctionalRequirement_Name WHERE { ?Test_Requirement TestTDO:Test_Requirement refers_to-Testable_Entity ?Testable_Entity. ?Test_Requirement TestTDO:Test_Requirement hasLabel ?TestRequirement_Label. ?Testable_Entity TestTDO:Testable_Entity-hasName ?TestableEntity_Name. ?Test_Requirement TestTDO:Test_Requirement-is_based-on Test_Basis ?Test_Basis. ?Test_Basis TestTDO:Work_Product-hasName ?TestBasis_Name. ?Test_Basis TestTDO:Test_Basis-is_linked_to-Functional_Requirement ?Functional_Requirement. ?Functional_Requirement TestTDO:Functional_Requirement hasName ?FunctionalRequirement_Name}

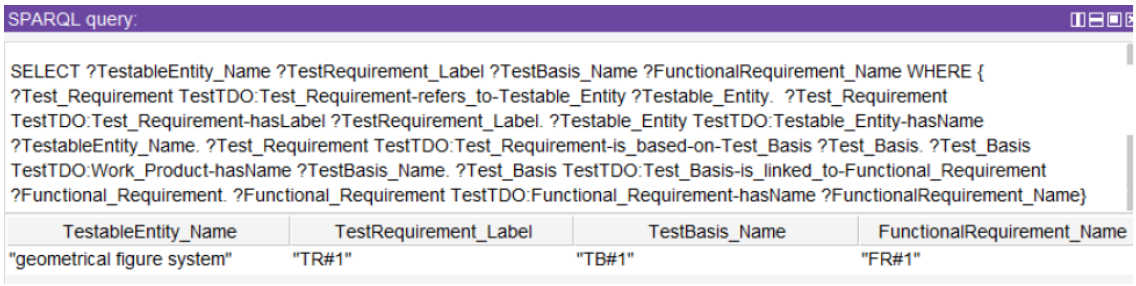
Resultado esperado:

1 instancia con los siguientes datos:

Testable Entity *name*	**Test Requirement** *label*	**Test Basis** *name*	**Functional Requirement** *name*

geometrical figure system	TR#1	TB#1	FR#1

Resultado Real:



CQ21. ¿Se puede derivar una Meta de Prueba en uno o más Requisitos de Prueba?

Caso de Prueba TC#8:

Precondición: Que existen individuos implementados de Requisito de Prueba y Meta de Prueba, y que estén relacionados entre ellos.

Entrada:

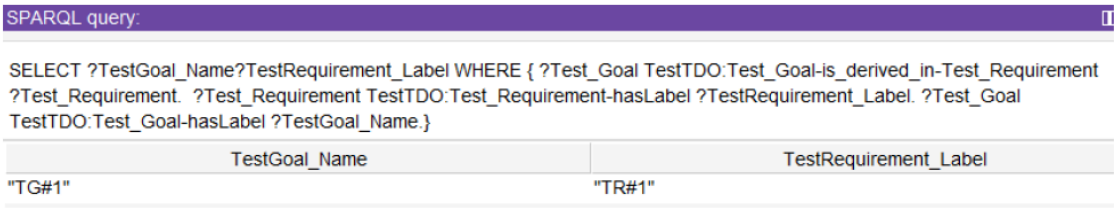
SELECT ?TestGoal_Name?TestRequirement_Label WHERE { ?Test_Goal TestTDO:Test_Goal-is_derived_in-Test_Requirement ?Test_Requirement. ?Test_Requirement TestTDO:Test_Requirement-hasLabel ?TestRequirement_Label. ?Test_Goal TestTDO:Test_Goal-hasLabel ?TestGoal_Name.}

Resultado esperado:

1 instancia con los siguientes datos:

Test Goal *label*	**Test Requirement** *label*
TG#1	TR#1

Resultado Real:



CQ23. Para un Requisito de Prueba que se refiere a un Objeto de Prueba, ¿cuál es el nivel de prueba?

Caso de Prueba TC#9:

Precondición: Que existen individuos implementados de Requisito de Prueba con un nivel de prueba instanciado, y una Entidad Comprobable asociada.

Entrada:

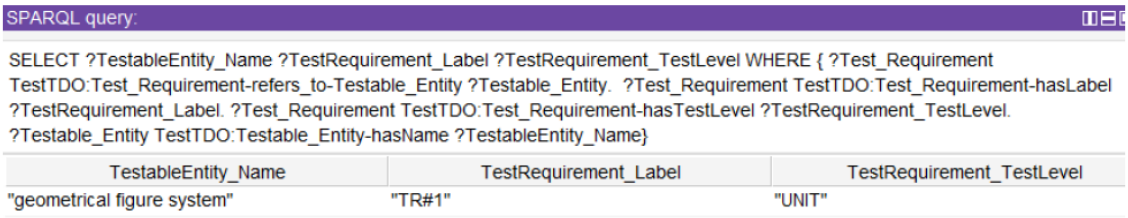
SELECT ?TestableEntity_Name ?TestRequirement_Label ?TestRequirement_TestLevel WHERE { ?Test_Requirement TestTDO:Test_Requirement-refers_to-Testable_Entity ?Testable_Entity. ?Test_Requirement TestTDO:Test_Requirement-hasLabel ?TestRequirement_Label. ?Test_Requirement TestTDO:Test_Requirement-hasTestLevel ?TestRequirement_TestLevel. ?Testable_Entity TestTDO:Testable_Entity-hasName ?TestableEntity_Name}

Resultado esperado:

1 instancia con los siguientes datos:

Testable Entity *name*	**Test Requirement** *label*	**Test Requirement** *test level*
geometrical figure system	TR#1	UNIT

Resultado Real:



CQ25. Para un proyecto de prueba que operacionaliza una meta de prueba, ¿tiene este proyecto una estrategia de testing asociada que ayuda a alcanzar el propósito de la meta de prueba?

Caso de Prueba TC#10:

Precondición: Que existen individuos implementados de Proyecto de Prueba, Meta de Prueba y Estrategia de Prueba, y están relacionados entre sí.

Entrada:

SELECT ?TestProject_Name ?TestingStrategy_Name ?TestGoal_Label ?TestGoal_Purpose WHERE { ?Test_Project TestTDO:Test_Project-operationalizes-Test_Goal ?Test_Goal. ?Test_Project TestTDO:Test_Project-associates-Testing_Strategy ?Testing_Strategy. ?Testing_Strategy TestTDO:Testing_Strategy-helps_to_achieve-Test_Goal ?Test_Goal. ?Test_Project TestTDO:Test_Project-hasName ?TestProject_Name. ?Testing_Strategy TestTDO:Testing_Strategy-hasName ?TestingStrategy_Name. ?Test_Goal TestTDO:Test_Goal-hasLabel

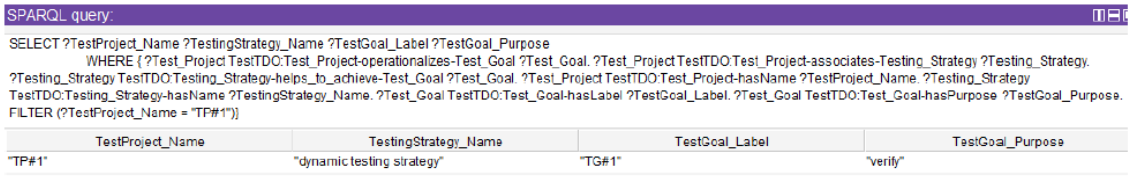
?TestGoal_Label. ?Test_Goal TestTDO:Test_Goal-hasPurpose ?TestGoal_Purpose.
FILTER (?TestProject_Name = "TP#1")}

Resultado esperado:

1 instancia con los siguientes datos:

Test Project *name*	**Testing Strategy** *name*	**Test Goal** *label*	**Test Goal** *purpose*
TP#1	dynamic testing strategy	TG#1	verify

Resultado Real:


SPARQL query:

SELECT ?TestProject_Name ?TestingStrategy_Name ?TestGoal_Label ?TestGoal_Purpose
WHERE { ?Test_Project TestTDO:Test_Project-operationalizes-Test_Goal ?Test_Goal. ?Test_Project TestTDO:Test_Project-associates-Testing_Strategy ?Testing_Strategy.
?Testing_Strategy TestTDO:Testing_Strategy-helps_to_achieve-Test_Goal ?Test_Goal. ?Test_Project TestTDO:Test_Project-hasName ?TestProject_Name. ?Testing_Strategy
TestTDO:Testing_Strategy-hasName ?TestingStrategy_Name. ?Test_Goal TestTDO:Test_Goal-hasLabel ?TestGoal_Label. ?Test_Goal TestTDO:Test_Goal-hasPurpose ?TestGoal_Purpose.
FILTER (?TestProject_Name = "TP#1")}

TestProject_Name	TestingStrategy_Name	TestGoal_Label	TestGoal_Purpose
"TP#1"	"dynamic testing strategy"	"TG#1"	"verify"
