Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de la Matanza
ISSN: 2525-1333
Vol.: 5 - Nro. 1 (AGOSTO-2020)

Artículo original

PROCESO DE DESIGN SCIENCE RESEARCH APLICADO A LA CONSTRUCCIÓN DE UNA ONTOLOGÍA DE TESTING DE SOFTWARE COMO ARTEFACTO

DESIGN SCIENCE RESEARCH PROCESS APPLIED TO THE CONSTRUCTION OF A SOFTWARE TESTING ONTOLOGY AS ARTIFACT

Luis OLSINA[(1)], María Belén RIVERA[(2)], María Fernanda PAPA[(3)], Pablo BECKER[(4)]

[(1)]GIDIS_Web, Facultad de Ingeniería, UNLPam
olsinal@ing.unlpam.edu.ar

[(2)]GIDIS_Web, Facultad de Ingeniería, UNLPam
riveramb@ing.unlpam.edu.ar

[(3)]GIDIS_Web, Facultad de Ingeniería, UNLPam
pmfer@ing.unlpam.edu.ar

[(4)]GIDIS_Web, Facultad de Ingeniería, UNLPam
beckerp@ing.unlpam.edu.ar

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

Resumen:

Design Science Research (DSR) es un enfoque de investigación riguroso que propone la construcción de artefactos para brindar una solución útil y efectiva a un problema de un dominio dado. El artefacto debe ser una solución innovadora a un problema no trivial. El desarrollo del artefacto implica un ciclo de actividades de diseño-construcción-evaluación, que iteran tantas veces como sean necesarias antes que el artefacto sea finalmente verificado, validado y comunicado para su utilización. La literatura existente respecto al enfoque DSR permite determinar que, al momento de este estudio, se encuentra débilmente especificado su proceso y sus perspectivas. Con el fin de contribuir en este aspecto, este trabajo presenta la especificación del proceso de DSR empleando perspectivas de modelado de proceso y usando el lenguaje SPEM (Software & Systems Process Engineering Meta-Model Specification). A su vez, se ilustrará este proceso mediante referencias a la construcción de un artefacto de software, el cual es una ontología de testing de software en el contexto de una arquitectura ontológica de cuatro capas.

Abstract:

Design Science Research (DSR) is a rigorous research approach that promotes the development of artifacts to provide a useful and effective solution to a given domain problem. The artifact should be an innovative solution for a non-trivial problem. The development of the artifact involves a cycle of design-development-evaluation activities, which iterates as many times as needed before the artifact is finally verified, validated and communicated for its use. The cutting-edge literature about the DSR approach permitted us to observe that, at the moment of this work, the perspectives of its process model are weakly specified. In order to contribute to this aspect, this work presents a DSR process specification using the SPEM (Software & Systems Process Engineering Meta-Model Specification) language to model its process perspectives. Furthermore, this process will be illustrated by applying its main activities in the construction of a software artifact, that is, a software testing ontology in the context of a four-tier ontological architecture.

Palabras Clave: *Design Science Research (DSR), Artefacto, Ontología de Pruebas de Software, Proceso DSR, SPEM*

Key Words: *Design Science Research (DSR), Artifact, Software Testing Ontology, DSR process, SPEM*

Colaboradores: *Guido TEBES, Denis PEPPINO*

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

I. INTRODUCCIÓN

Las organizaciones comúnmente establecen y persiguen metas de negocio con diferentes tipos de propósitos utilizando estrategias. Las metas de negocio son las metas principales o primarias que una organización intenta alcanzar y en su declaración, siempre subyace un propósito o intencionalidad. El propósito de una meta es la razón para alcanzarla. En organizaciones que gestionan proyectos de medición y evaluación de la calidad, las metas que dichos proyectos operacionalizan tienen propósitos de evaluación, como por ejemplo comprender, monitorear, mejorar, seleccionar una alternativa, controlar, entre otros. Si en cambio, el proyecto tiene que ver con desarrollo y/o mantenimiento, los propósitos de metas que se pueden distinguir son crear, agregar, eliminar o modificar una característica y/o capacidad concreta de una entidad. En tanto que proyectos relacionados a testing tienen que ver con metas que tengan propósitos tales como revisar, verificar, validar, entre otros.

Para alcanzar cualquiera de estos propósitos de metas de negocio, una organización se puede beneficiar a partir de la utilización de estrategias. En la práctica, una estrategia constituye un conjunto de acciones planificadas sistemáticamente en el tiempo que se llevan a cabo para lograr un fin o misión. Una estrategia proporciona la manera, junto a los medios, con el fin de alcanzar los propósitos que las metas de negocio establecen. En este trabajo se considera la definición del término estrategia como "un recurso de trabajo que involucra principios y capacidades integradas tales como bases conceptuales de un dominio (en el contexto de un marco conceptual), especificaciones de perspectivas de proceso, y especificaciones de métodos (y potenciales herramientas) que ayudan a alcanzar el propósito de una meta de proyecto". Por lo tanto, una estrategia integrada tendiente a satisfacer algún tipo de meta, debiera tener tres principios o capacidades integrados, a saber: i) una especificación de las perspectivas del proceso, ii) una especificación de métodos, y iii) una base conceptual de dominio bien establecida. Contar con estrategias que integren estos tres pilares permite comprender claramente qué hacer (procesos), cómo hacerlo (métodos y herramientas) y que exista un entendimiento común de los conceptos clave del dominio particular (bases conceptuales).

En Olsina y Becker [1] se discute una familia de estrategias que integran los tres pilares anteriormente mencionados para alcanzar propósitos de evaluación, es decir, aquellos relacionados con proyectos de medición, evaluación y mejora. Las bases conceptuales que dan soporte a dicha familia de estrategias se encuentran integradas en el marco conceptual denominado C-INCAMI v2.0 *(Contextual-Information Need, Concept Model, Attribute, Metric and Indicator)* desarrollado inicialmente en Olsina et al. [2], y luego actualizado en [3], [4]. Este marco se construye sobre terminologías estructuradas en ontologías. En la Fig. 1 se muestran los diferentes componentes de C-INCAMI v.2.0. Los módulos ya desarrollados hasta fines de 2018 se encuentran en la parte sombreada de dicha figura. Las ontologías de los componentes de Requisitos No Funcionales (RNFs), Requisitos Funcionales (RFs), meta de negocio, proyecto, contexto y vistas de RNFs se definen en [4], mientras que las ontologías de los componentes de medición (métricas) y evaluación (indicadores) se encuentran en [2]-[3]. Respecto a los componentes de testing, desarrollo y mantenimiento, vale acotar que, para dicha fecha se encontraban aún sin desarrollar.

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

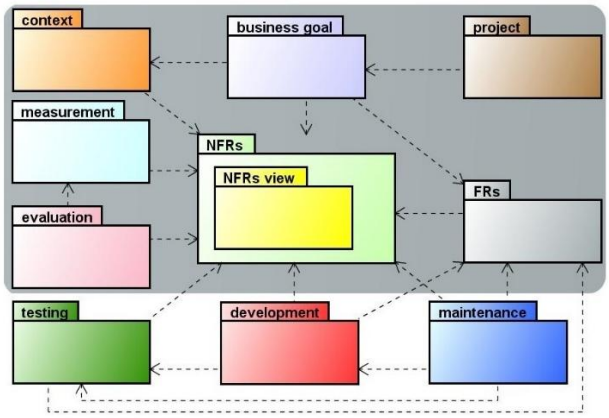

Fig. 1. Componentes conceptuales del marco C-INCAMI v2.0 y sus relaciones. Nota: el acrónimo en inglés NFRs, significa Requisitos No Funcionales (RNFs) en español; FRs, Requisitos Funcionales (RFs).

Teniendo presente que existen estrategias que brindan soporte para alcanzar propósitos de evaluación, es posible desarrollar otras estrategias integradas que ayuden a alcanzar los propósitos de metas de un proyecto de testing, mediante sus actividades y métodos bien establecidos. De acuerdo con lo indicado anteriormente, una estrategia de testing debería tener un marco conceptual, estructurado como una ontología para el dominio de testing. Contar con una ontología de testing es de utilidad pues permite tener definidos de manera explícita, formal, y sin ambigüedades los términos, propiedades, relaciones y axiomas (o restricciones) relacionados a las actividades de testing. Esto minimizaría los problemas de heterogeneidad que frecuentemente se observan en los diferentes trabajos que tratan sobre métodos y procesos de testing de software.

Por lo tanto, el presente trabajo propone utilizar el enfoque de investigación denominado Design Science Research (DSR) para la construcción de una ontología de testing de software como artefacto.

DSR es una metodología 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 según Hevner y Chatterjee [5], problemas no triviales. Es decir, aquellos problemas de difícil resolución debido a que están incompletos, son contradictorios o presentan requisitos cambiantes que, a menudo, son difíciles de relevar. Al iniciar el presente estudio, el problema detectado a resolver para el desarrollo de estrategias de testing de software residía en que, si bien existían bases conceptuales ontológicas para el dominio de testing, las mismas no se adecuaban semánticamente al robustecimiento del marco conceptual C-INCAMI con el fin de dar soporte a la especificación de una familia de estrategias para dicho dominio. La anterior declaración fue respaldada por el estudio realizado en Tebes et al. [6], como indicaremos más adelante.

DSR establece que el artefacto que se diseñe 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 [5]. Un artefacto se puede definir como una cosa u objeto creado mediante el trabajo de una o más personas con el subsecuente sentido de uso. Esta definición enfatiza dos aspectos importantes de un artefacto: que no ocurre naturalmente sino mediante el trabajo humano, y que tiene una utilidad determinada (Mckay y Marshall [7]).

De acuerdo al análisis realizado por March y Smith [8], los autores distinguen cuatro tipos de artefactos, a saber: constructores (vocabularios y símbolos), modelos (abstracciones y representaciones), métodos (algoritmos y prácticas) y también, considerando que 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. Por lo tanto, vale indicar que la ontología de testing de software

del presente estudio puede derivarse en un artefacto de tipo constructor y modelo, además en un artefacto de tipo implementación –por ejemplo, en un lenguaje semántico como OWL.

Para la construcción de cualquier tipo de artefacto de los arriba mencionados se lleva 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 (SI), se reconoce el trabajo de Peffers et al. [9] como uno de los pioneros en proponer el enfoque DSR para dar soporte a las investigaciones realizadas en esta disciplina. 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 [7], [9]-[13]. Las propuestas existentes coinciden en determinar que, para aplicar el enfoque DSR, se debe transitar por una primera etapa de identificar el problema y establecer su relevancia. Luego se debe diseñar, construir y evaluar (verificar/validar) el artefacto que dé solución al problema planteado para precisar que realmente significa una mejora. Finalmente, los resultados de los artefactos deben ser comunicados y difundidos efectivamente.

Al realizar un análisis de la literatura existente respecto al enfoque DSR nos permitió determinar, al momento del estudio, que se encuentra débilmente especificado su proceso. Tener un proceso formalmente especificado favorece repetitividad y reproducibilidad en su ejecución, y por lo tanto, se requiere para eso un conjunto de actividades con sus entradas (artefactos consumidos) y salidas (artefactos producidos), interdependencias, entre otros aspectos.

Curtis et al. [14] propone cuatro perspectivas (o vistas) para modelar un proceso, a saber: funcional, de comportamiento, informacional y organizacional. La primera describe qué actividades deben llevarse a cabo y qué flujo de artefactos (por ejemplo, documentos) es necesario para realizar las actividades y tareas. La vista de comportamiento especifica cuándo deben ejecutarse las actividades, incluyendo la identificación de secuencias, paralelismos e iteraciones. La vista informacional se centra en la estructura de los artefactos producidos o consumidos por las actividades, y en sus interrelaciones. Por último, la perspectiva organizacional, tiene como fin mostrar dónde y quiénes son los agentes (en cumplimiento de roles) que intervienen en la realización de las actividades. Un proceso que no presente sus actividades claramente modeladas es difícil de comprender, debido a que puede generar cierta ambigüedad en las descripciones de las actividades y también es difícil de comunicar.

La primera contribución de este trabajo está relacionada con la debilidad detectada en la falta de una especificación de un proceso robusto y formal que guíe las actividades sugeridas en el enfoque DSR. Para este objetivo, se propone una especificación del proceso de DSR utilizando para tal fin el lenguaje SPEM [15], y haciendo uso de las perspectivas de modelado funcional y de comportamiento propuestas en [14]. La segunda contribución es la ilustración de la aplicabilidad del proceso de DSR propuesto para la construcción de una ontología de testing de software denominada TestTDO que constituye el artefacto producido. La conceptualización de esta ontología debía estar semánticamente integrada al marco conceptual C-INCAMI v.2.0.

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza
ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

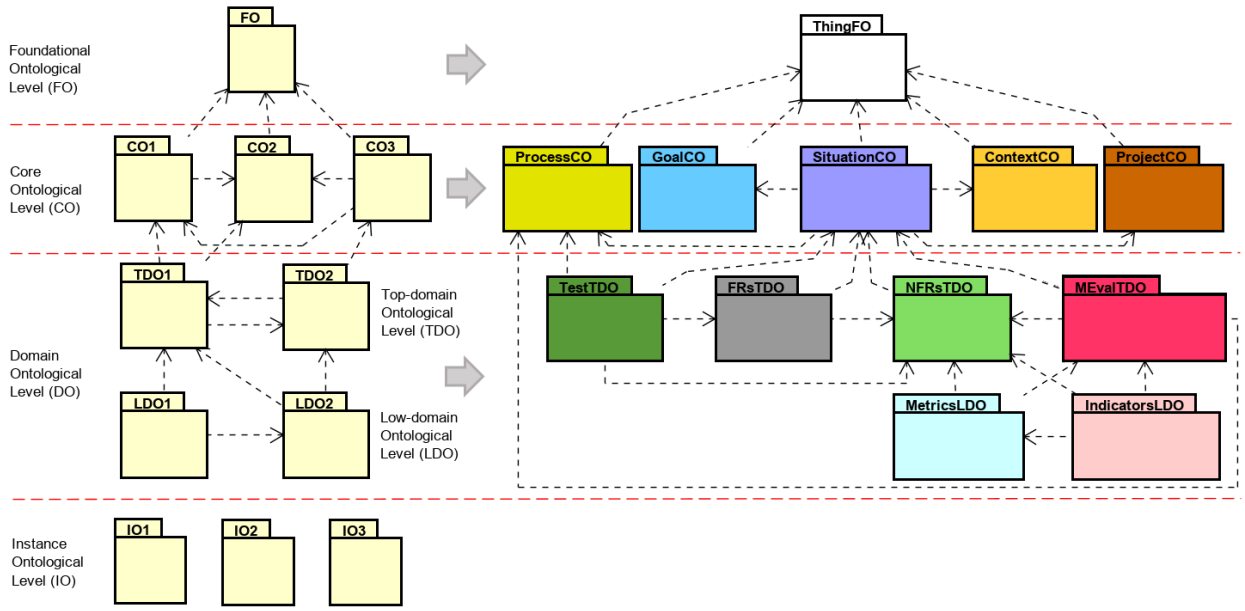

Fig. 2. Arquitectura ontológica de 4 capas la cual incluye niveles Fundacional, Core, de Dominio e Instancia. Los componentes conceptuales de la Fig. 1 fueron ubicados y armonizados según el nivel ontológico correspondiente. Ademas, se agregaron nuevos componentes como SituationCO. Notar que el acrónimo MEval significa en inglés Measurement and Evaluation.

Este artículo invitado a la presente revista digital es una actualización de los contenidos publicados en el evento CoNaIISI'19 [16]. Por lo tanto, respecto al artefacto desarrollado, ya existen al presente dos versiones. TestTDO v1.0 que representa la primera versión de la ontología de testing de software cuya conceptualización fue concluida a fines de Setiembre de 2019, y publicada en Tebes et al. [17]. La siguiente versión (TestTDO v1.1) fue concluida tanto la actualización de su conceptualización como su implementación en el lenguaje OWL a fines de Marzo de 2020 (`https://doi.org/10.13140/RG.2.2.26338.07368/1`).

Es importante remarcar que ambas versiones de la ontología TestTDO fueron integradas a una arquitectura ontológica de 4 capas la cual incluye niveles Fundacional, Core, de Dominio e Instancia, tal como se muestra en la parte izquierda de la Fig. 2. Por lo tanto, surgió el nuevo requerimiento de integrar y armonizar semánticamente el artefacto TestTDO a esta arquitectura y sus ontologías antes que a los componentes conceptuales de C-INCAMI v2.0. Esta arquitectura denominada FCD-OntoArch (*Foundational, Core, and Domain Ontological Architecture for Sciences*) es una evolución de C-INCAMI v2.0. Por lo tanto, todos los componentes conceptuales mostrados en la Fig. 1, se encuentran renombrados en la parte derecha de la Fig. 2, en donde además se encuentran algunos nuevos, como SituationCO y ThingFO. Sin embargo, está fuera del alcance de este articulo discutir aspectos de dicha arquitectura y de la armonización semántica de sus componentes ya sean los nuevos o los preexistentes en C-INCAMI v2.0.

El resto del artículo se estructura de la siguiente manera. A seguir, la Sección II aborda trabajos relacionados referidos a procesos definidos para el enfoque DSR. La Sección III documenta la especificación del proceso de DSR propuesto en este trabajo, enfatizando a la perspectiva funcional y a la de comportamiento. En tanto se discute dicho proceso, se lo ilustra principalmente con la construcción del artefacto TestTDO v1.0 con

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

referencias a la última versión. Por último, la Sección IV presenta las conclusiones y líneas de avance futuro.

II. TRABAJOS RELACIONADOS

En Offermann et al. [12], los autores presentan un proceso para DSR estructurado en tres fases principales: 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 pre-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 pre-relevancia se lleva a cabo interrogando distintos investigadores a fin de determinar si acuerdan con las hipótesis planteadas para el problema detectado.

En el trabajo arriba citado, 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, en [12], 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, dichos autores sugieren utilizar la técnica de entrevistas con expertos para demostrar que hay interés general en la solución. La entrevista puede tener la forma de "¿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, artículos de revistas y/o eventos científicos. De esta manera, se espera una retroalimentación de la comunidad científica interesada sobre los resultados obtenidos y expuestos. El trabajo de dichos autores es relevante y las fases han sido consideradas para la construcción del proceso propuesto en el presente artículo. Sin embargo, el proceso en [12] no está modelizado formalmente sino más bien presentado en forma de esquema sin notación definida y estandarizada, y sin tener en cuenta productos de trabajo producidos y consumidos por las distintas actividades. Además, solo considera la perspectiva de modelado de proceso de comportamiento.

Otros trabajos de interés son los de Hevner et al. [11], [18], 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 Información (SI). Además, el mismo contiene siete guías que ayudan a entender los requisitos para utilizar de manera efectiva el enfoque de 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 como casos de estudio, testing, simulaciones, entre otros. 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 [12], en [11], [18] tampoco se presenta un modelo de proceso que indique formalmente las actividades y tareas a desarrollar, teniendo en consideración estas guías. Menos aún hace mención de los productos de trabajo que se consumen y producen a lo largo del proceso de DSR, tal como se ilustra en la Sección III del presente trabajo.

Peffers et al. [13] presentan lo que denominan un modelo de proceso para DSR, pero no formalizado en un lenguaje de modelado estandarizado, sino simplemente con una figura de cuadros, que ilustra la secuencia de actividades (es decir, la vista de comportamiento) que proponen seguir para implementar el enfoque de investigación. El trabajo es también relevante y sirvió como material para formalizar el proceso que se propone en este artículo. Las actividades planteadas en [13] 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 el proceso con la actividad de comunicar los resultados para diseminar el conocimiento obtenido a la comunidad interesada.

Vaishnavi et al. [19] describen un modelo de proceso genérico que siguen 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 luego dar 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 analizados previamente, realiza más bien la descripción de las tareas enmarcadas dentro de "pasos del proceso" (como los autores prefieren denominar) pero carece de una especificación formal, tal como se representa en nuestra propuesta. Es importante destacar que en [19] los autores tuvieron en cuenta las vistas de comportamiento y funcional, al igual que en el presente artículo. No obstante, dentro de la perspectiva funcional, solo se ilustran los artefactos producidos por las

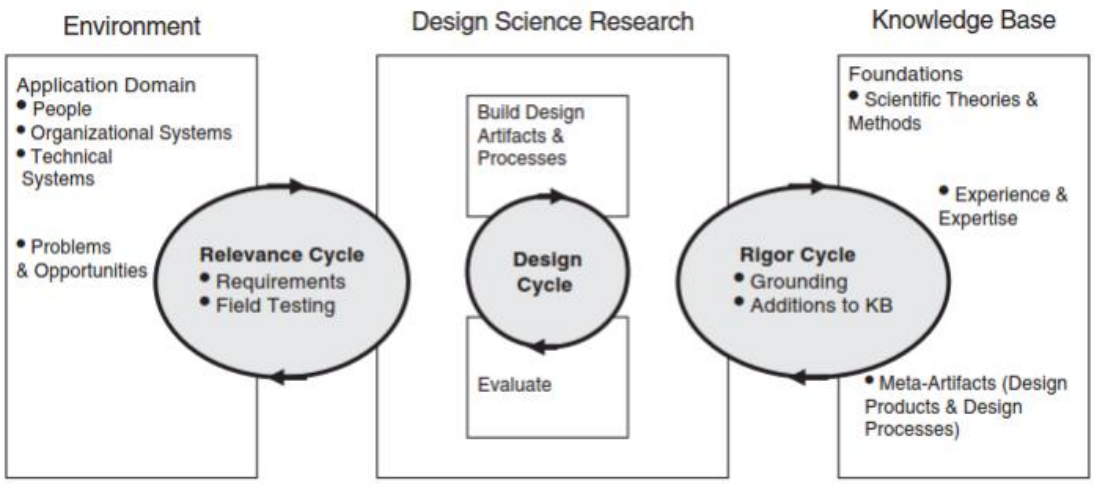

Fig. 3. Ciclos en DSR (extraído de Hevner [18]).

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

actividades, sin modelar las entradas a las mismas.

Finalmente, varios trabajos coinciden en argumentar la falta de un proceso estandarizado para implementar el enfoque DSR en SI, como por ejemplo en [7], [9], y [10].

III. ESPECIFICACIÓN DEL PROCESO DE DSR PROPUESTO

De acuerdo a Hevner et al. [18], la aplicación del enfoque DSR en SI consiste de tres ciclos de investigación, denominados ciclo de Relevancia, de Diseño y de Rigor, tal como se ilustran en la Fig. 3.

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 identifica qué proceso de diseño, métodos o heurísticas se utilizarán para construir el artefacto, como así también cuáles evaluaciones se ejecutan para evaluar el diseño. Además 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 modelos reconocidos, además de procesos y métodos bien establecidos. En caso de producirse un artefacto útil para resolver el problema en cuestión, y que luego de comunicado sea aceptado como tal, podrá ser incluido entonces a la base de conocimiento para contribuir al acervo de la comunidad científica y profesional.

Considerando estos ciclos además de los trabajos relacionados discutidos en la Sección II, la Fig. 4 muestra el modelo de proceso para DSR que se propone en este trabajo, utilizando las perspectivas de proceso funcional y de comportamiento.

Las actividades que se corresponden con el ciclo de Relevancia de Hevner et al. están agrupadas en la actividad que se denomina "A1 Identificar Problema/Solución". Por su parte, las actividades comprendidas en el ciclo de Diseño, corresponden a 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 A1, A2 y A3 de la Fig. 4, aunque también se relacionan con la difusión y diseminación del conocimiento obtenido como producto de la investigación. Esto se corresponde a la actividad "A4 Comunicar la Investigación".

Las sub-secciones siguientes describen cada una de las 4 actividades del proceso de DSR propuesto. Además, se ejemplifica y documenta cómo se llevaron a cabo estas actividades para la construcción de una ontología de testing de software, aspecto que se considera como un caso aplicado del proceso propuesto en este trabajo.

A. Identificar Problema/Solución (A1)

De acuerdo a la Fig. 4, la primera actividad "A1 Identificar Problema/Solución" comprende las siguientes sub-actividades: i) Definir Problema/Solución, ii) Investigar soluciones actuales, y iii) Especificar requisitos de diseño. Para la primera sub-actividad ("Definir Problema/Solución") se incluyen las tareas "Definir el objetivo/problema de negocio" y "Definir la solución". Para definir el problema se requiere de las metas de necesidad de información organizacional, es

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza
ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

decir, de la comprensión 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".

Fig. 2 muestra dicha arquitectura, y al componente TestTDO, el cual será el artefacto a construir o adoptar. La Fig. 5 ilustra el documento "Definición del problema de negocio" siguiendo a la plantilla de especificación de

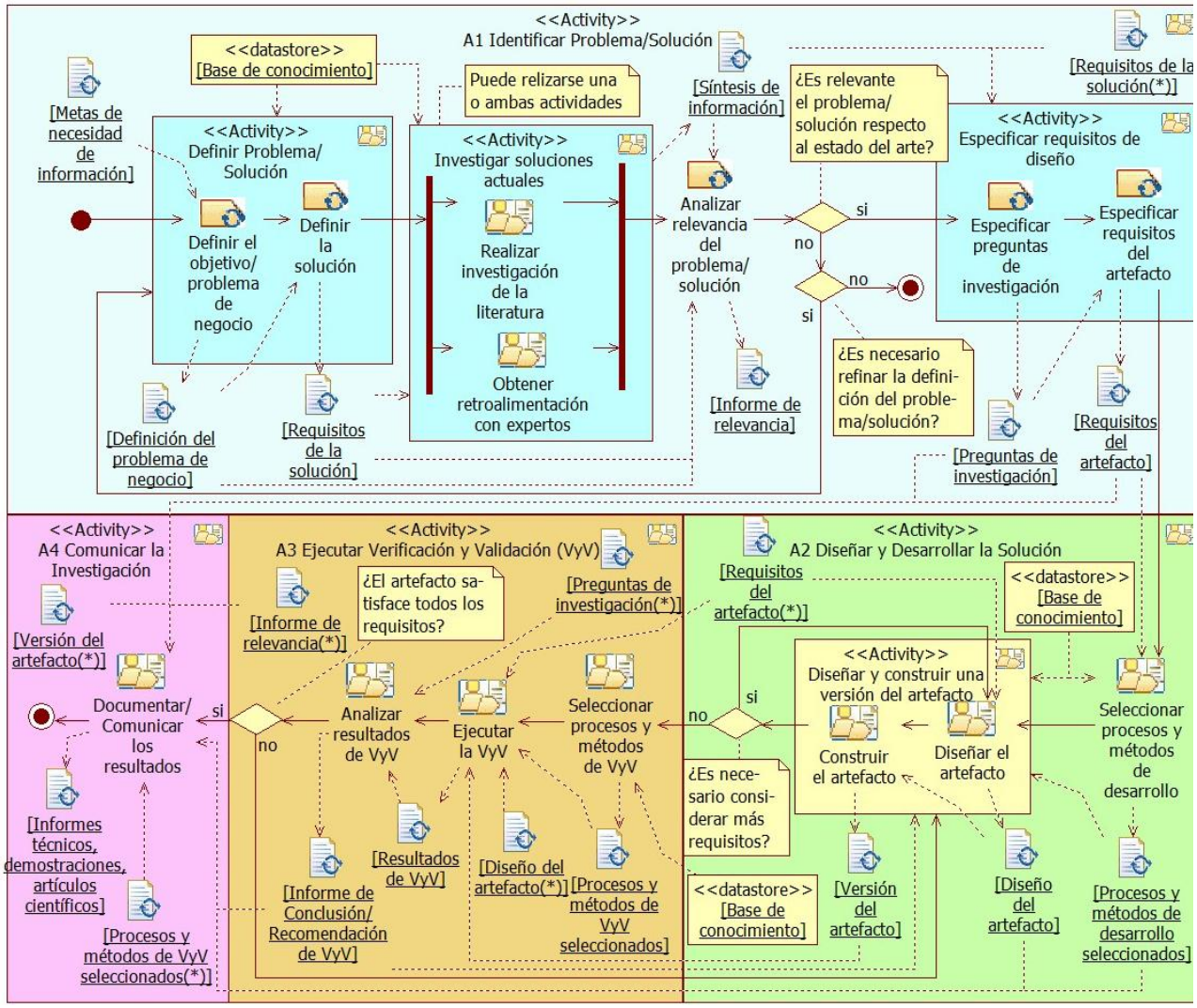

Fig. 4. Perspectiva funcional y de comportamiento del proceso de DSR propuesto, especificado en SPEM. Notar que aquellos nombres de artefactos terminados en "(*)" indican que están replicados para mejorar la legibilidad del modelo.

Para el caso aplicado, la entrada relacionada a la meta de necesidad de información es la siguiente: *"el grupo de investigación GIDIS_Web requiere una conceptualización ontológica del dominio de testing de software para ser integrada a la arquitectura FCD-OntoArch y que se relacione con los componentes de requisitos funcionales, requisitos no funcionales"*. La

Definición del Problema de Negocio
***Obtener*:** una conceptualización rigurosa del dominio de testing.
***Por medio de*:** la estructuración del dominio como una ontología.
***Tal que*:** la ontología satisfaga los siguientes requisitos: i) relacionarse con conceptos provenientes de los componentes de requisitos funcionales (FRsTDO) y requisitos no funcionales (NFRsTDO) en el contexto de la arquitectura FCD-OntoArch (Fig. 2); ii) que sea de dominio de alto nivel.
***De manera de lograr*:** soporte para la especificación de una familia de estrategias integradas destinadas a satisfacer metas con propósitos de testing.

Fig. 5. Artefacto "Definición del problema de negocio".

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

un problema, propuesto por Wieringa [20].

Con el problema especificado se procede a definir la solución esperada 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. Para el caso aplicado se recuperaron de la base de conocimiento los documentos referenciados en [21]-[25].

Tras revisar estos documentos y en consideración del problema establecido, se generaron los requisitos de la solución, estableciendo para cada uno de ellos una prioridad de acuerdo a las necesidades que requiere la ontología a adoptar o construir. Como resultado, los requisitos con prioridad alta que se identificaron fueron los siguientes:

- La ontología debe estar conceptualizada y no necesariamente implementada.
- La ontología debe contener conceptos de testing que se puedan vincular con conceptos de requisitos funcionales y requisitos no funcionales.
- La ontología debe tener una cobertura amplia del dominio de testing de software, es decir, debe contener conceptos relacionados a testing dinámico y estático como así también a testing funcional y no funcional.
- La ontología debe considerar relaciones tanto taxonómicas (es-un y todo-parte) como no taxonómicas.
- La ontología debe estar desarrollada siguiendo alguna metodología rigurosa y/o formal.

Por otra parte, como requisitos con prioridad media, se identificaron los siguientes:

- La conceptualización de la ontología debe estar representada gráficamente en algún lenguaje de modelado para estos propósitos.
- La ontología debe estar desarrollada considerando estándares internacionales y/u otras ontologías, taxonomías o glosarios de testing de software.
- La ontología debiera ser de dominio de alto nivel (Top-Domain Ontological level –TDO- en Fig. 2).
- Los términos de la ontología de testing de software deberían estar enriquecidos semánticamente por términos de ontologías de más alto nivel (core y fundacional).

De acuerdo a la Fig. 4, la segunda sub-actividad en “A1 Identificar Problema/Solución” se denomina “Investigar soluciones actuales” y consiste en otras dos sub-actividades: “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”.

Síntesis de Información
No existe una ontología que cumpla todos los requisitos propuestos de la solución. Dentro de las que tienen mejor cobertura se encuentran ROoST [23] y la propuesta de Asman *et al*. [21], pero no están vinculadas directamente con conceptos de requisitos funcionales y no funcionales. Además, ROoST no tiene en cuenta conceptos de testing estático.

Lo fundamental en este punto es hacer un análisis sistemático del estado del arte de la situación actual respecto al problema/solución planteada y, en caso de ser necesario, consultar con expertos del dominio. Para la primera de las sub-actividades se presenta en Becker et al. [26] un proceso que guía las actividades implicadas en una Revisión Sistemática de Literatura (RSL). En el presente estudio se utilizó dicho proceso de RSL, el cual es considerado como el artefacto de entrada desde la base de conocimiento para llevar a cabo la actividad "Realizar investigación de la literatura". Como salida se generó el documento "Síntesis de información" que se muestra en la Fig. 6. En dicha figura se muestra un extracto de la síntesis que se encuentra documentada de forma completa en Tebes et al. [6]. Es decir, en dicho trabajo se documenta y analiza la RSL en donde se seleccionaron y evaluaron 12 conceptualizaciones ontológicas para testing de software. La evaluación se basó en características y atributos expresados en los requisitos de la solución y en algunas prácticas de calidad ontológica propuestas por D'Aquin et al. [22]. Los requisitos de calidad y resultados de la evaluación se pueden acceder de un modo directo en `http://bit.ly/OntoQualityEval`.

Por último, cabe aclarar que en el presente caso aplicado no se realizó la otra sub-actividad, a saber; "Obtener retroalimentación con expertos".

Siguiendo con el proceso de la Fig. 4, se debe ejecutar la tarea "Analizar 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 (o no) mediante el enfoque DSR.

Informe de Relevancia
Se decidió que el problema/solución es relevante para ser abordado por DSR por dos razones: 1) no existe una ontología que satisfaga todos los requisitos propuestos para la solución, 2) construir/adaptar una ontología que satisfaga todos los requisitos propuestos no es rutinario, por el contrario, es una tarea compleja de llevar a cabo, por los siguientes motivos: i) se deben considerar diferentes fuentes de las definiciones de la terminología del dominio de testing (como estándares internacionales, otras ontologías, etc.) para poder obtener mayor cobertura que las soluciones existentes; ii) se debe tener en cuenta en el diseño que la ontología a desarrollar se pueda vincular directamente con conceptos de requisitos funcionales y requisitos no funcionales (y reusar o extender términos de otologías de más alto nivel) para beneficiar a las estrategias de testing a desarrollar; y iii) sería deseable diseñar la ontología de dominio de alto nivel (TDO) ya que una ontología de este tipo favorece el desarrollo de ontologías de dominio de más bajo nivel y promueve la existencia de un vocabulario común de referencia para el dominio de testing de software.

Fig. 7. Artefacto "Informe de relevancia".

La Fig. 7 ilustra dicho documento producido para el caso ilustrado en este trabajo. 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 sub-actividades de A1. En contrapartida, si es relevante, se continua con la sub-actividad "Especificar requisitos de diseño".

Para este fin, 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 (PI) de las cuales se derivan lo requisitos del artefacto (RA) a construir servirán posteriormente para evaluar, validar y verificar el artefacto construido. La clara formulación de las PI y RA representa una actividad fundamental en cualquier estudio de investigación debido a que direccionan la investigación y transmiten su esencia, según Thuan et al. [27].

Para el caso aplicado en este artículo, las preguntas principales de investigación formuladas fueron las siguientes tres, con sus respectivas sub-preguntas:

- **PI#1:** ¿Qué requisitos para la ontología de testing de software son los más apropiados para dar soporte a estrategias que ayuden a alcanzar metas de testing? A partir de esta pregunta se derivaron las siguientes 5 sub-preguntas de investigación:
  - *PI#1.1:* ¿La ontología a desarrollar debería ser de dominio de alto nivel (TDO), o de dominio de bajo nivel (LDO)?
  - *PI#1.2:* ¿Cuáles son las terminologías documentadas (estructuradas en glosarios, taxonomías u ontologías) más robustas y ricas para el dominio de testing de software?
  - *PI#1.3:* ¿Cuáles son los principales términos que debería tener la ontología de testing de software para alcanzar una buena cobertura?
  - *PI#1.4:* ¿La ontología a desarrollar debería estar a nivel de conceptualización y/o implementación?
  - *PI#1.5:* ¿Cuál es la metodología de desarrollo de ontologías más adecuada?
- **PI#2:** ¿Cómo integrar la ontología de testing con los componentes conceptuales (sub-ontologías) y niveles existentes del marco ontológico FCD-OntoArch? A raíz de ella surgieron 2 sub-preguntas de investigación:
  - *PI#2.1:* ¿Con qué sub-ontologías y niveles de FCD-OntoArch se debe relacionar la nueva ontología de testing de forma directa?
  - *PI#2.2:* ¿Qué términos son necesarios para relacionar e integrar las diferentes sub-ontologías de FCD-OntoArch con la nueva ontología de testing de software?
- **PI#3:** ¿Cómo asegurar la calidad de la ontología de testing de software? Esta pregunta de investigación también se derivó en 2 sub-preguntas, a saber:
  - *PI#3.1:* ¿Qué características de calidad se deberían evaluar de la ontología?
  - *PI#3.2:* ¿Qué estrategias de verificación y validación son las más adecuadas para evaluar la ontología de testing de software?

Después de llevar a cabo la tarea "Especificar requisitos del artefacto" (ver Fig. 4) utilizando como entrada a las PI antes formuladas, produjimos el siguiente documento con los 10 requisitos del artefacto (RA), a saber:

- RA#1 –relacionado con PI#1.1. Diseñar y construir una ontología de dominio de alto nivel.
- RA#2 –relacionado con PI#1.2. Considerar i) los estándares internacionales ISTQB [25], ISO 29119 (partes 1, 2, 3 y 4) [24], y ii) la ontología ROoST [23] y la ontología de dominio de alto nivel descripta por Asman et al. [21] las cuales fueron los dos mejores ranqueadas entre las 12 ontologías seleccionadas y evaluadas en la RSL [6].
- RA#3 –relacionado con PI#1.3. La cobertura terminológica de la ontología de dominio de alto nivel debe ser la necesaria y suficiente para ser

extendida por ontologías de dominio de más bajo nivel. Para ello, debe considerar términos relacionados a testing estático y dinámico, como así también a testing funcional y no funcional. Además, se deben tener en cuenta conceptos de definiciones de trabajo de testing (proceso de trabajo, actividad y tarea), productos de trabajo de testing (artefacto, resultado), métodos, agentes, entidades de testing, meta y proyecto de test, y estrategia de testing los cuales serán enriquecidos semánticamente con conceptos de otras ontologías a nivel core y fundacional. Notar que, en particular, para el artefacto ontología, aspectos del alcance se especifican por medio de Preguntas de Competencia (PC). Las 25 PC resultantes se pueden acceder en `http://bit.ly/TestTDO-CQuestions`.

- RA#4 –relacionado con PI#1.3. Debe haber un balance en la cobertura de relaciones taxonómicas y no taxonómicas.
- RA#5 –relacionado con PI#1.4. La ontología debe estar desarrollada a nivel de conceptualización, y no necesariamente a nivel de implementación en un lenguaje formal, dado que el principal objetivo es enriquecer a especificaciones de procesos y métodos de las estrategias de testing.
- RA#6 –relacionado con PI#1.5. Utilizar principalmente Methontology [28] –y algunos aspectos de Ontology 101 [29]- para la construcción de la ontología hasta su fase de conceptualización.
- RA#7 –relacionado con PI#2.1. Relacionar los términos de la ontología de testing directamente con conceptos de las ontologías de requisitos funcionales (FRsTDO) y requisitos no funcionales (NFRsTDO). Adicionalmente, se deberían relacionar términos de la ontología de testing de software con otros componentes conceptuales de nivel superior como el de proceso (ProcessCO) y el de situación (SituationCO), tal cual se observa en la Fig. 2. Notar que a su vez SituationCO se relaciona con otros componentes a nivel ontológico core.
- RA#8 –relacionado con PI#2.2. Los términos necesarios para relacionar e integrar las diferentes ontologías del marco FCD-OntoArch (ver Fig. 2) con la ontología de testing de software son con el: i) componente de requisitos no funcionales (NFRsTDO): el término requisito no funcional; ii) componente de requisitos funcionales (FRsTDO): el término requisito funcional; iii) componente meta de negocio (GoalCO): los términos meta de negocio y meta de necesidad de información; iv) componente de proyecto (ProjectCO): proyecto, estrategia y plan de proyecto; v) componente contexto (ContextCO): ente de contexto; vi) componente proceso (ProcessCO): proceso de trabajo, actividad, producto de trabajo, método, rol y agente; y vii) componente situación (SituationCO): los términos situación particular y entidad target (entidad evaluable y entidad desarrollable).
- RA#9 –relacionado con PI#3.1. Las características a evaluar de la ontología de testing de software son: calidad estructural ontológica, disponibilidad balanceada de relaciones, cobertura terminológica de testing estático y

dinámico (funcional y no-funcional), adherencia con otros vocabularios.

- RA#10 –relacionado con PI#3.2. La estrategia para evaluar la ontología resultante es GOCAME (*Goal-Oriented Context-Aware Measurement and Evaluation*) [1]. Además, se verificará las PC con una matriz de correspondencia compuesta de términos, propiedades, relaciones y axiomas, y se la validará aplicando los conceptos de la ontología a un proyecto de testing de tipo académico, además de examinarla con otros grupos de expertos del dominio para obtener retroalimentación. En caso de que la ontología sea implementada (opcional) se podrá verificar dinámicamente.

B. Diseñar y Desarrollar la Solución (A2)

De acuerdo a la Fig. 4 que ilustra el proceso propuesto de DSR, la primera actividad que da inicio al diseño y construcción de la solución (rectángulo verde) se denomina "Seleccionar procesos y métodos de desarrollo". Tiene como entrada el documento de "Requisitos del artefacto" y la base de conocimiento. Para el caso aplicado, y según el RA#6 antes enunciado, se recuperaron dos documentos de la base de conocimiento que presentan métodos y procesos para desarrollar ontologías, referenciados en Fernández-López et al. [28] y Noy et al. [29]. Como salida se genera un artefacto con la especificación de los procesos y métodos de desarrollo seleccionados.

A continuación, sigue la sub-actividad "Diseñar y construir una versión del Artefacto" que se compone de las sub-actividades "Diseñar el artefacto" y "Construir el arte-facto". 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 bien, para la primera de las sub-actividades ("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 sub-actividad siguiente, "Construir el artefacto".

Como indicado previamente, el diseño de la ontología de testing se realizó siguiendo las etapas que propone Methontology [28]. Methontology fue la metodología principal elegida para diseñar la ontología, y realizar esencialmente hasta la etapa de conceptualización que dicha metodología propone. Por lo tanto, de acuerdo a la misma, se llevaron a cabo los siguientes pasos:

1. Etapa de Especificación: Methontology propone en esta etapa delimitar el objetivo y alcance de la ontología a construir. Para el caso aplicado que aquí se ilustra se distinguen dos objetivos. Por un lado, el desarrollo de la ontología de testing tiene como propósito principal servir como base o dar soporte a la especificación de una familia de estrategias de testing, tal como indicado en la Sección I. Para ello, debe poder cubrir conceptos que involucren tanto procesos como métodos de testing. Por otro, se requiere que la ontología pueda agilizar la comprensión del dominio de testing para futuros profesionales de pruebas de software (testers), brindándoles explícitamente la terminología compactada y resumida de diferentes fuentes para su estudio. Respecto al alcance de la ontología (RA#3), debido a que debe proveer un soporte terminológico de

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza
ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

procesos y métodos de testing, la ontología debe cubrir principalmente conceptos de actividades, productos de trabajo (entradas y salidas), roles y métodos de testing, y que los mismos sean de alto nivel. A su vez, la ontología debe abarcar conceptos de testing estático y dinámico, niveles de testing y también de testing funcional y no funcional. Por último, se debe tener en cuenta que se debe relacionar con conceptos de requisitos funcionales y no funcionales. Como fuentes de conocimiento se consideran ISTQB [25], ISO 29119 (partes 1, 2, 3 y 4) [24] y todas las ontologías obtenidas en la revisión sistemática de literatura (documentadas en [6]).

2. Etapa de Conceptualización: es importante remarcar que esta sub-actividad se realizó varias veces y produjo 2 versiones principales. Con respecto a Ontology 101 [29], esta metodología se utilizó en conjunto con la etapa de conceptualización de Methontology para obtener la primera versión de la ontología. Para la primera versión estable de la ontología (TestTDO v1.0), se realizaron varios ciclos de diseño-construcción de la misma, logrando como artefactos las tablas de definiciones de términos, propiedades, relaciones y axiomas involucrados, además del diagrama ontológico documentados en [17]. Luego de esta versión se produjo una

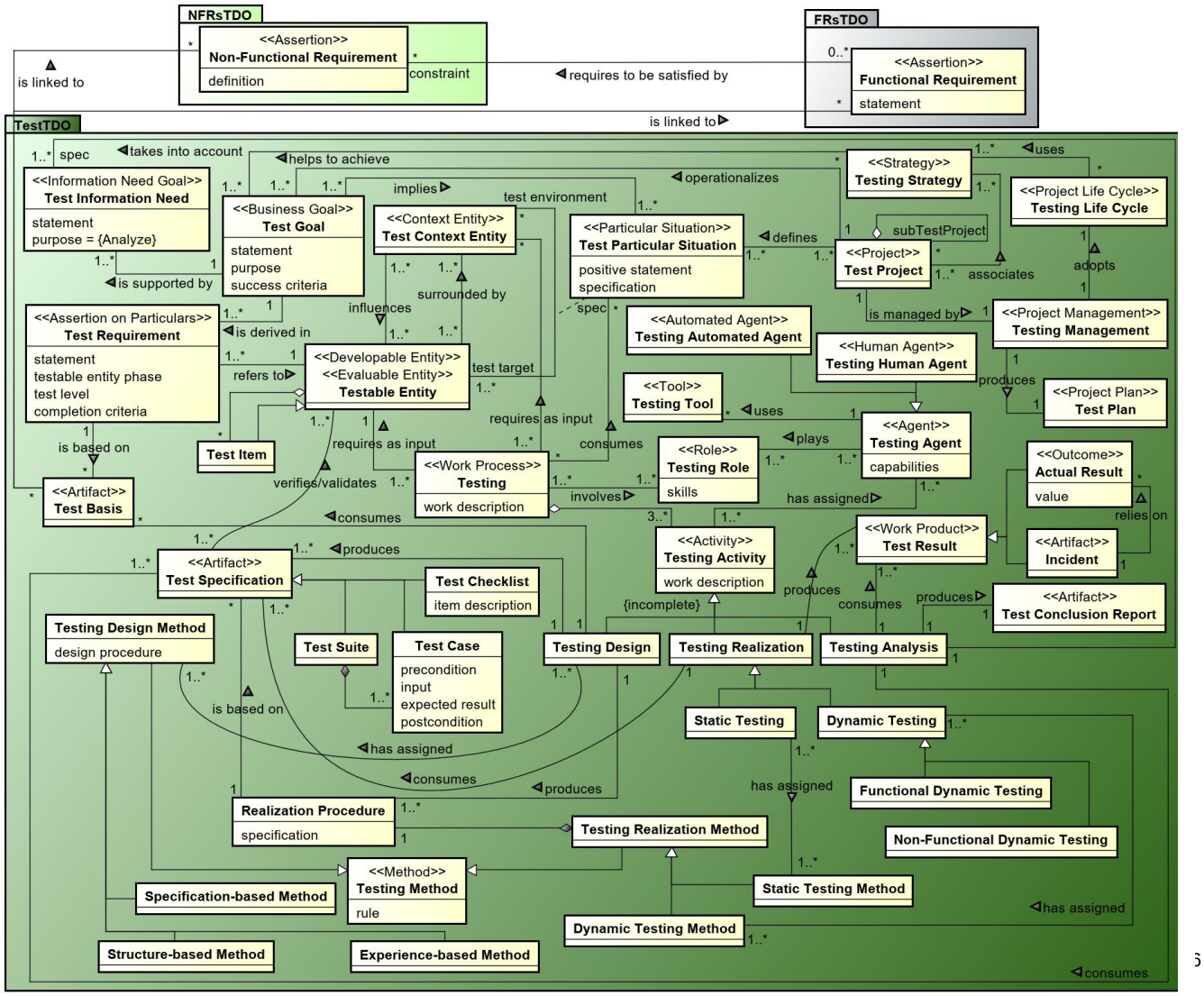

Fig. 8. Main terms, properties and relationships of the TestTDO v1.1 ontology and its relation with Non-Functional Requirement and Functional Requirement terms.

nueva versión denominada TestTDO v1.1 cuyo diagrama ontológico resultante se muestra en la Fig. 8.

En definitiva, respecto de A2, con el diseño del artefacto se inicia su construcción, obteniendo como salida una versión del mismo. Si es necesario re-definir el diseño ya que se deben considerar más requisitos, entonces se itera otra vez a la sub-actividad de diseño del artefacto. Caso contrario, el proceso continúa con la actividad A3, la cual se detalla en la siguiente sub-sección. Cabe aclarar que, aunque no se ilustre en el proceso de la Fig. 4 para evitar sobrecargarlo, si se volviera a realizar la actividad de diseñar el artefacto también tendría como entrada alguna versión del diseño para poder re-definirlo.

Por último, aunque no menos importante, respecto de la sub-actividad "Construir el artefacto", para TestTDO v1.0 solo se realizó hasta la Etapa de Conceptualización, conforme a los RA establecidos inicialmente. Sin embargo, para TestTDO v1.1 se realizó además la Etapa de Implementación según Methontology, produciendo como artefacto su código OWL. Esto permitió realizar una verificación adicional de la ontología.

C. Ejecutar Verificación y Validación (A3)

El proceso de DSR continúa con la tercera actividad denominada "A3 Ejecutar Verificación y Validación (VyV)". De acuerdo a la Fig. 4, la primera sub-actividad que comprende A3 se denomina "Seleccionar procesos y métodos de VyV" que produce como salida el documento "Procesos y métodos de VyV seleccionados". Por lo que para generar este documento se utiliza a la base de conocimiento como entrada.

Ejemplos de procesos y/o métodos de VyV pueden ser evaluación mediante métricas e indicadores específicos, métodos de testing de caja blanca o de caja negra (ya sea para testing estático o dinámico), métodos de simulación y chequeo de modelos, entre varios otros métodos y técnicas. Para verificar y validar el alcance de una ontología comúnmente se utiliza como base de pruebas a las preguntas de competencia (PC), referenciadas previamente en el RA#3.

La siguiente sub-actividad a realizar es "Ejecutar la VyV", que requiere como entrada a 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 (la conceptualización y/o implementación de la ontología en nuestro caso) como su diseño contra los requisitos establecidos. Con los procesos y/o métodos seleccionados 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 sub-actividad "Analizar resultados de VyV" se corroboran los mismos contra lo establecido en las preguntas de investigación –y sus requisitos de artefacto derivados-, puesto que estas deben ser respondidas en la mayor totalidad posible para asegurarse de que el artefacto va a tener el alcance y 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 actividad de diseño y construcción (ver Fig. 4, "A2 Diseñar y Desarrollar la Solución") debido a que el artefacto puede no satisfacer todos los requisitos. En caso contrario, se puede avanzar hacia la última actividad (A4) del proceso de DSR. De aquí la importancia de establecer las preguntas de investigación y requisitos con el fin de iterar tanto cuanto sea necesario hasta asegurarse que se cubren todos los aspectos propuestos en el diseño y construcción del artefacto.

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza
ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

Como indicado en la Sección I, respecto al artefacto desarrollado, ya existen al momento presente dos versiones. TestTDO v1.0 que representa la primera versión final de la ontología de testing de software cuya conceptualización fue concluida a fines de Setiembre de 2019, y publicada en Tebes et al. [17]. En este artículo se presentan tres métodos de VyV, de modo que remitimos al lector al mismo.

A modo de ejemplo, la Tabla 1 muestra un fragmento de la matriz de verificación de correspondencia de cobertura solo para una pregunta de competencia (la PC #18). Las 25 PC establecidas se chequean contra los Términos, Propiedades, Relaciones y Axiomas desarrollados de TestTDO v1.0. En caso en que una PC no se encuentre cubierta, se deberá retornar a la actividad de diseño y construcción de A2, para cubrir el alcance requerido.

La siguiente versión de la ontología, TestTDO v1.1, fue concluida a fines de Marzo de 2020 tanto la actualización de su conceptualización como la realización de su implementación –usando el lenguaje OWL y el entorno Protégé. Para esta versión se aplicó un nuevo método de verificación (método de testing dinámico funcional) sobre el código en ejecución y empleando consultas (queries) SPARQL. Detalles de este método están documentados en otro artículo de revista, recientemente sometido a revisión.

D. Comunicar la Investigación (A4)

La última actividad del proceso es "A4 Comunicar la Investigación", según se observa en la Fig. 4. La sub-actividad que comprende se denomina "Documentar/Comunicar los resultados" y recibe como entrada siete artefactos, a saber: 1) "Informe de Conclusión/Recomendación de VyV"; 2) Versión final del artefacto generado; 3) "Informe de relevancia"; 4) Preguntas de investigación (PI); 5) Requisitos del artefacto, que incluye a las PC; 6) Procesos y métodos (de desarrollo y de VyV) seleccionados; y 7) Diseño del artefacto. Haciendo uso de todos estos artefactos principales, se produce como salida reportes técnicos, científicos, tesinas o tesis, que permitan documentar la investigación realizada en el desarrollo del artefacto. Esto permitirá constituir un acervo de investigación que contribuya a la comunidad interesada.

Para el caso aplicado en el presente artículo, el ámbito de incumbencia es la comunidad dedicada al aseguramiento de calidad del software y desarrollo de ontologías, particularmente, a la comunidad de ontologías de testing de software. Si el impacto en la comunidad científica relacionada es positivo y de interés, es decir, el artefacto es ampliamente adoptado debido a su utilidad, relevancia y carácter de innovador, entonces todo el material producido y almacenado con los resultados del artefacto, pueden pasar a integrar la base de conocimiento, y ser

TABLA 1. FRAGMENTO DEL TEST CHECKLIST PARA INSPECCIONAR COMO LAS PREGUNTAS DE COMPETENCIA (PC O CQS EN INGLES) ESTAN CUBIERTAS POR LOS CONCEPTOS DE TESTTDO 1.0, ESTO ES, POR SUS TERMINOS, PROPIEDADES, RELACIONES, Y AXIOMAS

CQ	What are the Terms, relationships, *properties* and Axioms of TestTDO that answer the corresponding CQ?	Actual Result (pass/fail)	Incident description (if it fails)
CQ18. *In which particular situation is a testable entity considered an evaluable entity?*	Test Requirement refers to Testable Entity, Test Requirement is based on Test Basis, Test Basis is linked to Non-Functional Requirement. Axiom #5 (*)	Pass	

(*) Axiom #5: Any Testable Entity is an Evaluable Entity iff the Test Requirement that refers to this Thing is linked to a Non-Functional Requirement. That is, $\forall te: TestableEntity(te) \wedge EvaluableEntity(te) \leftrightarrow \exists tr, tb, nfr: TestRequirement(tr) \wedge TestBasis(tb) \wedge NonFunctionalRequirement(nfr) \wedge refersTo(tr, te) \wedge isBasedOn(tr, tb) \wedge isLinkedTo(tb, nfr)$

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)

empleado para el diseño y construcción de otros artefactos con el enfoque DSR. De esta manera, queda disponible para futuras implementaciones del proceso para la construcción de otros artefactos.

Al presente, como resultado de A4, ya se encuentra publicado un artículo científico (Tebes et al. [17]) en evento iberoamericano de Ingeniería de Software, el cual documenta en el idioma inglés, a TestTDO v1.0 y todos sus artefactos que lo integran. Además, una extensión del mismo, con las actualizaciones conceptuales e implementación de la ontología TestTDO v1.1, están documentados en artículo de revista internacional, recientemente sometido a revisión.

Varios de los documentos de la última versión se pueden acceder públicamente en ResearchGate. Por ejemplo, las definiciones de todos los términos, propiedades, relaciones y las especificaciones de los 12 axiomas de TestTDO v1.1 se encuentran como reporte técnico en `https://doi.org/10.13140/RG.2.2.26338.07368/1`, y su implementación en `http://bit.ly/TestTDO-OWL`.

IV. CONCLUSIONES Y TRABAJO FUTURO

Las ciencias de la computación, en particular la disciplina de SI, han adoptado el enfoque de investigación de DSR para el diseño y construcción de artefactos. Sin embargo, un mero análisis de la literatura al respecto permite observar que las actividades que se proponen seguir para el enfoque no están modelizadas formalmente bajo un proceso bien especificado que guíe y establezca entradas, salidas e iteraciones entre ellas.

Con el objetivo de aportar en este aspecto, el presente trabajo describe un modelado de proceso para DSR considerando a la perspectiva funcional y a la de comportamiento, las cuales fueron especificadas siguiendo la notación SPEM. Esto permite, de manera bien establecida y sistemática, llevar a cabo las actividades y tareas implicadas en el enfoque DSR. Tener modelizado el proceso de DSR facilita el entendimiento y la comunicación del mismo para la comunidad interesada, redundando en consistencia y repetitividad en la ejecución de las actividades y tareas que lo componen.

Con el fin de ilustrar aspectos de dicho proceso se empleó, en conjunto con la descripción del mismo, un caso aplicado para el diseño y construcción de una ontología de testing de software. Por lo tanto, las contribuciones que se persiguieron fueron dos, a saber: especificar el proceso para DSR y aplicarlo en su totalidad para construir un artefacto no trivial y relevante, como es una ontología de testing de software.

La ontología de testing es de especial interés para nuestro grupo de I+D –y para la comunidad científica y profesional en general- ya que, en los últimos años, los esfuerzos de investigación estuvieron enfocados hacia el diseño de estrategias integradas que permitan establecer un curso de acción específico (además de proveer los métodos y la base conceptual necesaria) para alcanzar metas con propósitos de evaluación. En especial, el desarrollo de una ontología de testing como artefacto empleando DSR permite a partir de la misma construir estrategias que definan el qué (proceso) y el cómo (métodos) para llevar adelante metas con propósitos de testing de software.

Como trabajo futuro se prevé utilizar el enfoque DSR y su proceso aquí especificado para la construcción de nuevos artefactos. Entre ellos, se van a desarrollar procesos y métodos de testing para dar lugar, junto con la ontología del caso aplicado, a una familia de estrategias integradas de testing de software. Además, el grupo de investigación está avanzando en la armonización conceptual de otras ontologías previamente desarrolladas con la arquitectura ontológica FCD-OntoArch (ver Fig.

2), en donde la ontología de testing aquí presentada está ubicada a nivel de dominio superior (top-domain level). Por otra parte, prevemos construir nuevos componentes conceptuales para desarrollo y mantenimiento de software (ver Fig. 1) en el contexto de dicha arquitectura. Por último, pero no menos importante, todos estos nuevos artefactos a construir enmarcados dentro del enfoque DSR, permitirán a su vez una validación y verificación más amplia del proceso de DSR aquí propuesto.

AGRADECIMIENTOS

Este trabajo y línea de investigación están soportados parcialmente por el proyecto de I+D titulado "Familias de Estrategias de Testing Funcional/No-Funcional y de Evaluación para diferentes propósitos de Metas de Negocio", aprobado en 2019 por la Facultad de Ingeniería de la Universidad Nacional de La Pampa. Además, esta investigación está soportada por la Agencia Argentina de Ciencia y Tecnología en el proyecto PICT 2014-1224 ejecutado en la Facultad de Ingeniería de la Universidad Nacional de La Pampa. En este contexto, uno de los colaboradores cuenta con soporte económico de Beca Doctoral de Iniciación a la Investigación.

V. REFERENCIAS

[1] L. Olsina, P. Becker, "Family of Strategies for different Evaluation Purposes", *XX Conferencia Iberoamericana en Software Engineering (CIbSE'17),* CABA, Argentina, 2017, Published by Curran Associates, pp. 221-234, ISBN 978-99967-839-2-0.

[2] L. Olsina, M.F. Papa, H. Molina, "How to Measure and Evaluate Web Applications in a Consistent Way", Ch. 13 in *Springer book: Web Engineering: Modeling and Implementing Web Applications*, Rossi, Pastor, Schwabe & Olsina (Eds), pp. 385–420, 2008.

[3] P. Becker, M.F. Papa, L. Olsina, "Process Ontology Specification for Enhancing the Process Compliance of a Measurement and Evaluation Strategy", *CLEI Electronic Journal*, 2015, 18:(1), pp. 1-26.

[4] L. Olsina, P. Becker, "Linking Business and Information Need Goals with Functional and Non-functional Requirements", *XXI Conferencia Iberoamericana en Software Engineering (CIbSE'18)*, Bogotá, Colombia, Published by Curran Associates, 2018, pp. 381-394.

[5] A.R. Hevner, S. Chatterjee, "Design Research in Information Systems: Theory and Practice", Springer Publishing Company, Incorporated, 1st Edition, 2010.

[6] G. Tebes, D. Peppino, P. Becker, G. Matturro, M. Solari, L. Olsina, "Analyzing and documenting the systematic review results of software testing ontologies", *Information and Software Technology*, Elsevier, vol. 123, pp. 1-32, 2020, doi: 10.1016/j.infsof.2020.106298.

[7] J. Mckay, P. Marshall, "A Review of Design Science in Information Systems", *ACIS 2005 Proceedings - 16th Australasian Conference on Information Systems*, 2005.

[8] S. March, G. Smith, "Design and Natural Science Research on Information Technology", 1995, 15:(4), pp. 251-266, doi: 10.1016/0167-9236(94)00041-2.

[9] K. Peffers, T. Tuunanen, M. Rothenberger, S. Chatterjee, "A design science research methodology for information systems research", *Journal of Management Information Systems*, 2007, 24:(3), pp. 45-77.

[10] G.L. Geerts, "A design science research methodology and its application to accounting information systems research", *International Journal of Accounting Information Systems*, 2011, 12:(2), pp. 142-151.

[11] A.R. Hevner, S.T. March, J. Park, S. Ram, "Design Science in Information Systems Research". *Management Information Systems Quarterly*, 2004, 28:(1), pp. 75-105.

[12] P. Offermann, O. Levina, M. Schönherr, U. Bub, "Outline of a design science research process". *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, Philadelphia, Pennsylvania, 2009, pp 1-11. ACM.

[13] K. Peffers, T. Tuunanen, C. Gengler, et al., "The design science research process: A model for producing and presenting information systems research", *Proceedings of First International Conference on Design Science Research in Information Systems and Technology*, DESRIST,

2006, pp. 83-106, doi: 10.2753/MIS0742-1222240302.

[14] B. Curtis, M. Kellner, J. Over, "Process Modelling", *Com. of ACM*, 1992, 35:(9), pp. 75-90.

[15] Object Management Group (OMG), "Software & Systems Process Engineering Meta-Model Specification", v. 2.0, abril, 2008.

[16] G. Tebes, D. Peppino, B. Rivera, P. Becker, M.F. Papa, L. Olsina, "Especificación del Proceso de Design Science Research: Caso Aplicado a una Ontología de Testing de Software", *7mo Congreso Nacional de Ingeniería Informática – Sistemas de Información (CoNaIISI'19)*, pp. 1-10 (2019)

[17] G. Tebes, L. Olsina, D. Peppino, P. Becker, "TestTDO: A Top-Domain Software Testing Ontology", in *23rd Ibero-American Conference on Software Engineering (CIbSE'20)*, 2020, pp. 1–14.

[18] A.R. Hevner, "A Three Cycle View of Design Science Research", *Scandinavian Journal of Information Systems*, 2007, 19:(2), pp. 87-92.

[19] V. Vaishnavi, B. Kuechler, P. Stacie, "Design Research in Information Systems", Association for Information Systems, 2004, URL: http://desrist.org/design-research-in-information-systems.

[20] R.J. Wieringa, "Design science methodology for information systems and software engineering", London: Springer, 2014, doi: https://doi.org/10.1007/978-3-662-43839-8.

[21] A. Asman, R.M. Srikanth, "A Top Domain Ontology for Software Testing", Real Instituto de Tecnología, Estocolmo, Suecia, Tesis de Master, 2015.

[22] M. D'Aquin, A. Gangemi, "Is there beauty in ontologies?", *Applied Ontology*, 2011, 6:(3), pp. 165-175.

[23] É.F. De Souza, R. De Almeida Falbo, N.L. Vijaykumar, "ROoST: Reference ontology on software testing", *Applied Ontology*, 2017, 12:(1), pp. 59-90.

[24] ISO/IEC/IEEE 29119:2013. The international standard for software testing. Software and systems engineering - Software Testing - Parts 1, 2, 3 and 4.

[25] ISTQB–International Software Testing Qualifications Board. Available at https://www.istqb.org/.

[26] P. Becker, Olsina L., D. Peppino, G. Tebes, "Specifying the Process Model for Systematic Reviews: An Augmented Proposal", *Journal of Software Engineering Research and Development*, v.7, pp. 1-23, ISSN 2195-1721, 2019.

[27] N.H. Thuan, A. Drechsler, P. Antunes, "Construction of Design Science Research Questions", *Communications of the Association for Information Systems*, 2019, 44, doi: https://doi.org/10.17705/1CAIS.04420.

[28] M. Fernández-López, A. Gómez-Pérez, N. Juristo, "METHONTOLOGY: From Ontological Art Towards Ontological Engineering", *Spring Symposium on Ontological Engineering of AAAI*, Stanford University, California 1997, pp. 33–40.

[29] N.F. Noy, D.L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology", *Stanford knowledge systems laboratory technical report KSL-01-05* and *Stanford medical informatics technical report SMI-2001-0880*, 2001.

Recibido: 2020-07-29
Aprobado: 2020-08-06
Hipervínculo Permanente: http://www.reddi.unlam.edu.ar
Datos de edición: Vol. 5-Nro. 1-Art. 4
Fecha de edición: 2020-08-15

ISSN: 2525-1333. Vol.: 5 - Nro. 1 (AGOSTO-2020)