Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Ontología de apoyo a las pruebas de software en la UCI.

Autor:

Carlos Parker Leiva

Tutores:

Ing. Aliuska Castañeda Martínez

MSc. Yoan Antonio López Rodríguez

Dr.C. Yamilis Fernández Pérez

La Habana,21 de junio del 2018

Declaración de autoría

Declaro ser único autor del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a ser uso del mismo en su beneficio.

Para que así conste firmo la presente a los __ días del mes de ____ del año _____

_____

Carlos Parker Leiva

Firma del Autor

_____

Dr.C. Yamilis Fernández Pérez

Firma del Tutor

_____

Ing. Aliuska Castañeda Martínez

Firma del Tutor

_____

MSc. Yoan Antonio López Rodríguez

Firma del Tutor

Dedicatoria

A mis abuelos y a mi tía Iliana que a pesar de no estar físicamente me han guiado hasta aquí y sé que estarán orgullosos de mí.

Agradecimientos:

A mi mama por ser mi guía, apoyarme en cada una de mis decisiones, darme fuerzas y sobre todo por la confianza que siempre ha depositado en mí.

A mi papa por su preocupación, dedicación, apoyo constante y porque sé que este también es su sueño.

A Olguita por su preocupación y por siempre preguntar: ¿cómo va la tesis?

A mi hermana por ser el espejo en el que me miro cada día, por estar siempre a mi lado en todo momento.

A mi cuñado por ser el hermano mayor que no tuve, así como a su familia por hacernos parte de ellos y apoyarme en toda la carrera.

A mi tía por ser mi segunda mama y complacer en cada uno de mis caprichos.

A Julio y Julito por el apoyo que siempre me han demostrado.

A mis tutoras de las que he aprendido mucho en todo el tiempo que me han dedicado, gracias por la atención y el cariño.

A mi tutor Yoan por estar siempre disponible cuando lo necesitábamos sin importar día ni hora, por su paciencia y dedicación.

A todos profesores que han aportado su granito de arena y han hecho posible que haya llegado hasta aquí, en especial a la profesora Karenia por su entrega y dedicación.

A mi hermano Julio que viene soportándome desde la infancia.

A Pila, Mery, Wendy, Liana y el resto de la familia por haberme acogido y apoyarme en todo momento.

A mi hermana del alma Daniela que desde que la conocí siempre ha estado a mi lado en las buenas y malas (mi ángel de la guarda) y a su familia por todo el cariño.

Al Yuni que más que un amigo ha sido un hermano, a mi hermanito El Flaco, a la Flaca, al Vargas, al Serguinho, a todos mis compañeros de aula por la paciencia, a mis compañeros de apartamento, a Ale y en general a todas mis amistades de la facultad.

A Claudia y a Betty por su cariño incondicional.

A mis excelentes amigas de la facultad 3 Eilen, Nay y Jessi.

A mi compañero, hermano y amuleto "Carrión" por ser mi compañero de batalla durante todos estos años.

En general a todos los que han hecho más fácil mi estancia en esta escuela y han colaborado para la elaboración de esta tesis.

Resumen

El proceso productivo de la Universidad de las Ciencias Informáticas está certificado CMMi nivel 2. Se tiene definido el área de proceso de Aseguramiento de la Calidad del proceso y del producto, y dentro de este los subprocesos de pruebas de software. Estos subprocesos son complejos por la cantidad de personal involucrado de diferentes entidades con diversos niveles de conocimiento. Además, influye también el gran volumen de documentación generada. Todo ello, provoca dificultades en el entendimiento y comprensión total del mismo para una adecuada institucionalización. La presente investigación tiene como objetivo desarrollar una ontología que apoye el subproceso de pruebas de software en la Universidad, permitiendo la organización y comunicación del conocimiento generado. Para la construcción de la ontología se utilizó la metodología Methontology.

La ontología incluye el conocimiento tácito del personal involucrado en las pruebas, así como el conocimiento explícito representado en diferentes fuentes como el estándar NC ISO / IEC 25000, los procedimientos definidos en la entidad, etc. Dicha ontología proporciona un vocabulario común y soluciona los problemas de inconsistencias e integridad detectados. Se valida la solución ontológica utilizando un esquema para evaluar ontologías únicas para un dominio de conocimiento, evaluando el lenguaje utilizado para la codificación, la exactitud de la estructura taxonómica, el significado de los términos y conceptos representados; y por último la adecuación a los requerimientos especificados al inicio del desarrollo. Se garantizó de esta forma la calidad de la propuesta ontológica.

Palabras clave: Calidad, Gestión de conocimiento, Metodología, Ontologías, Pruebas de Software.

Índice

Índice de Figuras.

Índice de Tablas.

Introducción

En la actualidad, con el surgimiento de las TICs, el manejo de la información ha evolucionado considerablemente y en consecuencia la informática como ciencia fundamental ha jugado un papel esencial, así como la computación. Ha posibilitado un mecanismo mediante el cual el ser humano a través de sistemas informáticos, ha podido dar solución a interrogantes y situaciones cuyo procedimiento de manera manual sería muy tedioso y complicado.

El surgimiento de Internet ha propiciado que con el transcurso del tiempo se hayan logrado grandes avances, dentro de los cuales se encuentra la World Wide Web (WWW) mayormente conocida como la red de redes, creada alrededor de 1989 por Tim Berners-Lee (Lapuente, 2013). El acceso a la información a través de la misma es cada vez mayor, teniendo en cuenta que su uso facilita la obtención de información referente a disímiles temas desde cualquier parte del mundo.

A pesar del gran éxito que ha alcanzado mundialmente la web aún presenta algunas limitaciones, atendiendo a que existe un gran cúmulo de información que a su vez genera desorganización de la misma. Esto provoca que cuando se desee realizar una búsqueda, a pesar de contar con la información idónea, los resultados no sean los más convenientes.

Una vía de solución a este problema podría ser la Web Semántica (WS), Hendler, Berners-Lee y Miller en 2012 la definen como: "Una extensión de la actual Web en la que a la información disponible se le otorga un significado bien definido, que permita a los ordenadores y a las personas trabajar en cooperación. Está basada en la idea de proporcionar en la Web datos definidos y enlazados, permitiendo que aplicaciones heterogéneas localicen, integren, razonen y reutilicen la información presente en la Web" (Hendler, Berners-Lee y Miller, 2012).

La web semántica consiste en un nuevo paradigma web para acceder, buscar, compartir y gestionar información a través de la combinación de tecnologías y de estructuras de gestión del conocimiento. El concepto de web semántica proporciona herramientas para el almacenamiento, intercambio y consulta de esta información. Uno de los componentes básicos de la web semántica son las ontologías, modelo para la representación, especificación y comunicación del conocimiento de manera genérica en un determinado dominio. Según el objetivo de la web semántica deberá ser interpretable además de los humanos, por máquinas y agentes inteligentes para desarrollar tareas rutinarias y específicas. Las ontologías juegan un rol importante no solo en la web, sino también en diversas áreas como: Sistemas de Información, Almacenes de Datos, entre otras.

Este paradigma puede ser aprovechado para la gestión del conocimiento en los procesos de la industria del software, teniendo en cuenta que la idea es organizar los términos asociados a un determinado proceso, con el fin de facilitar los procedimientos y estructurar la información asociada al mismo, para con esto representar todo el conocimiento que se haya generado.

Para desarrollar un software y que el mismo cuente con la calidad requerida, se tiene en cuenta una serie de etapas o procesos por los que dicho sistema debe pasar antes de ser entregado al cliente. En estas etapas se comprueba el correcto funcionamiento del software dígase fiabilidad, eficiencia, portabilidad, compatibilidad, usabilidad entre otros aspectos fundamentales. La realización de las pruebas de software es esencial para velar por la correcta confección del sistema, y que este cumpla los requisitos de cada etapa en la que se esté trabajando.

El proceso productivo de la UCI está certificado CMMI nivel 2, por lo que se definen actividades de calidad, respondiendo al área de proceso aseguramiento de la calidad de los procesos y productos (PPQA). Dentro de los subprocesos para la gestión de las actividades de calidad se encuentran la ejecución de las pruebas a nivel de proyectos, de centro y de gerencia. Las pruebas a nivel de gerencia se desarrollan en la Dirección de Calidad de la UCI, al igual que las de liberación de producto. Estos subprocesos son complejos por la cantidad de personal involucrado de diferentes entidades con diversos niveles de conocimiento. Además, la fluctuación del personal es un factor influyente, pues cuando las personas se marchan se llevan consigo todo el conocimiento tácito adquirido impidiendo que pueda ser convertido en explícito. Influye también el gran volumen de documentación generada. Todo ello, provoca dificultades en el entendimiento y comprensión total de dichos subprocesos, fundamentalmente de los términos y conceptos utilizados.

En los estándares internacionales que guían y orientan los aspectos fundamentales de estos subprocesos, existen conceptos un tanto ambiguos y solapados además de disímiles significados para un mismo término, lo que acrecienta esta problemática.

Las soluciones ontológicas analizadas (Felipe, 2011), (Mateus Ferreira, 2006), (Perez, 2011) y (Souza, 2013) no abarcan todos los términos y conceptos del subproceso de pruebas de software a los diferentes niveles que caracterizan el proceso de desarrollo en la UCI.

Todo esto conlleva a concluir que hay una insuficiente organización de los términos utilizados, no existe un vocabulario común y por ende implica la no correcta ejecución de las pruebas. Además, la UCI y específicamente la dirección de calidad no cuenta con una herramienta que ayude a organizar, recuperar y comunicar el conocimiento generado en estos subprocesos.

Esta situación conlleva a definir como **problema de la investigación** el siguiente:

¿Cómo contribuir a la organización y comunicación del conocimiento generado en el subproceso de pruebas de software en la UCI?

El **objeto de estudio** sería la gestión del conocimiento basado en ontologías y enmarcando como **campo de acción** la ontología en el subproceso de prueba de software en la dirección de calidad de la UCI.

Como **objetivo general** se propone:

Desarrollar una ontología que apoye el subproceso de pruebas de software en la UCI, permitiendo la organización y comunicación del conocimiento generado.

El objetivo general se desglosa en los **objetivos específicos** siguientes:

- Revisar la bibliografía para generar el marco teórico conceptual de la investigación que represente las tendencias actuales de las ontologías y del proceso de pruebas de software.
- Seleccionar la metodología, herramientas y tecnologías a usar para el desarrollo de ontologías.
- Diseñar una ontología que represente el conocimiento generado en las pruebas de software en la UCI.
- Validar la propuesta ontológica en función de la organización y comunicación del conocimiento generado en las pruebas de software.

Para dar cumplimiento a los objetivos específicos se utilizaron los métodos científicos que se exponen a continuación. Entre los **métodos teóricos** se tienen:

- **Análisis Histórico y Lógico:** Se usa este método para el análisis y evolución de conceptos relacionados con las ontologías, la gestión del conocimiento y las pruebas de software.

- **Analítico-Sintético:** Utilizado para estudiar, analizar los trabajos, tendencias similares, sobre el desarrollo de las ontologías y determinar los elementos que permitieron agrupar los detalles a tener en cuenta para gestionar el conocimiento generado en las pruebas de software.

- **Modelación:** Se utiliza este método para modelar los diferentes artefactos de diseño que necesita la ontología.

Entre los **métodos empíricos** tenemos**:**

- **Experimentación:** Se usa a la hora de validar la propuesta ontológica en función de la comunicación, la organización de la información y la calidad del procedimiento de pruebas.

- **Entrevista:** Se realizan entrevistas informativas, focalizadas en el tema a investigar. Las mismas se efectuaron a especialistas de empresas como XETID, Calisoft y de la Dirección de Calidad UCI, para de esta forma conocer sobre el funcionamiento del subproceso de prueba de software en estas entidades.

- **Análisis documental:** Se emplea en la consulta de la literatura especializada en las temáticas afines de la investigación y la documentación del subproceso de pruebas de software en la UCI.

Posibles resultados

Una ontología que apoye el subproceso de pruebas de software en la UCI. La misma garantizará la organización y comunicación del conocimiento generado en dicho subproceso.

El presente documento está estructurado en: introducción, tres capítulos, conclusiones generales, recomendaciones y bibliografía. A continuación, se resumen los principales elementos tratados en cada capítulo:

- Capítulo 1: Fundamentación Teórica. En el mismo se realiza un análisis de los principales conceptos asociados al tema de investigación, se valoran las herramientas, tecnologías y metodologías usadas para el desarrollo de ontologías. Se investigan soluciones existentes para este tipo de problemas y las dificultades halladas para su aplicación en la UCI.

- Capítulo 2: Diseño de la Propuesta. En este capítulo se detalla el desarrollo de la propuesta, enfatizando en cada uno de sus elementos y aspectos fundamentales guiados por la metodología Methontology para de esta forma concluir con la creación de la ontología.

- Capítulo 3: Validación de la ontología. El capítulo está destinado a la validación de la solución propuesta, proporcionando como evidencia una herramienta informática basada en la ontología que deberá responder a las preguntas de competencias definidas.

Capítulo 1: Fundamentación teórica

En el presente capítulo se analizan los conceptos relacionados con calidad, especificando en las pruebas de software. Además, se describe el proceso analizando los principales problemas encontrados en cuanto a la gestión del conocimiento. Por otra parte, se investigan soluciones existentes para este tipo de problemas y las dificultades halladas para su aplicación en la UCI. Por último, se valoran las herramientas, metodologías y tecnologías utilizadas para el desarrollo de ontologías.

1.1 Calidad, control y pruebas de software

En estos últimos años la tecnología ha evolucionado vertiginosamente, lo que conlleva a que tengamos clientes cada vez más exigentes a la hora de adquirir un software. Teniendo en cuenta esto un gran número de organizaciones han agregado la calidad al proceso de creación y desarrollo de software como una condición indispensable que este debe cumplir. Al mismo tiempo han utilizado gran parte de sus recursos al estudio del término calidad de software, por ser este una pieza fundamental durante el proceso de desarrollo de un producto de software. Ejemplo de esto es la IEEE la cual define la calidad de software como: “El grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario” (Vasco, 2009).

Para lograr que un software tenga la claridad requerida se debe llevar a cabo su control durante todas las etapas del ciclo de vida del mismo. La IEEE define como control de la calidad a un conjunto de actividades para evaluar la calidad de los productos desarrollados (IEEE, 1991). El aspecto a considerar en el control de la calidad del software es la Prueba de Software”.

En el caso específico del desarrollo de software, las pruebas se han definido de diversas maneras:

Pressman plantea que “Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.” (Pressman, 2010). Por su parte la IEEE plantea que las pruebas constituyen “Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.” (IEEE, 1991). Por su parte (Leonard L. Tripp, 2014) precisa que la prueba de software es la verificación dinámica del comportamiento de un programa contra el comportamiento esperado, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada desde el dominio infinito de ejecución. Luego del análisis de estos conceptos el autor entiende que las pruebas de software no son más que las actividades que se realizan utilizando diseños de prueba donde los artefactos son evaluados bajo

requisitos especificados y se guardan los resultados, estos resultados son comparados con los esperados, luego se debe emitir una evaluación basada en aspectos determinados del artefacto.

Como se menciona anteriormente, las pruebas de software se consideran parte fundamental del ciclo de vida de desarrollo de cualquier aplicación y a su vez tienen su propio ciclo de vida. Dependiendo de la organización, este ciclo puede tener más o menos fases, a continuación, se describen las fases más importantes:

Plan de prueba: en esta fase se redacta un plan de prueba donde se describe la estrategia a seguir durante el desarrollo de las pruebas, con el fin de organizar las actividades para que el proceso se lleve a cabo con la calidad requerida.

Análisis de requerimientos: no es más que ramificación de la fase de planificación, en esta fase se comienzan a documentar los planes detallados y se empiezan a analizar los casos de prueba.

Desarrollo de prueba: es donde se obtienen los disímiles artefactos utilizados para la ejecución efectiva de las pruebas de software.

Ejecución de pruebas: aquí es donde se ejecutan las pruebas basadas en el plan y los documentos de prueba además de reportarse los errores encontrados.

Reporte de pruebas: una vez terminado el proceso, se generan los reportes finales y se llega a la conclusión de si el software está listo o no para ser liberado.

Análisis de resultados: se precisa entre el equipo de desarrollo y el cliente los desperfectos que deben ser corregidos.

Pruebas de regresión: se realizan para asegurar que al integrar nuevas funcionalidades al software las anteriormente sigan funcionando correctamente y en su conjunto, el funcionamiento del software sea el esperado.

Cierre de prueba: para concluir se archivan los resultados y documentos generados durante el proceso, que luego serán usados como referencia en futuros proyectos (Sánchez, 2013).

Luego de realizado el cierre de las pruebas se da por concluido el ciclo. Este proceso genera documentos que deben estar bien descritos y con calidad, pues el conocimiento generado puede ser utilizado o reutilizado en un contexto determinado.

1.2 Gestión del Conocimiento

La Gestión de Conocimiento es definida como un proceso integrador en el que confluyen la gestión de la información, la tecnología y los recursos humanos, así como su implementación se orienta al perfeccionamiento de los procesos de mayor impacto, mejor explotación del conocimiento en función de los procesos y su distribución en toda la organización, sobre la base del uso intensivo de las redes y las tecnologías (María Aurora Soto Balbón, Norma M. Barrios Fernández, 2006). Es el proceso de identificar, agrupar, ordenar y compartir continuamente conocimiento de todo tipo para satisfacer necesidades presentes y futuras, para identificar y explotar recursos de conocimiento tanto existentes como adquiridos y para desarrollar nuevas oportunidades.

Existen dos tipos de conocimiento: explícito, que se ha definido como la noción objetiva y racional que puede ser expresada con palabras, números, fórmulas, etc. y el tácito que es aquel que una persona, comunidad, organización o país, tiene incorporado o almacenado en su mente, en su cultura y es difícil de explicar. Es necesario explicar que este conocimiento puede estar compuesto por: ideas, experiencias, destrezas, habilidades, costumbres, valores, historias y creencias. El problema que presenta este tipo de conocimiento es su dificultad a la hora de ser transmitido, por ello es necesario gestionarlo creando códigos que faciliten su transmisión (Figueroa, et al., 2006).

El conocimiento no siempre está disponible cuando es necesario para la organización. Para tratar este problema ha surgido con fuerza en los últimos años una nueva disciplina, la gestión del conocimiento. La gestión del conocimiento se ocupa de la identificación, captura, recuperación, compartimiento y evaluación del conocimiento organizacional. El objetivo es que todo el conocimiento que reside en una empresa pueda ser utilizado por quien lo necesite para actuar de manera adecuada en cada momento (Simón, et al., 2006). Para poder cumplir con lo expuesto anteriormente es necesario que intervengan los sistemas de información de organización del conocimiento(SOCs).

Al mismo tiempo (Barité, 2014), propone los siguientes tipos específicos de sistemas de organización del conocimiento, con excepción de los diccionarios y los glosarios, teniendo en cuenta que estos no cumplen estrictamente con los objetivos propios de esta clase de herramientas:

- **Tesauros:** Un tesauro es un vocabulario controlado y estructurado formalmente, formado por términos que guardan entre sí relaciones semánticas y genéricas: de equivalencia, jerárquicas y asociativas. Se trata de un instrumento de control terminológico que permite convertir el lenguaje natural de los documentos en un lenguaje controlado, dado que representa, de manera

particular, el contenido de estos, con el fin de servir tanto para la indización, como para la recuperación de los documentos (Lapuente, 2013).

- **Taxonomía:** La taxonomía es la ciencia que estudia los principios, métodos y fines de la clasificación. Este término se utiliza especialmente en biología para referirse a una clasificación ordenada y jerarquizada de los seres vivos y en educación para ordenar y diseñar los objetivos del aprendizaje. (Significados.com, 2017).Al mismo tiempo (Centelles, 2005) define una taxonomía como un tipo de vocabulario controlado en que todos los términos están conectados mediante algún modelo estructural (jerárquico, arbóreo, facetado...) y especialmente orientado a los sistemas de navegación, organización y búsqueda de contenidos de los sitios web.

- **Ontología:** Ontología es una antigua disciplina que, en sentido filosófico, se define como un esquema específico de categorías que refleja una visión específica del mundo. Desde el punto de vista informático las ontologías son teorías que especifican un vocabulario relativo a un cierto dominio las mismas toman un papel clave en la resolución de interoperabilidad semántica entre sistemas de información y su uso. (Luna, 2012)

1.3 Ontología

El término ontología se ha empleado desde hace muchos siglos en el campo de la filosofía y del conocimiento y hace ya varias décadas cobró especial relevancia en el campo de la biblioteconomía y la documentación. Hoy ha sufrido un nuevo impulso debido al desarrollo de la Web Semántica donde prima la idea de transformar la red no sólo en un espacio de información, sino también en un espacio de conocimiento.

La ontología es una antigua disciplina que se define como un esquema específico de categorías que refleja una visión específica del mundo. la misma es una especificación de una conceptualización, esto es, un marco común o una estructura conceptual sistematizada y de consenso no sólo para almacenar la información, sino también para poder buscarla y recuperarla. Una ontología define los términos y las relaciones básicas para la compresión de un área del conocimiento, así como las reglas para poder combinar los términos para definir las extensiones de este tipo de vocabulario controlado. Actualmente las ontologías son materia de investigación, desarrollo, y aplicación en disciplinas relacionadas con la computación, la información y el conocimiento. (Lapuente, 2013)

En sistemas de Inteligencia Artificial (IA), lo que existe es lo que puede ser representado. Esos conjuntos de objetos, y las relaciones que se establecen entre ellos, son reflejados en un vocabulario con el cual representamos el conocimiento en un sistema basado en conocimiento. Así, en el contexto

de IA, podemos describir la ontología de un programa como un conjunto de términos. En tal ontología, las definiciones asocian nombres de entidades del universo del discurso con textos comprensibles por los humanos que describen el significado de los nombres, y axiomas formales que limitan la interpretación y buen uso de dichos términos. Formalmente, una ontología es una teoría lógica. (Valencia, 2010).

Para (Weigand, 1997) una ontología es una base de datos que describe los conceptos en el mundo o algún dominio, algunas de sus propiedades y cómo los conceptos se relacionan entre sí.

Luego de analizados los conceptos anteriores, se decide tomar como referencia la definición de ontología dada por (Torcoroma Velásquez Pérez, 2011) donde define a las ontologías como una técnica de representación del conocimiento en un contexto determinado. Se realiza tal elección teniendo en cuenta que es la más abarcadora y la que más se ajusta al dominio de la investigación.

Para llevar a cabo el diseño de una ontología, es de gran importancia tener en cuenta algunas de las características que las mismas deben presentar.

1.3.1 Principios para el diseño de ontologías

Los autores (Esmeralda Ramos, Haydemar Núñez, Roberto Casañas, 2011) plantean que los principios de diseño a tener en cuenta son los siguientes:

- Claridad y Objetividad: Definir los conceptos en forma clara y objetiva utilizando lenguaje natural para evitar ambigüedades.
- Coherencia: Garantizar que todas las inferencias derivadas sean consistentes con los axiomas.
- Completitud: Los conceptos deben ser expresados en términos necesarios y suficientes.
- Estandarización: Siempre que sea posible, los nombres asignados a los términos deberán seguir un estándar, definiendo y respetando reglas para la formación de los mismos.
- Máxima extensibilidad monótona: Deberá ser posible incluir en la ontología especializaciones o generalizaciones, sin requerir una revisión de las definiciones existentes.
- Principio de distinción ontológica: Las clases de la ontología con diferente criterio de identidad, deberán ser disjuntas.

- Diversificación de las jerarquías: Para que la ontología se vea favorecida con los mecanismos de herencia múltiple, es conveniente usar tantos criterios de clasificación como sea posible.

- Minimización de la distancia semántica: Conceptos similares deberán ser agrupados y representados utilizando las mismas primitivas.

- Mínimo compromiso ontológico: Una ontología debería imponer las menores exigencias posibles sobre el dominio que modela, es decir, se deben construir sólo los axiomas necesarios para representar el mundo a ser modelado (Esmeralda Ramos, Haydemar Núñez, Roberto Casañas, 2011).

1.3.2 Utilidad de las ontologías

En (Carrasco, 2010) se exponen una serie de aportes relacionados con la utilidad de las ontologías.

Para los ingenieros de software:

- Disminuye la dificultad de la comunicación entre analista e interesado para definir los requisitos de un sistema.

- Puede disminuir la baja reutilización de componentes y la escasa generación automática de código.

- Mejoran la comunicación.

- Mejoran interoperabilidad entre sistemas de información.

- Mejoran la calidad de los sistemas de software.

En los sistemas de software posibilitan:

- La capacidad de reutilización.

- Generan confiabilidad en los sistemas, pues permiten automatizar chequeo de la consistencia.

- Los sistemas que usan ontologías en su construcción sirven para mejorar la documentación del software y así reducir costos de mantenimiento.

1.3.3 Beneficios de utilizar antologías

María Jesús Lamarca Lapuente (Lapuente, 2013) expone entre los beneficios de utilizar ontologías los siguientes:

- Proporcionan una forma de representar y compartir el conocimiento utilizando un vocabulario común
- Permiten usar un formato de intercambio de conocimiento
- Proporcionan un protocolo específico de comunicación
- Permiten una reutilización del conocimiento

1.3.4 Tipos de ontología

Se pueden establecer distintos tipos de ontologías atendiendo a diversos aspectos. Podemos destacar la siguiente clasificación, aunque existen otras muchas:

Según su dependencia y relación con una tarea específica desde un determinado punto de vista, (Guarino, 1995) clasifica las ontologías en:

Ontologías de Alto nivel o Genéricas describen conceptos más generales.

Ontologías de Dominio describen un vocabulario relacionado con un dominio genérico.

Ontologías de Tareas o Técnicas básicas describen una tarea, actividad o artefacto, por ejemplo, componentes, procesos o funciones.

Ontologías de Aplicación describen conceptos que dependen tanto de un dominio específico como de una tarea específica, y generalmente son una especialización de ambas.

Luego de analizados los tipos de ontologías, basados en el ámbito del conocimiento al que serán aplicados, se decide utilizar las **ontologías de aplicación**, teniendo en cuenta que esta clasificación es la que más se adapta a la propuesta ontológica. Una vez definido esto es importante saber que el conocimiento en las ontologías es formalizado usando cinco tipos de componentes.

1.3.4 Componentes de las ontologías

Según (Gruber, 1993) las ontologías se componen de:

Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

Relaciones: representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio.

Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.

Instancias: se utilizan para representar objetos determinados de un concepto.

Reglas de restricción o axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos (Gruber, 1993).

1.3.5 Metodologías para la creación de ontologías

Las metodologías aportan un conjunto de normas que indican cómo llevar a cabo las actividades durante el proceso de desarrollo de una ontología, qué técnicas son las adecuadas en cada actividad y qué produce cada una de ellas. Entre los requisitos del cliente está que se trabaje con Methontology teniendo en cuenta que esta metodología cuenta con una abundante comunidad de usuarios y que brindan una amplia documentación lo cual facilita su uso.

Methontology: permite la construcción de ontologías en el nivel del conocimiento. Incluye: la identificación del proceso de desarrollo de la ontología, un ciclo de vida propuesto y la metodología como tal. El proceso de desarrollo de la ontología identifica las tareas que deben realizarse cuando se construye una ontología (**especificación, conceptualización, formalización, implementación, mantenimiento**). El ciclo de vida basado en prototipos evolutivos identifica las etapas que atraviesa la ontología durante su tiempo de vida. Finalmente, la metodología en sí misma, específica los pasos que se deben seguir para desarrollar cada actividad, las técnicas usadas, los productos que se producirán y como serán estos evaluados. La fase principal en el proceso de desarrollo de una ontología usando el enfoque de Methontology es la fase de conceptualización (Felipe, 2011).

Especificación: mediante esta actividad se determina el por qué se construyó la ontología, el uso que se le dará y quiénes serán sus usuarios finales.

Conceptualización: esta actividad es la encargada de convertir una percepción informal del dominio en una especificación semi-formal, además de contribuir a la organización obteniéndose como resultado el modelo conceptual de la ontología.

Formalización: su función es transformar el modelo conceptual antes mencionado en un modelo formal o semicomputable.

Implementación: construye modelos computables en un lenguaje de ontologías (RDF, OWL.).

Mantenimiento: se encarga en caso de ser necesario de la corrección y actualización de la ontología.

1.3.6 Ontologías existentes

El uso de ontologías en la actualidad es cada vez mayor, dado que las mismas proporcionan la vía para representar el conocimiento que se pueda generar en un contexto determinado. Existen disímiles trabajos con diferentes enfoques en los que pueden ser utilizadas, con el objetivo de estructurar la información, simplificarla, modelarla, mejorar el rendimiento entre otros aspectos. En el transcurso de la investigación, específicamente durante la etapa de diseño y desarrollo de la ontología, es de gran importancia estudiar la posibilidad de la reutilización de las mismas y extenderla.

Se han consultado varias fuentes bibliográficas organizadas para su estudio en dos grupos. En el primer grupo, se encuentran los artículos dedicados al uso y desarrollo de las ontologías y el segundo grupo está integrado por aquellos trabajos que muestran ontologías desarrolladas (Felipe, 2011).

En el primer grupo se encuentra el artículo (Lapuente, 2013), donde la autora analiza y expone una serie de aspectos relacionados con las ontologías dígase: beneficios del uso de ontologías, definiciones de las mismas, componentes, características, entre otros aspectos. En (Hernández, 2015), la autora propone el desarrollo de un Modelo para el diseño y construcción de un sistema de recuperación de información basado en ontologías.

Para el segundo grupo, el cual constituye el fundamental para el desarrollo de este trabajo de diploma, fue necesario la búsqueda de ontologías relacionadas con las pruebas de software y la calidad. La revisión documental practicada durante la presente investigación arrojó un conjunto de ontologías aplicadas al dominio de calidad de software. Las mismas, como solución a la problemática actual,

llegan a ser muy generales o incompletas, no abordan todos los términos o las relaciones surgidas en la dirección de calidad.

En (Felipe, 2011) se hace referencia al uso de una ontología como herramienta para la gestión del conocimiento con el objetivo de apoyar la realización de un Modelo Cubano de Calidad para el Desarrollo de Aplicaciones Informáticas (MCDAI) en la Industria Cubana de Software.

Una ontología que describe la calidad de software la presenta (Katia Cristina Duarte, Ricardo de Almeida Falbo, 2000). En ella solo aparecen elementos fundamentales de la calidad en general, sin abordar las pruebas de software, ni definir el modelo de calidad. Además, presentan conceptos no explicados correctamente y con ambigüedad.

La ontología desarrollada por (Mateus Ferreira, 2006), caracteriza el proceso de medición. Es abarcadora y elimina problemas de homonimia, sinonimia y otras dificultades presentes entre estándares. No abarca el subproceso de prueba de software.

Investigaciones como las de (Perez, 2011) y (Souza, 2013), abarcan las pruebas de software y presentan ontologías generalizadoras de este dominio, pero hay conceptos a nivel de proyecto que no se incluyen, lo que limita la toma de decisiones.

Tras un análisis de dichas ontologías, no se encontró una que abarcara todo el subproceso de pruebas de software a los diferentes niveles y conceptos que caracterizan el proceso de desarrollo en la UCI. Además, entre las que abordan términos semejantes hay contradicciones.

Ninguna fuente analizada cubre las necesidades de los centros de desarrollo de la UCI y la Dirección de calidad. El conjunto de conceptos cubiertos por cada fuente no es tampoco homogéneo e igual.

Existen diferencias entre ellas ya que corresponden a desarrollo e investigaciones efectuadas por separado, en distintos periodos de tiempo. Esta es la razón por la cual se requiere un esfuerzo de integración que proporcione una vista común en función de eliminar las diferencias y conflictos.

Por todo lo anterior, se decide desarrollar una ontología para apoyar el subproceso de pruebas de software en la UCI, abarcando los conceptos fundamentales y adaptándolas a las nuevas necesidades de los proyectos y a la Dirección de Calidad de la UCI.

1.4 Selección de las tecnologías y herramientas.

1.4.1 Herramienta CASE

Visual Paradigm: Visual Paradigm es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Dicha herramienta fue concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 8.0 Community Edition fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Pressman, 2009).

Teniendo en cuenta las características antes descrita y que es una herramienta multiplataforma que facilita la organización de los diagramas, se decide utilizar Visual Paradigm UML 8.0 como herramienta CASE para la modelación, específicamente para describir el funcionamiento del subproceso de prueba de software en la dirección de calidad, así como en la confección del diagrama de relaciones binarias.

1.4.2 Herramientas para la construcción de ontologías

Luego de haberse realizado el estudio del estado del arte se identificaron una serie de herramientas destinadas a la construcción de ontologías, acorde a los requisitos del cliente se utilizó Protégé en su versión 5.2, a pesar de esto para mayor información y conocimiento se analizaron otras herramientas entre las que se encuentran:

Ontology Server: herramienta que permite al usuario la construcción de ontologías que comparten grupos geográficamente distribuidos. Fue desarrollado en el laboratorio de Sistemas de Conocimiento en la Universidad de Stanford. Este servidor es una extensión de Ontolingua. Al comienzo el término Ontolingua se usaba para referirse tanto al lenguaje para representar ontologías como a la herramienta utilizada para construirlas. Hoy el término se utiliza para referirse al lenguaje proporcionado por el Ontology Server. Su arquitectura permite el acceso a una librería de ontologías, traductores de lenguajes y a un editor para crear y navegar por una ontología (University, 2016).

OntoEdit: es una herramienta de edición de ontologías que apoya el desarrollo y mantenimiento de las mismas utilizando medios gráficos en un entorno web. Permite la representación semántica de lenguajes conceptuales y estructuras mediante conceptos, jerarquías de conceptos, relaciones y axiomas (Casamayor, 2005)

Protégé: herramienta basada en java mediante la cual el usuario puede construir ontologías, generar usuarios de entrada de datos y efectuar la propia entrada de datos. Es una herramienta que permite acceso a aplicaciones externas basadas en conocimiento. Además, es una biblioteca a la que otras aplicaciones pueden acceder, permitiéndoles acceder a las bases de conocimiento de las cuales se dispone. Cuenta con el apoyo activo de una sólida comunidad de usuarios y desarrolladores que formulan preguntas, escriben documentación y contribuyen con complementos, es totalmente compatible con las últimas especificaciones OWL 2 Web Ontology Language y RDF del World Wide Web Consortium, además de ser extensible y proporcionar un entorno plug-and-play que lo convierte en una base flexible para el desarrollo rápido de prototipos y aplicaciones (Research, 2016).

Se decidió utilizar Protégé en su versión 5.2 como herramienta de modelado para la construcción de la ontología dado que permite realizar ontologías con extensión OWL y RDF, debido a su entorno gráfico basado en plugins. Además, cuenta con un entorno extensible que se adapta a la construcción de numerosos tipos de ontologías, tiene una amplia comunidad de desarrolladores que brindan una bibliografía extensa y actualizada, y es la herramienta más utilizada en la actualidad para el trabajo con ontologías.

1.4.3 Razonadores de ontologías

Entre las herramientas más utilizadas para el trabajo con ontologías están los razonadores, los cuales sirven para realizar inferencia utilizando los componentes de las ontologías, dígase conceptos, relaciones, instancias y reglas de restricción o axiomas. Generalmente difieren en el lenguaje formal en el que se especifica el conocimiento, así como los lenguajes de consulta que puedan ser utilizados. Actualmente, son muchos los sistemas deductivos o razonadores basados en lógica descriptiva que permiten el razonamiento y la inferencia en las ontologías. Entre los principales se encuentran:

FaCT: Herramienta que puede ser usada para chequear el grado de satisfacción de las descripciones. Permite reglas transitivas, inversas, restricciones cualificadas, jerarquías etc. Es lo suficientemente expresivo para soportar y razonar sobre cualquier base de conocimiento. Escrito en Common Lisp, fácilmente ejecutable por cualquier programa Lisp de forma local, tiene escrita una versión servidor FaCT para ser usada sobre cualquier sistema con acceso a la red. Actualmente es el razonador empleado por defecto en el editor de ontologías OilEd para clasificar los conceptos en una jerarquía según las descripciones que tengan (Samper Zapater, 2005)

JTP: es una arquitectura para razonar sobre conocimiento descrito en DAML + OIL. Escrito en Java, soporta el modelo de axiomas de DAML + OIL, realiza una pre computación cuando se carga la ontología, y ha añadido un clasificador para realizar la jerarquía de conceptos de la ontología. Por el momento, es el único razonador que permite el lenguaje de consulta DQL específico para DAML + OIL (Samper Zapater, 2005).

HermiT: es un razonador de ontologías, escrito utilizando el Lenguaje de ontologías Web (OWL). Dado un archivo OWL, HermiT puede determinar si la ontología es consistente, identificar las relaciones de subsunción entre las clases, y mucho más. HermiT es el primer razonador de OWL a disposición del público basado en "hypertableau", cálculo que proporciona razonamiento mucho más eficiente que cualquier algoritmo previamente conocido (Samper Zapater, 2005).

Pellet: es un razonador de OWL-DL basado en Java. Puede ser utilizado conjuntamente con bibliotecas del API de Jena o del OWL. Mediante su uso es posible validar, comprobar la consistencia de ontologías, clasificar la taxonomía y contestar a un subconjunto de consultas RDQL (conocido como consultas a ABox en terminología del DL). Se trata de un razonador DL basado en los algoritmos tableaux desarrollados para DL expresiva. Soporta todas las construcciones del OWL DL incluyendo las relacionadas con los nominales, es decir, OWL: oneOf y OWL: hasValue (Samper Zapater, 2005).

A pesar de que se pueden utilizar indistintamente varios razonadores durante el proceso de desarrollo de la ontología, el razonador que se empleará será el pellet. El mismo está implementado en Java por lo que permite hacer inferencias y consultas basadas en ontologías descritas en OWL DL, así como validar y comprobar de esta forma la consistencia de ontologías. Además, es de Código Abierto y viene como plugins de Protégé en su versión 5.2.

Conclusiones del capítulo

A partir del estudio del estado del arte referente a las pruebas de software y las ontologías, se arribó a las consideraciones siguientes:

- Es imprescindible la gestión del conocimiento en las pruebas de software debido a la complejidad de este subproceso, producto a la cantidad de especialistas que participan y el gran cúmulo de documentación que genera.
- La ontología es una herramienta factible para la representación, especificación y comunicación del conocimiento que genera en el subproceso de prueba.
- No se encontró una ontología que abarcara todo el subproceso de pruebas de software a los diferentes niveles y conceptos que caracterizan el proceso de desarrollo en la UCI por lo que se decide desarrollar una desde cero.
- Teniendo en cuenta los requisitos del cliente y el análisis de las diferentes metodologías herramientas y tecnologías se decide utilizar Methontology como metodología de desarrollo, Protégé para la construcción de la ontología y pellet como razonador de ontologías.

Capítulo 2 Propuesta de solución

En el presente capítulo se describirá el procedimiento de prueba de software en la dirección de calidad, así como se llevará a cabo el diseño de la solución, el cual será guiado por la metodología Methontology donde se representarán las preguntas de competencias y se especificarán los conceptos, relaciones, instancias y reglas o axiomas.

2.1 Descripción de la propuesta:

En la dirección de calidad de software es donde se llevan a cabo las pruebas de liberación y aceptación a nivel de gerencia en la UCI, estas se realizan de la siguiente forma: Un proyecto tiene especialistas y producto, este producto a su vez tiene módulos, cada módulo responde a determinados requisitos. Al analizarse el cumplimiento de estos requisitos pueden ser detectadas no conformidades, lo cual es realizado por un revisor. De cada no conformidad se analiza el impacto que provoca, el tipo y el estado.

Es esencial controlar el paso de un estado a otro de una no conformidad. Al desarrollar un proyecto informático se obtienen artefactos. Un artefacto se puede dividir en partes. Estos artefactos de proyectos para ser analizados requieren de documentación.

Para detectar las no conformidades se ejecutan diferentes tipos de pruebas, que se realizan a diferentes niveles dígase unidad, integración, sistema, interna, liberación y aceptación. Cada tipo de prueba corresponde a una determinada característica de calidad.

En la figura 1 se muestra el Modelo conceptual confeccionado con la herramienta Visual Paradigm UML 8.0 Enterprise Edition, donde se presentan todas las clases y relaciones que se utilizarán para diseñar la ontología.

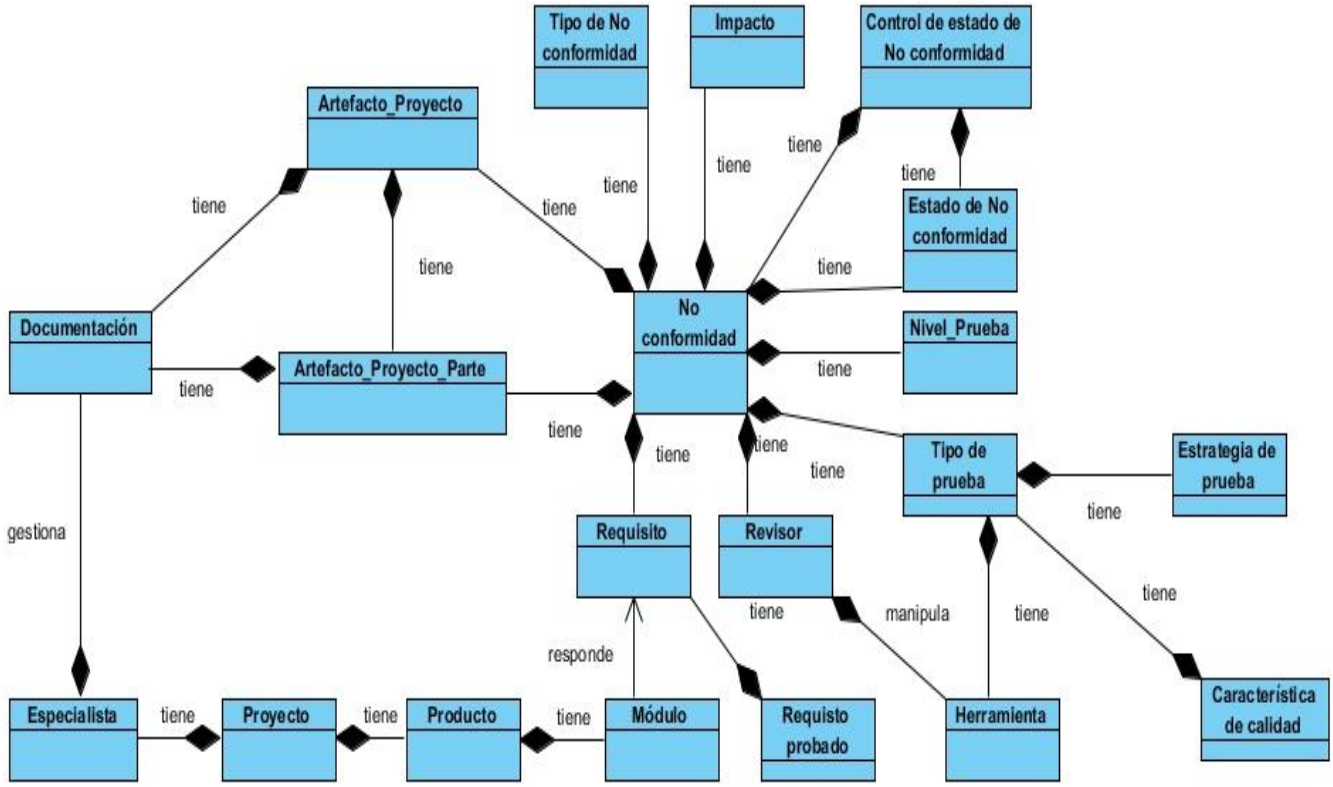


Figura 1.Descripción del subproceso de prueba.

2.2 Definición y desarrollo de la ontología

La metodología Methontology, seleccionada en el capítulo 1, proporciona guías sobre cómo llevar a cabo el desarrollo de una ontología mediante las actividades de especificación, conceptualización, formalización, implementación y mantenimiento. A continuación, se detallan los artefactos resultantes de cada actividad.

2.2.1 Actividad: Especificación

El objetivo principal por el que se construye la ontología es para proporcionar una mejor estructura para la base de conocimientos existente en la dirección de calidad, lo cual permite estructurar, organizar, almacenar y comunicar el conocimiento generado en el subproceso de prueba.

Según (Vitelli, 2011) para la especificación de la Ontología se crea una plantilla con el nombre Documento de Especificación de Requerimientos. En esta plantilla se coloca el dominio al que se refiere

la Ontología, fecha en que comienza el desarrollo, quienes son los desarrolladores, cual es el propósito, que nivel de formalidad alcanzara la Ontología, su alcance especificando las preguntas de competencia y cuáles serán las fuentes de conocimiento. A continuación, se muestra la plantilla para la propuesta ontológica.

Documento de especificación de requerimientos	
Dominio	Subproceso de Pruebas de software
Fecha	3 de febrero del 2018
Desarrollador(es)	Carlos Parker Leiva
Propósito	Desarrollar una ontología que apoye el subproceso de pruebas de software en la UCI, permitiendo la organización y comunicación del conocimiento generado,
Nivel de Formalidad	Formal
Alcance	Preguntas de Competencia: • ¿Qué herramientas manipula el revisor a la hora de realizar la detección de las no conformidades? • ¿Qué tipos de prueba corresponden a una determinada característica de calidad? • ¿De qué tipo es una determinada no conformidad? • ¿Qué no conformidades presento un determinado artefacto de proyecto y cuál es su clasificación? • ¿Qué impacto tiene una determinada no conformidad? • ¿Que módulos tiene un determinado producto? • ¿Que módulos pertenecen a un determinado requisito? • ¿Qué no conformidad detectó un determinado revisor? • ¿Cuál es el estado de una no conformidad? • ¿Qué documentación requiere un determinado artefacto de proyecto para ser analizado? • ¿A qué proyecto pertenece un determinado especialista? • ¿Cuál es la clasificación de una determinada característica de calidad acorde a los tipos de prueba? • ¿Qué no conformidades presento un proyecto?
Fuentes de Conocimiento	Expertos de empresas como XETID, Calisoft y de la Dirección de Calidad UCI (Entrevista y encuesta)

2.2.2 Actividad: Conceptualización

Methontology recomienda realizar una serie de tareas de conceptualización en un orden determinado: (Vitelli, 2011)

1. Construcción del glosario de términos.
2. Construcción de la taxonomía de conceptos.
3. Construcción del diagrama de relaciones binarias.
4. Construcción del diccionario de conceptos.

5. Describir relaciones binarias.
6. Describir atributos de instancias
7. Describir atributos de clases
8. Describir constantes
9. Describir axiomas formales
10. Describir reglas
11. Describir Instancias.

Glosario de Términos

El glosario de términos incluye todos los términos relevantes del dominio dígase (conceptos, instancias y relaciones entre conceptos).

Tabla 1 Glosario de términos (Conceptos)

No.	Conceptos	Descripción
1	Herramientas	Recurso que se utiliza para realizar el subproceso
2	Característica de Calidad	Bases sobre las cuales se edifica la aptitud de un producto.
3	Funcionalidad	Es la capacidad del producto software para proporcionar funciones declaradas e implícitas cuando se usa bajo condiciones especificadas.
4	Confiabilidad	Es la capacidad del producto de software para mantener un nivel de ejecución especificado cuando se usa bajo las condiciones especificadas
5	Usabilidad	Es la capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones especificadas.
6	Eficiencia	Es la capacidad del producto software para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas.
7	Portabilidad	Es la capacidad del producto software para ser transferido de un entorno a otro.
8	Control de estado de no conformidad	Controla el estado en el que se va a encontrar una no conformidad
9	Tipos de Pruebas	Clasificación de las Pruebas.
10	Función	Consisten en la revisión de los requisitos aceptados por el cliente contra las funcionalidades presentes en la aplicación.
11	Seguridad	Asegurar que los datos o el sistema solamente es accedido por los actores definidos según niveles de acceso. En 3 niveles.
12	Volumen	Enfocada en verificar las habilidades de los programas para manejar grandes cantidades de datos, tanto como entrada, salida o residente en la BD.
13	Recuperación y tolerancia a fallas	Verificar que los procesos de recuperación (manual o automática) restauran apropiadamente la base de datos, aplicaciones y sistemas, y los llevan a un estado conocido o deseado.
14	Comparativa de código:	Se comparan dos ficheros, o partes del código, de manera que se puedan identificar los cambios que han sido realizados en el artefacto.
15	Usabilidad	Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.
16	Estructura	Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.

No.	Conceptos	Descripción
17	Contención	Enfocada a la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso (registro de recursos, memoria, etc.).
18	Carga	Usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales.
19	Estrés	Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible)
20	Rendimiento	Enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes.
21	Configuración	Enfocada a asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.
22	Instalación	Enfocada a asegurar la instalación en diferentes configuraciones de hardware y software bajo diferentes condiciones (insuficiente espacio en disco, etc.).
23	Estado de no conformidad	Estado en el que se encuentra una no conformidad dígase pendiente o resuelta.
24	Pendiente	Estado en el que se encuentra la no conformidad antes de ser resuelta..
25	Resuelta	Estado en el que se encuentra la no conformidad después de resolverse.
26	Estrategia	Describe el enfoque y los objetivos generales de las tareas de prueba,
27	Impacto	Impacto que tendrá una determinada no conformidad en el proyecto,
28	Módulo	Parte de las tareas que debe cumplir un producto de software
29	Revisor	Persona que se encarga de la revisión de las no conformidades,
30	Nivel de Prueba	Diferentes momentos dentro del desarrollo del software en que una prueba puede realizarse.
31	Aceptación	Su propósito es confirmar que el sistema está terminado y que es aceptado por los usuarios finales.
32	Integración	Su objetivo es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente
33	Interna	Subclase de Nivel de Prueba
34	Liberación	Subclase de Nivel de Prueba
35	Sistema	Su objetivo es verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones apropiadas funcionando como un todo.
36	Unidad	El objetivo es comprobar que el módulo, entendido como una unidad funcional, está correctamente codificado.
37	Especialista	Persona que utiliza herramienta, modifica artefactos y ejecuta actividades.
38	Requisito	Requisito probado
39	Tipo No Conformidad	Representa el tipo de no conformidad.
40	Documentación	Documentos generados durante el subproceso.
41	Artefacto Proyecto parte	Parte de los elementos utilizados
42	Descripción de las variables	Subclase de Artefacto de proyecto parte
43	Descripción de los escenarios	Subclase de Artefacto de proyecto parte
44	Artefacto Proyecto	Elementos que se utilizan de apoyo a las pruebas,

No.	Conceptos	Descripción
45	Instalador de la aplicación	Subclase de Artefacto de proyecto
46	Portal Web	Subclase de Artefacto de proyecto
47	Aplicación Web	Subclase de Artefacto de proyecto
48	Aplicación de escritorio	Subclase de Artefacto de proyecto
49	Multimedia	Subclase de Artefacto de proyecto
50	Sistemas Operativos	Subclase de Artefacto de proyecto
51	Juegos	Subclase de Artefacto de proyecto
52	Documento de caso de prueba	Subclase de Artefacto de proyecto
53	No Conformidad	Inconformidades e inconsistencia que genera el producto
54	Proyecto	Conjunto de actividades que se encuentran interrelacionadas y coordinadas por recursos humanos.
55	Producto	Producto al que se le realiza el subproceso de prueba

Tabla 2.Glosario de Términos(Relaciones)

No.	Relaciones	Descripción
1	tiene_tipos_de_prueba	Relación para determinar los tipos de prueba que corresponden a una determinada características de calidad
2	tiene tipo de prueba	Relación para determinar en qué tipo de prueba se detectó la no conformidad
3	tiene tipo de no conformidad	Relación para determinar el tipo de una no conformidad
4	tiene revisor	Relación para determinar el revisor que detectó la no conformidad
5	tiene requisito	Relación para determinar a qué requisito le corresponde la no conformidad
6	tiene requisito2	Relación para determinar si el requisito fue probado
7	tiene proyecto	Relación para determinar el proyecto que tiene un determinado producto
8	tiene producto	Relación para determinar el módulo que tiene un producto
9	tiene no conformidad	Relación para controlar el estado de una no conformidad
10	tiene nivel de prueba	Relación para determinar en qué nivel de prueba se detectó la no conformidad.
11	tiene módulo	Relación para determinar los requisitos que pertenecen a un determinado módulo
12	tiene impacto	Relación para determinar el impacto que puede tener una no conformidad
13	tiene estado de no conformidad	Relación para determinar el estado de una no conformidad
14	tiene especialista	Relación para determinar los especialistas que tiene un proyecto
15	tiene artefacto proyecto parte	Relación para determinar la parte del artefacto donde se detectó la no conformidad
16	tiene artefacto proyecto	Relación para determinar artefacto de proyecto que tiene una no conformidad
17	manipula herramienta	Relación para determinar la herramienta que manipula un determinado especialista
18	es documentación de	Relación para determinar la documentación de un artefacto de proyecto

Tabla 3.Glosario de Términos(Instancias)

No.	Instancias	Descripción
1	ap1	Instancia perteneciente a la clase Artefacto de Proyecto
2	apparte1	Instancia perteneciente a la clase Artefacto Proyecto parte
3	cc1	Instancia perteneciente a la clase Característica de Calidad
4	controlestado1	Instancia perteneciente a la clase Control de Estado de No Conformidad
5	documentación1	Instancia perteneciente a la clase Documentación
6	especialista1	Instancia perteneciente a la clase Especialista
7	estado1	Instancia perteneciente a la clase Estado de No Conformidad
8	estrategia1	Instancia perteneciente a la clase Estrategia
9	herramienta1	Instancia perteneciente a la clase Herramienta
10	imp1	Instancia perteneciente a la clase Impacto
11	módulo1	Instancia perteneciente a la clase Módulo
12	nivelp1	Instancia perteneciente a la clase Nivel de Prueba
13	nc1	Instancia perteneciente a la clase No Conformidad
14	producto1	Instancia perteneciente a la clase Producto
15	proyecto1	Instancia perteneciente a la clase Proyecto
16	r1	Instancia perteneciente a la clase Requisito
17	rp1	Instancia perteneciente a la clase Requisito Probado
18	revisor1	Instancia perteneciente a la clase Revisor
19	tipop1	Instancia perteneciente a la clase Tipo de prueba
20	tnc1	Instancia perteneciente a la clase Tipo de No Conformidad

Taxonomía de Conceptos

Las taxonomías representan mediante las clases y las instancias de esas clases una realidad contextualizada que permite la organización y recuperación de los conceptos representados. Para la construcción de la taxonomía de conceptos, se seleccionan del glosario de términos aquellos términos que son conceptos, en la Figura 2 se muestra el primer nivel (Ver anexo 3 para taxonomía completa) Dicha tarea se puede llevar a cabo de tres maneras: **top-down** que no es más que ir definiendo desde los conceptos más generales hasta los más específicos dentro del dominio. **Bottom-up** es el proceso inverso, es decir, ir desde lo más específico a lo más general. La tercera vía es la mezcla de los dos procesos anteriores, donde primeramente se detallan los conceptos más relevantes y luego se generalizan y especifican adecuadamente.

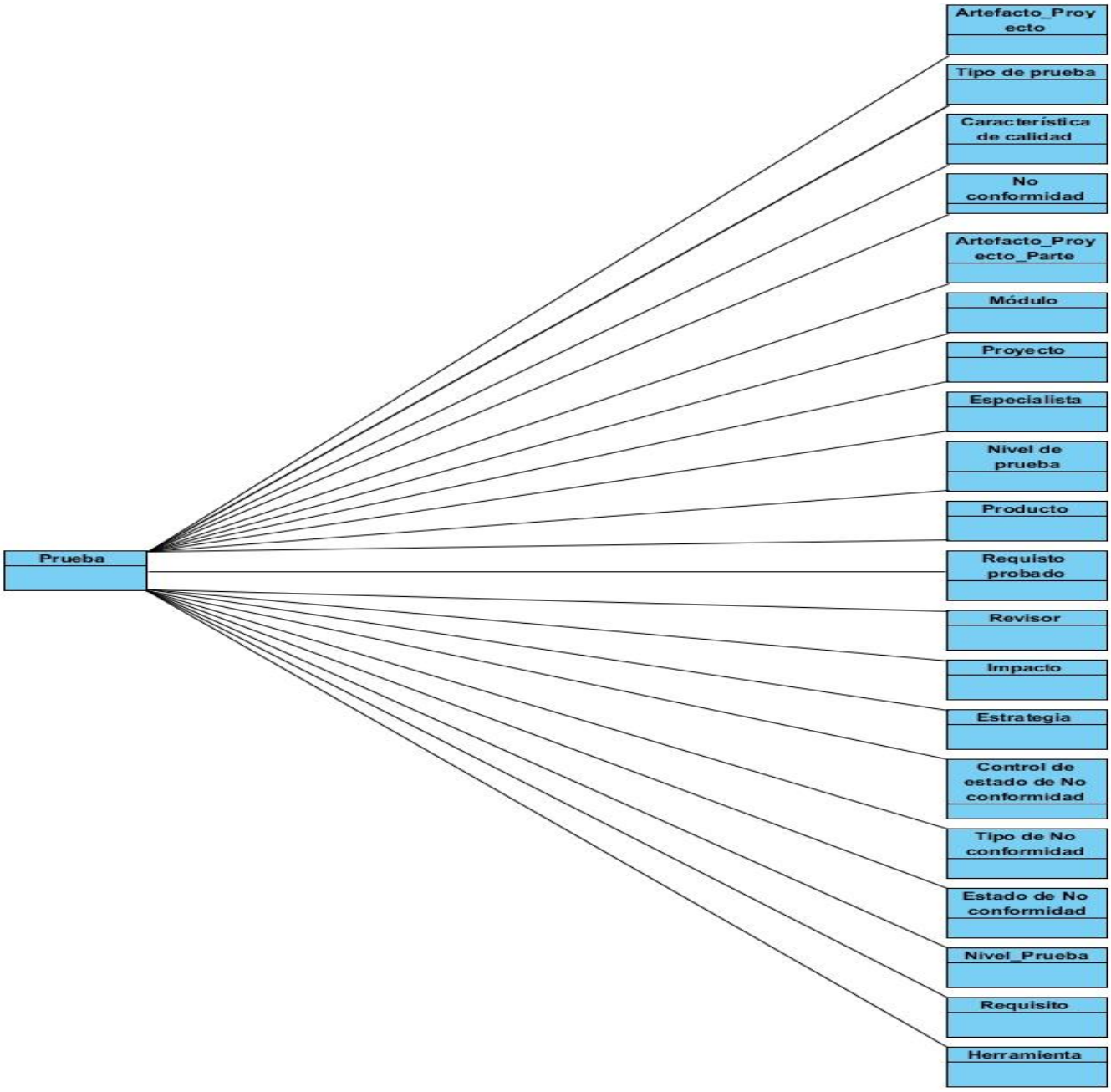


Figura 2.Descripción del subproceso de prueba.

Diagrama de Relaciones Binarias

Después de haber construido la taxonomía, se construyen los diagramas de relaciones binarias, que tiene como objetivo establecer las relaciones existentes entre clases. Con el modelo conceptual mostrado anteriormente en la figura 1, se puede constatar que todas las relaciones que existen son relaciones binarias.

En la figura 3 se presenta un fragmento de un diagrama de relaciones binarias inversa de uno de los conceptos del diagrama, con las relaciones “manipula” y “es manipulada por”. Estas relaciones conectan los conceptos raíces de las taxonomías de conceptos de entidades “Herramienta” y “Especialista”.

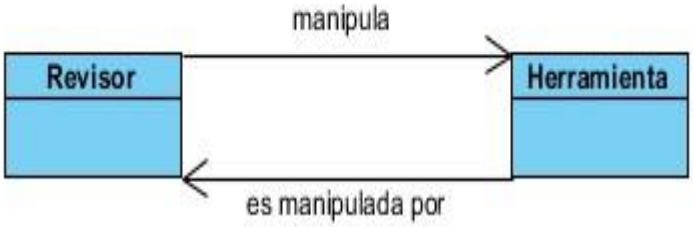


Figura 3. Relación binaria.

Diccionario de Conceptos

Después de haber generado las taxonomías de conceptos y haberse construido el diagrama de relaciones binarias, se define cuáles son las propiedades que describen cada concepto de la taxonomía y las instancias de cada uno de los mismos. El diccionario de conceptos contiene todos los conceptos del dominio, sus relaciones e instancias.

Tabla 4.Diccionario de conceptos.

No.	Conceptos	Instancias	Relaciones
1	Herramientas	herramienta1, herramienta2, herramienta3, herramienta4	manipula_herramienta
2	Característica de Calidad	cc1, cc2, cc3, cc4, cc5	tiene_tipos_de_prueba
3	Control de estado de no conformidad	controlestado1	tiene_no_conformidad
4	Estado de no conformidad	estado1, estado2	tiene_estado_de_no_conformidad
5	Estrategia	estrategia1, estrategia2, estrategia3	
6	Impacto	imp1, imp2, imp3	tiene_impacto
7	Módulo	módulo1, módulo2, módulo3. Módulo4, módulo5, módulo6, módulo7, módulo8, módulo9, módulo10, módulo11, módulo12, módulo13	tiene_módulo
8	Revisor	revisor1, revisor2, revisor3, revisor4, revisor5	tiene_revisor

No.	Conceptos	Instancias	Relaciones
9	Nivel de Prueba	nivelp1, nivelp2, nivelp3, nivelp4, nivelp5, nivelp6	tiene_nivel_de_prueba
10	Especialista	especialista1, especialista2, especialista3, especialista4	tiene_especialista, manipula_herramienta
11	Requisito	r1,r2, r3, r4, r5, r6, r7, r8, r9, r10 ,r11, r12,r13, r14, r15, r16, r17, r18, r19, r20, r21, r22, r23, r24, r25, r26, r27, r28, r29, r30, r31, r32, r33, r34, r35, r36, r37, r38, r39, r40, r41, r42, r43, r44, r45, r46, r47, r48, r49, r50, r51, r52, r53, r54, r55, r56, r57, r58, r59, r60, r61, r62, r63, r64, r65	tiene_requisito, tiene_requisito2, tiene_módulo,
12	Tipo No Conformidad	tnc1, tnc2, tnc3, tnc4, tnc5	tiene_tipo_de_no_conformidad
13	Documentación	documentación1, documentación2, documentación3, documentación4, documentación5, documentación6, documentación7, documentación8, documentación9, documentación10.	es_documentación_de
14	Artefacto Proyecto parte	apparte1, apparte2	tiene_artefacto_proyecto_parte
15	Artefacto Proyecto	ap1, ap2, ap3, ap4, ap5, ap6, ap7, ap8	tiene_artefacto_proyecto, es_documentación_de
16	No Conformidad	nc1, nc2, nc3, nc4, nc5, nc6, nc7, nc8, nc9, nc10, nc11, nc12, nc13, nc14, nc15, nc16, nc17, nc18, nc19, nc20, nc21, nc22	tiene_tipo_de_prueba, tiene_tipo_de_no_conformidad, tiene_revisor, tiene_requisito, tiene_no_conformidad, tiene_nivel_de_prueba, tiene_impacto, tiene_estado_de_no_conformidad ,tiene_artefacto_proyecto_parte, tiene_artefacto_proyecto
17	Proyecto	proyecto1, proyecto2, proyecto3	tiene_proyecto, tiene_especialista
18	Producto	producto1, producto2, producto3	tiene_proyecto, tiene_producto

Describir relaciones binarias

El objetivo de esta tarea es describir en detalle todas las relaciones binarias. Para cada relación binaria se específica su nombre, los nombres de sus conceptos origen y destino, su cardinalidad y su relación inversa.

Tabla 5 Descripción de relaciones binarias

Nombre de la relación	Concepto origen	Cardinalidad máxima	Concepto destino	Relación inversa
es_documentación_de	Documentación	N	Artefacto_Proyecto	documentación_necesaria
manipula	Revisor	N	Herramienta	es_manipulada_por
tiene_artefacto_ proyecto	No_conformidad	N	Artefacto_Proyecto	es_artefacto_de_proyecto_de
tiene_artefacto_ proyecto_parte	No_conformidad	N	Artefacto_Proyecto_Parte	es_artefacto_de_proyecto_parte_ de
tiene_especialista	Proyecto	N	Especialista	pertenece_al
tiene_estado_de_no_ conformidad	No_conformidad	N	Estado_de_No_ Conformidadad	es_estado_de_ no_ conformidad
tiene_impacto	No_conformidad	N	Impacto	es_impacto_de
tiene_módulo	Requisito	N	Módulo	responde_a
tiene_nivel_de_ prueba	No_conformidad	N	Nivel_de_Prueba	es_nivel_de_ prueba_de
tiene_no_ conformidad	Control_de_ estado_de_No_ Conformidad	N	No_conformidad	es_no_ conformidad_de
tiene_producto	Módulo	N	Producto	es_producto_de
tiene_proyecto	Producto	N	Proyecto	es_proyecto_de
tiene_requisito	No_conformidad	N	Requisito	es_requisito_de
tiene_requisito2	Requisito_ Probado	N	Requisito	es_del_requisito_probado
tiene_revisor	No_conformidad	N	Revisor	es_revisor_de
tiene_tipo_de_no_ conformidad	No_conformidad	N	Tipo_No_ Conformidad	es_un_tipo_de_ no_conformidad_de
tiene_tipo_de_ prueba	No_conformidad	N	Tipo_de_prueba	es_tipo_de_ prueba_de
tiene_tipos_de_ prueba	Característica_ de_Calidad	N	Tipo_de_prueba	corresponde_a_la

Describir atributos de instancias

En esta tarea se crea la tabla de atributos de instancias en la que se describe detalladamente todos los atributos de instancias incluidos en el diccionario de conceptos. Para cada atributo de instancia, se debe especificar: nombre, concepto al que pertenece, tipo de valor, rango de valores (en el caso de valores numéricos) y cardinalidad.

Tabla 6 Descripción de atributos de instancias

No.	Nombre del atributo de instancia	Concepto	Tipo de valor	Rango	Cardinalidad
1	analista_de_requisito	Requisito	string	-	1…1
2	analista_lider_de_ requisito	Requisito	string	-	1…1
3	area_del_revisor	Revisor	string	-	1…1
4	consecuencia	Característica_de_ Calidad	string	-	1…1
5	denominación_de_ artefacto_proyecto	Artefacto_ Proyecto	string	-	1…1
6	denominación_de_ artefacto_proyecto_ parte	Artefacto_Proyecto_Parte	string	-	1…1
7	denominación_de_ característica_de_ calidad	Característica_de_ Calidad	string	-	1…1
8	denominación_de_ documentación	Documentación	string	-	1…1
9	denominación_de_ estado_de_no_ conformidad	Estado_de_No_ Conformidadad	string	-	1…1
10	denominación_del_ impacto	Impacto	string	-	1…1
11	denominación_de_ la_estrategia	Estrategia	string	-	1…1
12	denominación_de_la_herramienta	Herramienta	string	-	1…1
13	denominación_de_ módulo	Módulo	string	-	1…1
14	denominación_de_ nivel_de_prueba	Nivel_de_ Prueba	string	-	1…1
15	denominación_de_ requisito	Requisito	string	-	1…1
16	denominación_de_ tipo_de_ no_conformidad	Tipo_No_ Conformidad	string	-	1…1
17	denominación_de_ tipo_de_prueba	Tipo_de_prueba	string	-	1…1
18	denominación_del_ producto	Producto	string	-	1…1
19	denominación_del_ proyecto	Proyecto	string	-	1…1
20	descripción_de_ artefacto_proyecto	Artefacto_ Proyecto	string	-	1…1
21	descripción_de_ característica_de_calidad	Característica_de_ Calidad	string	-	1…1
22	descripción_de_ estado_de_no_ conformidad	Estado_de_No_ Conformidadad	string	-	1…1
23	descripción_del_ impacto	Impacto	string	-	1…1
24	descripción_de_la_ estrategia	Estrategia	string	-	1…1
25	descripción_de_la_ herramienta	Herramienta	string	-	1…1
26	descripción_de_ módulo	Módulo	string	-	1…1
27	descripción_de_ nivel_de_prueba	Nivel_de_ Prueba	string	-	1…1
28	descripción_de_no_ conformidad	No_ Conformidad	string	-	1…1
29	descripción_de_ requisito	Requisito	string	-	1…1
30	descripción_de_tipo_de_no_conformidad	Tipo_No_ Conformidad	string	-	1…1
31	descripción_de_tipo_de_prueba	Tipo_de_prueba	string	-	1…1

No.	Nombre del atributo de instancia	Concepto	Tipo de valor	Rango	Cardinalidad
32	descripción_del_ especialista	Especialista	string	-	1…1
33	descripción_del_ producto	Producto	string	-	1…1
34	descripción_del_ proyecto	Proyecto	string	-	1…1
35	descripción_del_ revisor	Revisor	string	-	1…1
36	no_iteración_de_la_ no_conformidad	No_ Conformidad	integer	-	1…1
37	no_iteración_ requisito_probado	Requisito_ Probado	integer	-	1…1
38	nombre_del_especialista	Especialista	string	-	1…1
39	nombre_del_revisor	Revisor	string	-	1…1
40	responsabilidad_del_especialista	Especialista	string	-	1…1
41	resultado_de_la_ revisión	Requisito_ Probado	string	-	1…1
42	versión_de_la_ herramienta	Herramienta	double	-	1…1

Describir atributos de clases

En esta tarea se describen cada uno de los atributos de clase que se encuentran en el diccionario de conceptos. Los conceptos a manejar en la investigación solo se identifican mediante las instancias/individuos presentes en cada uno de ellos, por lo que no es necesario emplear atributos para caracterizar los conceptos, razón por la cual este aspecto no se encuentra reflejado en el diccionario de conceptos

Describir constantes

Esta tarea tiene como objetivo describir cada una de las constantes identificadas en el glosario de términos, en este caso solo fueron identificados términos de tipo concepto, relaciones e instancias. En la propuesta ontológica a desarrollar, no se utilizan las constantes, debido a que el conocimiento almacenado está estructurado a través de los conceptos identificados y sus relaciones, los cuales no hacen uso de constantes.

Describir axiomas formales

En esta actividad se deben identificar los axiomas formales que son necesarios en la Ontología y describirlos de manera precisa. Los axiomas formales no son más que expresiones lógicas siempre verdaderas que se utilizan para definir restricciones en la ontología. En la propuesta ontológica no se identificaron axiomas formales, razón por la cual esta tarea no se realiza.

Describir reglas

En esta tarea se debe identificar en primer lugar qué reglas o axiomas se necesitan en la ontología, y entonces describirlas en la tabla. En la presente investigación solo fueron identificadas reglas. Para cada

regla, Methontology propone incluir la siguiente información: nombre, descripción en lenguaje natural, expresión que describe formalmente la regla, conceptos y relaciones utilizados en la regla. Además, propone especificar las expresiones de las reglas utilizando el formato *si <condiciones> entonces <consecuente>*. La parte izquierda de la regla es una conjunción de condiciones simples, mientras que la parte derecha es una simple expresión de un valor de la ontología.

Tabla 7. Reglas

Nombre	Descripción	Expresión	Conceptos	Relaciones
Artefacto_Proyecto Aplicación_escritorio	El artefacto de proyecto Aplicación de escritorio para ser analizado requiere de alguna documentación y es artefacto de proyecto de alguna no conformidad	Artefacto_Proyecto and (documentación_necesaria some Documentación) and (es_artefacto_de_proyecto _de some No_conformidad)	Artefacto_Proyecto No_conformidad	documentación_necesarias_artefacto_de_proyecto _de
Módulo Requisitos	Un módulo responde a algunos requisitos	Módulo and (responde_a some requisitos)	Módulo Requisito	responde_a
Revisor Herramienta	Un revisor manipula alguna herramienta	Revisor and (manipula some Herramienta)	Revisor Herramienta	manipula
Característica de Calidad Eficiencia	A la característica de calidad eficiencia le corresponde algún tipo de prueba	Característica_de_Calidad and (tiene_tipos_de_prueba some Tipo_de_prueba)	Característica de Calidad tipos de prueba	tiene_tipos_de_prueba

Descripción de las Instancias

En la tabla 8, se describen las instancias con su nombre, concepto al que pertenece y valores o atributos que presentan.

Tabla 8.Descripción de las instancias

Nombre	Concepto al que pertenece	Valor o Atributo
tipop1	Tipo_de_prueba	Denominación: Función Descripción: Consiste en la revisión de los requisitos aceptados por el cliente contra las funcionalidades presentes en la aplicación.
tipop2	Tipo_de_prueba	Denominación: Seguridad Descripción: Asegurar que los datos o el sistema solamente es accedido por los actores definidos según niveles de acceso. En 3 niveles.
tipop3	Tipo_de_prueba	Denominación: Volumen Descripción: Enfocada en verificar las habilidades de los programas para manejar grandes cantidades de datos, tanto como entrada, salida o residente en la BD.
tipop4	Tipo_de_prueba	Denominación: Recuperación y tolerancia a fallas Descripción: Verificar que los procesos de recuperación (manual o automática) restauran apropiadamente la base de datos, aplicaciones y sistemas, y los llevan a un estado conocido o deseado.
tipop5	Tipo_de_prueba	Denominación: Comparativa de código Descripción: Se comparan dos ficheros, o partes del código, de manera que se puedan identificar los cambios que han sido realizados en el artefacto.
tipop6	Tipo_de_prueba	Denominación: Usabilidad Descripción: Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.

Nombre	Concepto al que pertenece	Valor o Atributo
tipop7	Tipo_de_prueba	Denominación: Estructura Descripción: Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.
tipop8	Tipo_de_prueba	Denominación: Contención Descripción: Enfocada a la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso (registro de recursos, memoria, etc.).
tipop9	Tipo_de_prueba	Denominación: Carga Descripción: Usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales.
tipop10	Tipo_de_prueba	Denominación: Estrés Descripción: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible)
tipop11	Tipo_de_prueba	Denominación: Rendimiento Descripción: Enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes.
tipop12	Tipo_de_prueba	Denominación: Configuración Descripción: Enfocada a asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.

Nombre	Concepto al que pertenece	Valor o Atributo
tipop13	Tipo_de_prueba	Denominación: Instalación Descripción: Enfocada a asegurar la instalación en diferentes configuraciones de hardware y software bajo diferentes condiciones (insuficiente espacio en disco, etc.).

2.2.3 Actividad: Formalización

La actividad de formalización es la encargada de la transformación del modelo conceptual obtenido en la anterior actividad en un modelo formal o semi-computable. Para la transformación se utilizó el editor Protégé en su versión 5.2 (Ver figura 4).

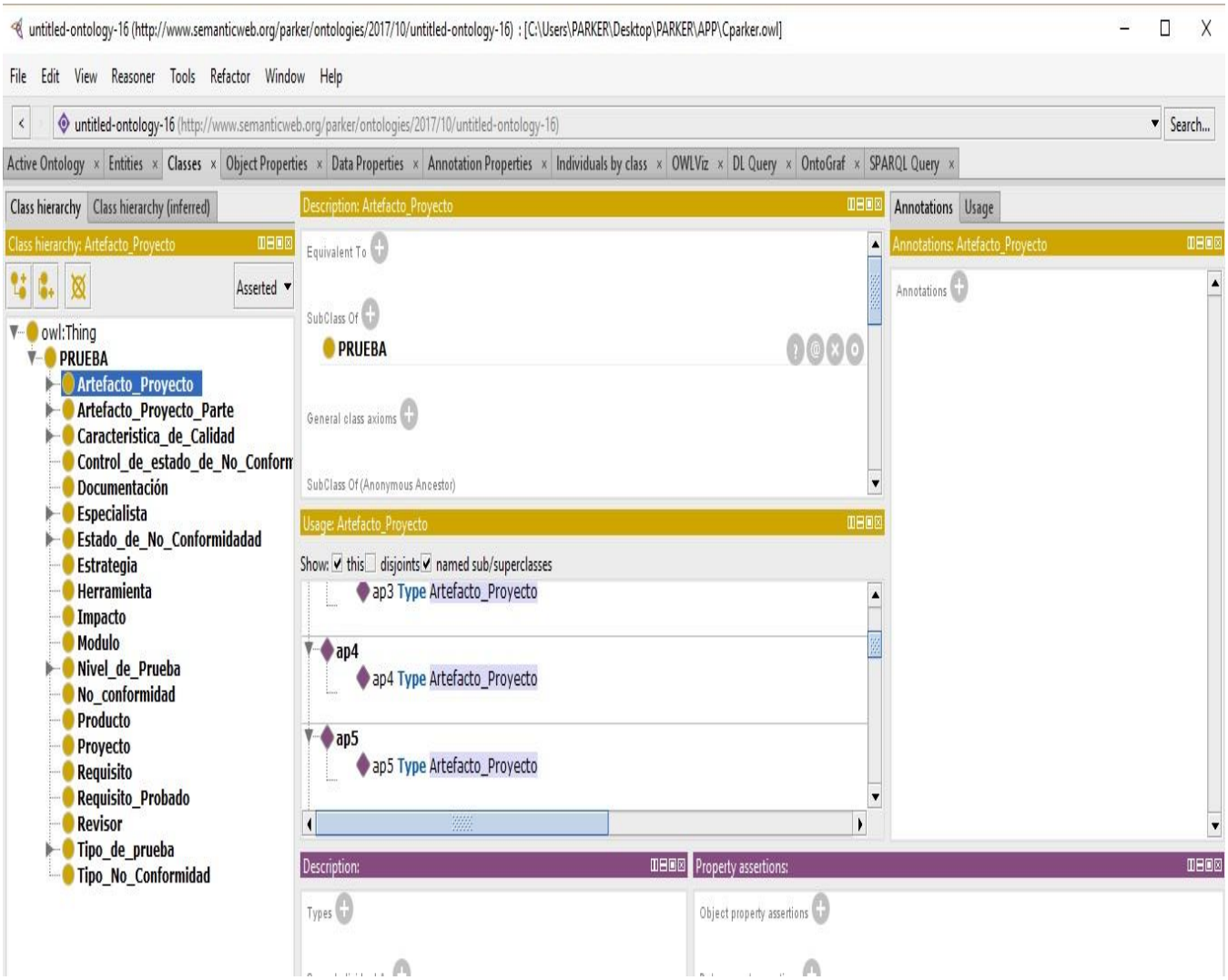


Figura 4.Interfaz Principal

Creación de las Clases

Las Clases son creadas partiendo desde la Clase Padre “Thing”, palabra clave que hace referencia a la W3C. Cada clase presentará: sub-clases, instancias, restricciones, anotaciones, clases equivalentes entre otros elementos necesarios. La jerarquía de clases se muestra en la figura 5.

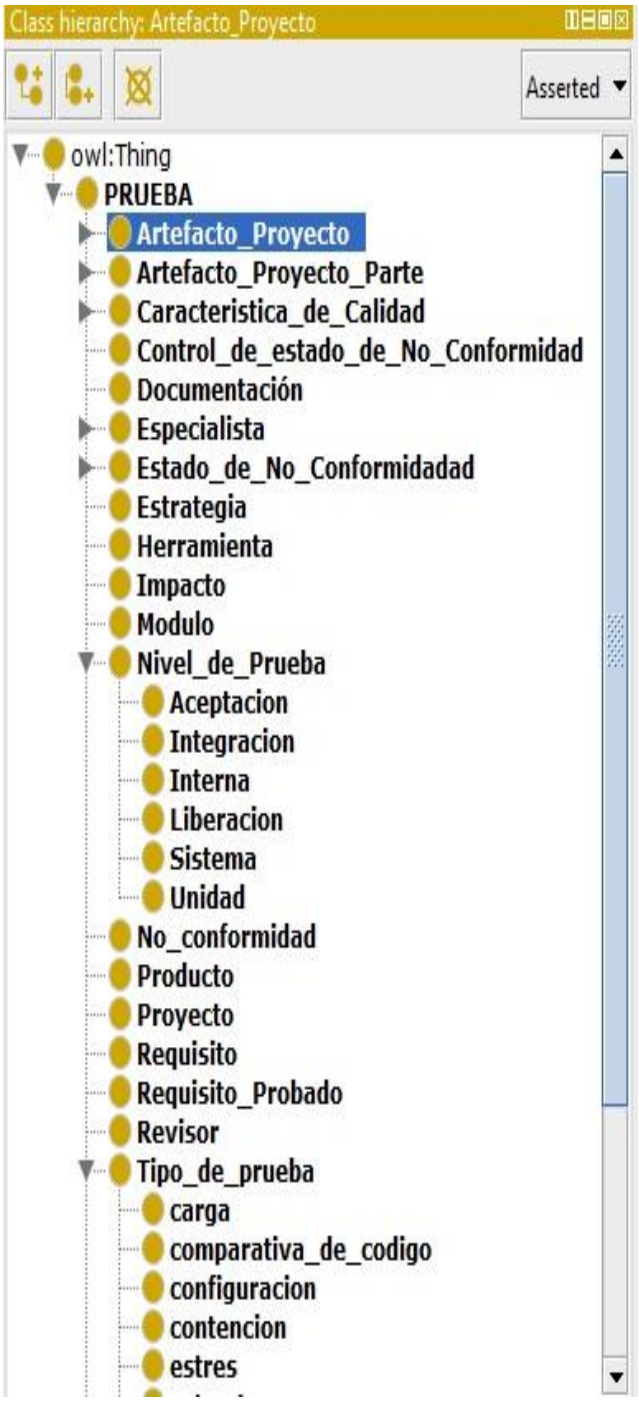


Figura 5.Jerarquía de clases creada en Protégé.

.

Creación de las Propiedades

Las clases no proveen la información suficiente para dar respuesta a las preguntas de competencia, a su vez las propiedades permiten constituir las relaciones, dado que estas representan las cualidades internas de los conceptos y constituyen las propiedades distintivas de los objetos. Las mismas pueden ser de 2 tipos, "**object properties**" y "**data properties**". En la figura 6 se muestran las relaciones "object properties" y en la figura 7 los "datatype properties" que están presentes en la propuesta ontológica.

La manera de especificar las propiedades que relacionan dos clases dadas es mediante la definición del rango y el dominio de la misma. El dominio es el conjunto de clases que tendrá la propiedad (Ej. "Característica_de_Calidad" que es la clase que inicia la relación: "tiene_tipos_de_prueba"); mientras que el rango es la clase que se implica en la relación en este caso "Tipo_de_prueba".

Figura 6.Relaciones "object properties" presentes en la Ontología.

Los data property son las propiedades que relaciona un individuo con un valor de tipo dato. Por ejemplo, el individuo “tipop1” perteneciente a la clase Tipo_de_prueba que tiene como datos, “denominación” y “descripción”.

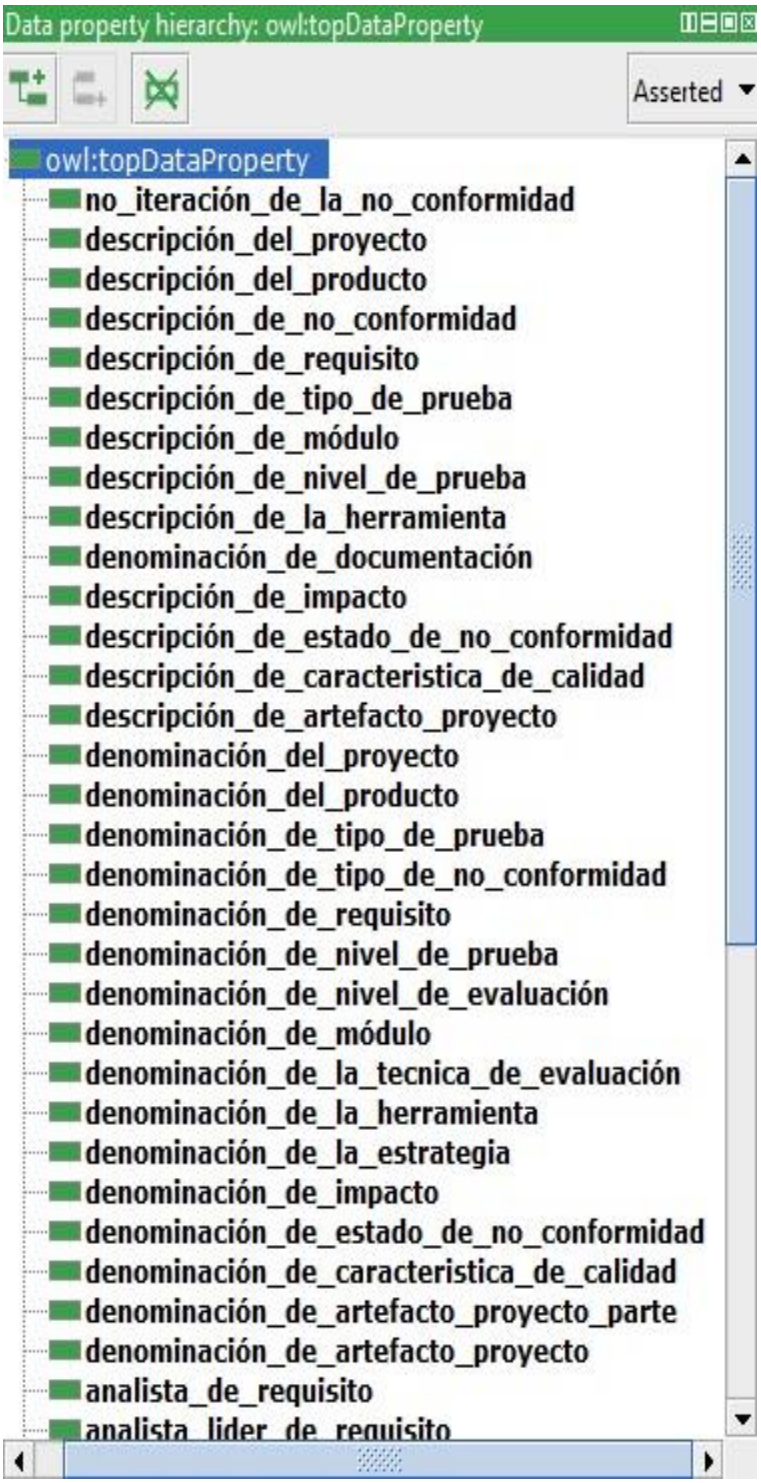


Figura 7.Descripción de los data properties.

Creación de las Instancias

Las instancias son indispensables dentro de un sistema de almacenamientos ya que mediante las mismas se efectúa gran parte del proceso de razonamiento, además de que muestran la funcionalidad del sistema. En esta propuesta se crearon instancias en varias entidades con el objetivo de comprobar las preguntas definidas (Ver figura 8).

Figura 8.Creación de las instancias.

Creación de las reglas

Para la definición de las reglas se utilizan las relaciones y las clases existentes en la ontología, cuantificándolas de forma universal y existencial (Ver Figura 9).

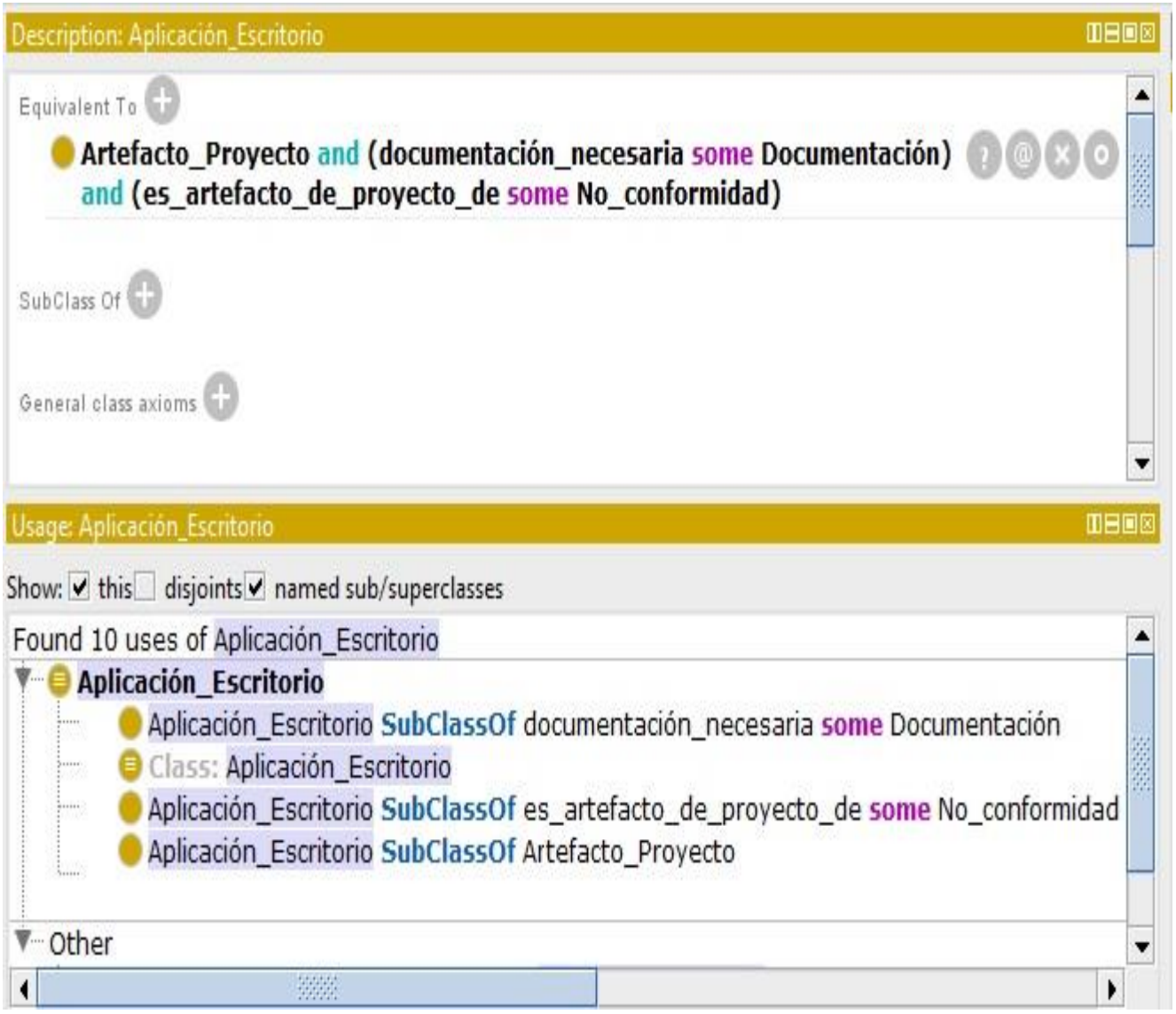


Figura 9 Creación de las reglas en Protégé.

Obtención del grafo de la ontología propuesta

OWLViz es un plugin para Protégé cuya función es permitir la visualización de los conceptos y relaciones creadas mediante grafos, además de este plugin se debe tener instalado el GraphViz. (Ver Figura 10)

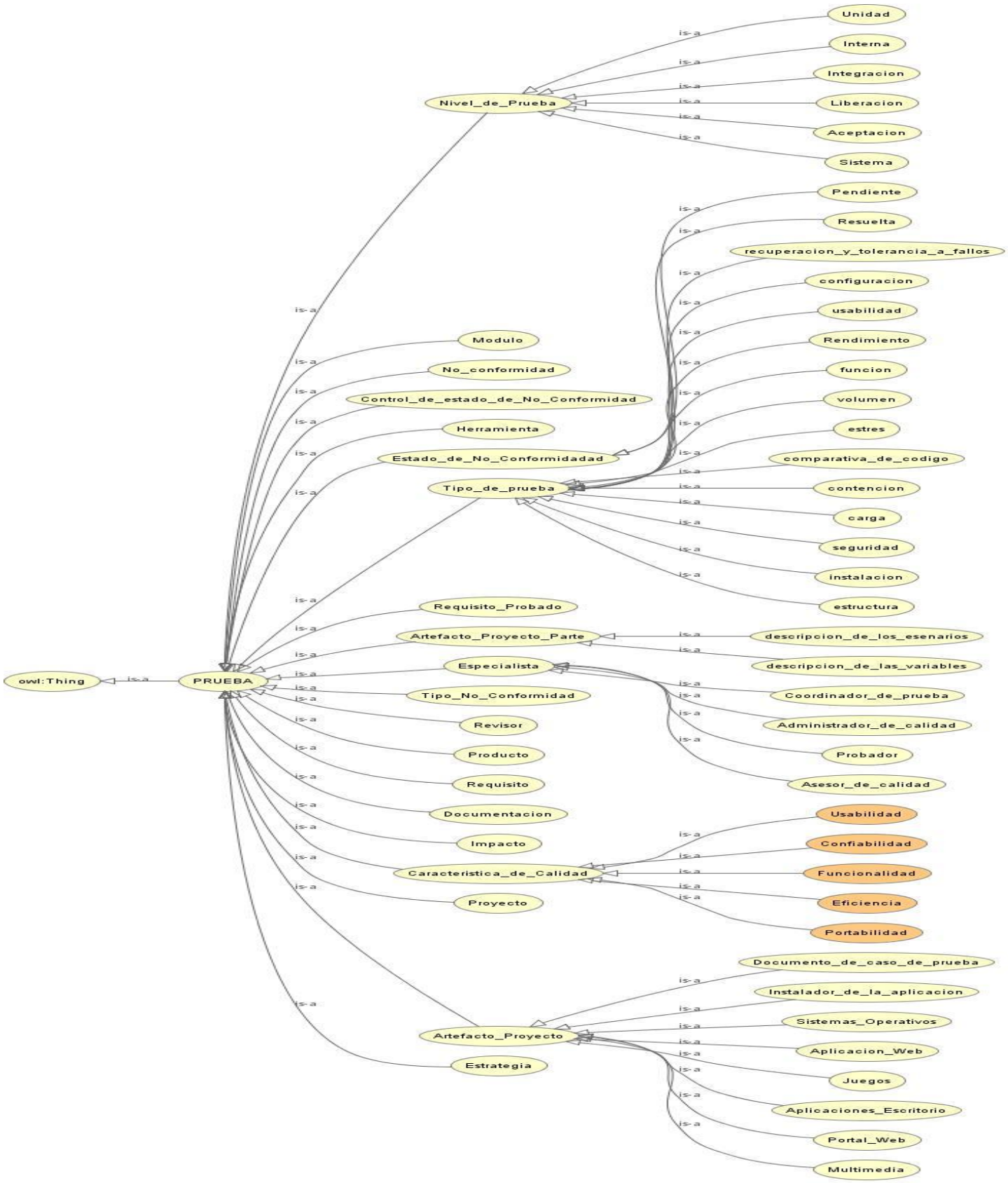


Figura 10.Grafo de la ontología generado en Protégé (OWLViz).

2.2.4 Actividad: Implementación

OWL facilita un lenguaje para definir ontologías estructuradas basadas en Web con esto procura proporcionar un lenguaje para describir clases, las propiedades de las clases y la relación entre las clases. OWL está construida sobre RDF, por lo que ofrece una base apropiada para desarrollar ontologías.

Una vez finalizada la ontología, existen varios lenguajes en los que se puede exportar, en este caso por requerimientos del cliente se exportó en el lenguaje OWL, a continuación, se muestran fragmentos donde se describe la manera en que se codifican los datos usando este lenguaje de ontologías.

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

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

La declaración de las clases se hace de la siguiente manera:

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

<Declaration>

<Class IRI="#Herramienta"/>
</Declaration>

<Declaration>
<Class IRI="#Especialista"/>
</Declaration>

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

Donde se declaran las clases Especialista y herramienta.

La declaración de las subclases de una clase, se realiza de la siguiente manera:

```
<SubClassOf>
 <Class IRI="#Confiabilidad"/>
<Class IRI="#Característica_de_Calidad"/>
</SubClassOf>

 <SubClassOf>
<Class IRI="#Eficiencia"/>
 <Class IRI="#Característica_de_Calidad"/>
</SubClassOf>

<SubClassOf>
 <Class IRI="#Funcionalidad"/>
<Class IRI="#Característica_de_Calidad"/>
</SubClassOf>

<SubClassOf>
 <Class IRI="#Portabilidad"/>
<Class IRI="#Característica_de_Calidad"/>
</SubClassOf>

<SubClassOf>
 <Class IRI="#Usabilidad"/>
<Class IRI="#Característica_de_Calidad"/>
</SubClassOf>
```

Donde las clases confiabilidad, eficiencia, funcionalidad, portabilidad y usabilidad son subclases de la clase Característica_de_Calidad.

La declaración de las relaciones se realiza de la siguiente manera:

```
 <Declaration>
 <ObjectProperty IRI="#tiene_tipos_de_prueba"/>
 </Declaration>

<Declaration>
 <ObjectProperty IRI="#tiene_impacto"/>
</Declaration>
```

Dónde se describe las relaciones tiene_tipos_de_prueba que relaciona la clase Característica_de_Calidad con la clase Tipo_de_prueba y tiene_impacto que relaciona la clase No_conformidad con la clase Impacto.

Una de las ventajas del uso de la herramienta Protégé es su entorno basado en plugins, entre los que se encuentra SPARQL Query, mediante el cual se pueden realizar consultas a la ontología. A continuación, se presenta la pregunta de competencia y su respuesta.

Pregunta: ¿Que no conformidades están en estado pendiente?

Respuesta: nc1 y nc10

La figura 11 muestra la consulta y la respuesta que da la herramienta a la pregunta después de realizada la consulta SPARQL.

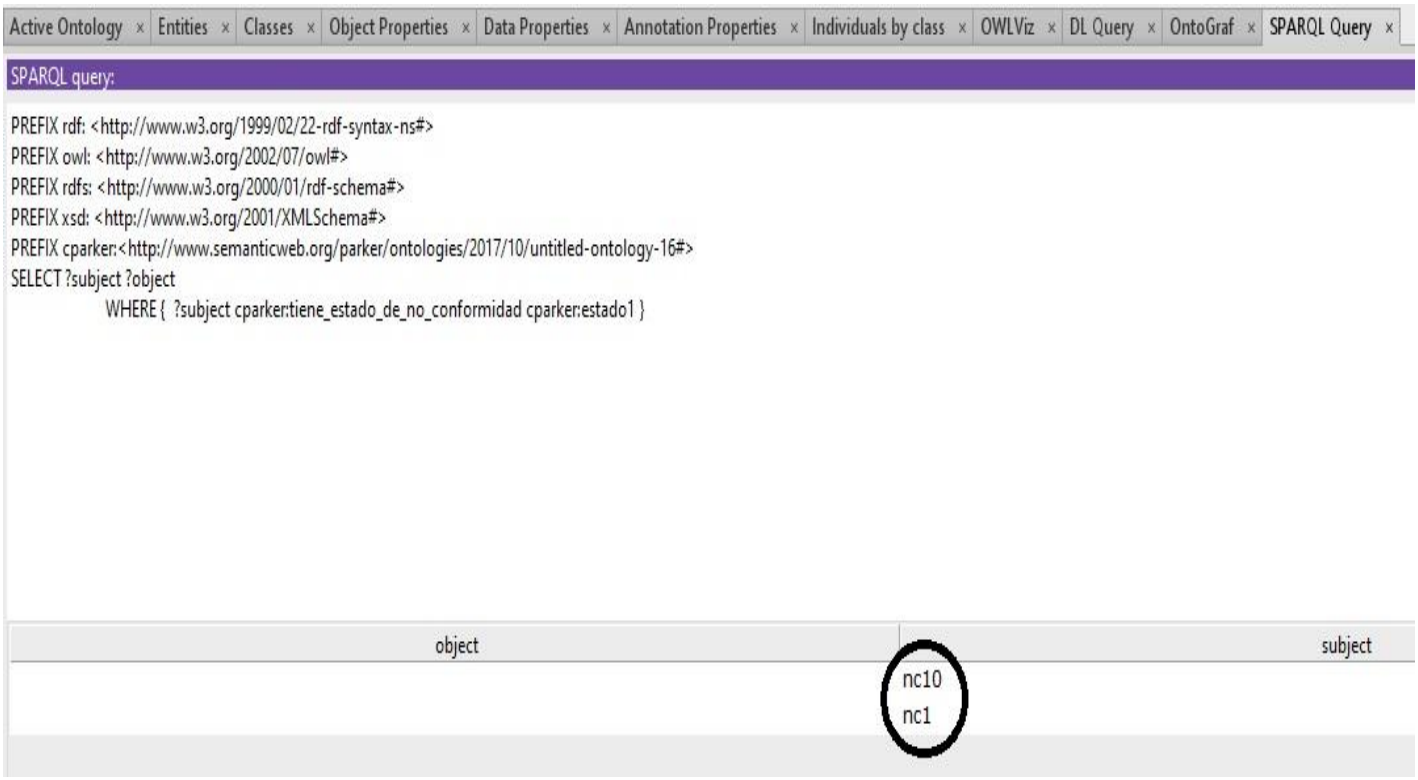


Figura 11. Interfaz de consulta SPARQL

2.2.5 Actividad: Mantenimiento

El mantenimiento de software se define como “cualquier modificación de un producto de software, después de su entrega, para corregir errores, mejorar el rendimiento u otros atributos, o a la acción de adaptar el producto a un entorno que cambia” (Montoya, 2013).

Al llevar a cabo el mantenimiento de la ontología se tendrán en cuenta las modificaciones que se puedan haber originado durante el proceso, además de apreciar los cambios que se puedan producir en la

estrategia de pruebas de la dirección de calidad. De producirse cambios estos podrán incluirse en la ontología.

En la creación de la ontología se han utilizado las anotaciones donde se describen los componentes y las restricciones. Además, la ontología cuenta con una documentación que permite su comprensión, necesaria para poder realizar cualquier cambio sobre ella.

Conclusiones del capítulo

Con el desarrollo de este capítulo, se arribaron a las siguientes conclusiones:

- La correcta descripción del subproceso de pruebas de software permitió determinar los principales conceptos y sus relaciones.
- La utilización de Visual Paradigm como herramienta CASE para el modelado del subproceso de prueba de software. contribuyó a lograr un mejor entendimiento del funcionamiento del mismo.
- La aplicación de la metodología Methontology facilitó la creación de una ontología representativa del conocimiento generado en el subproceso de pruebas de software donde se crearon 57 clases, 178 instancias, 42 data property y 35 object property .

Capítulo 3. Validación de la ontología

En el presente capítulo se realizará la validación de la ontología mediante el esquema para evaluar ontologías propuesto por Ramos (Ramos, 2009), se abordarán algunas de las características principales relacionadas con el contenido del dominio en este caso las pruebas de software y se mostrará el funcionamiento de la ontología creada.

3.1 Validación de la ontología

Ramos (Ramos, 2009) expone que los criterios a evaluar en cada fase del ciclo de vida del desarrollo de una ontología son:

- Uso correcto del lenguaje utilizado para la codificación.
- Exactitud de la estructura taxonómica.
- Significado de los términos y conceptos representados.
- Adecuación a los requerimientos especificados al inicio del desarrollo.

Para cumplir con estos criterios es indispensable que se evalúen los resultados parciales de cada una de las fases del ciclo de vida de desarrollo garantizando de esta forma la calidad de la propuesta ontológica.

Al mismo tiempo manifiesta que, a pesar de que se ha profundizado en el estudio de las metodologías para la creación de ontologías, aún no se ha considerado con suficiente interés el tema de su evaluación. Al mismo tiempo manifiesta que este proceso debe efectuarse de manera similar a la evaluación de un componente de software enfocándose en que las definiciones sean modeladas de la manera más exacta posible al dominio para el cual fueron creadas y que estas den respuesta a las preguntas de competencia.

Por tal motivo propone cuatro fases acordes a los criterios antes descritos con el fin de evaluar ontologías

Fase 1: Uso correcto del lenguaje

- Validar que el lenguaje cumpla con los estándares de desarrollo ontológicos.
- Evaluar sintácticamente la ontología en cada fase de su desarrollo.

Fase 2: Exactitud de la estructura taxonómica

- Chequeo de inconsistencias, completitud y redundancia de los términos de la taxonomía.

Fase 3: Validez del vocabulario

- Identificar, extraer y organizar los términos significativos del dominio a partir de los documentos.
- Evaluar el vocabulario considerando las medidas de calidad de resultados usando medidas tales como la precisión y el *recall (*exhaustividad*)*.

Fase 4: Adecuación a los requerimientos

- Verificar que las especificaciones de requerimientos se cumplan.
- Verificar que las respuestas proporcionadas por la ontología a las preguntas de competencia sean correctas y pertinentes.

A continuación, se describe la forma en la que cada una de estas fases fue aplicada a la ontología.

Fase 1. Uso correcto del lenguaje.

Se utilizó el lenguaje OWL el cual cumple con los estándares para desarrollos ontológicos. Es un lenguaje completo teniendo en cuenta que cualquier expresión que esté lógicamente implícita en la base de conocimiento puede ser derivada, y sólido dado que cualquier expresión puede ser derivada a partir del conocimiento codificado. A raíz de esto se pueden aplicar métodos de razonamiento sobre la ontología de manera satisfactoria. Vale resaltar que en cada fase del ciclo de desarrollo se utilizó el marco de chequeo que provee el editor Protégé-OWL específicamente el razonador pellet, con el objetivo de corregir inconsistencias sintácticas y obtener un código libre de errores.

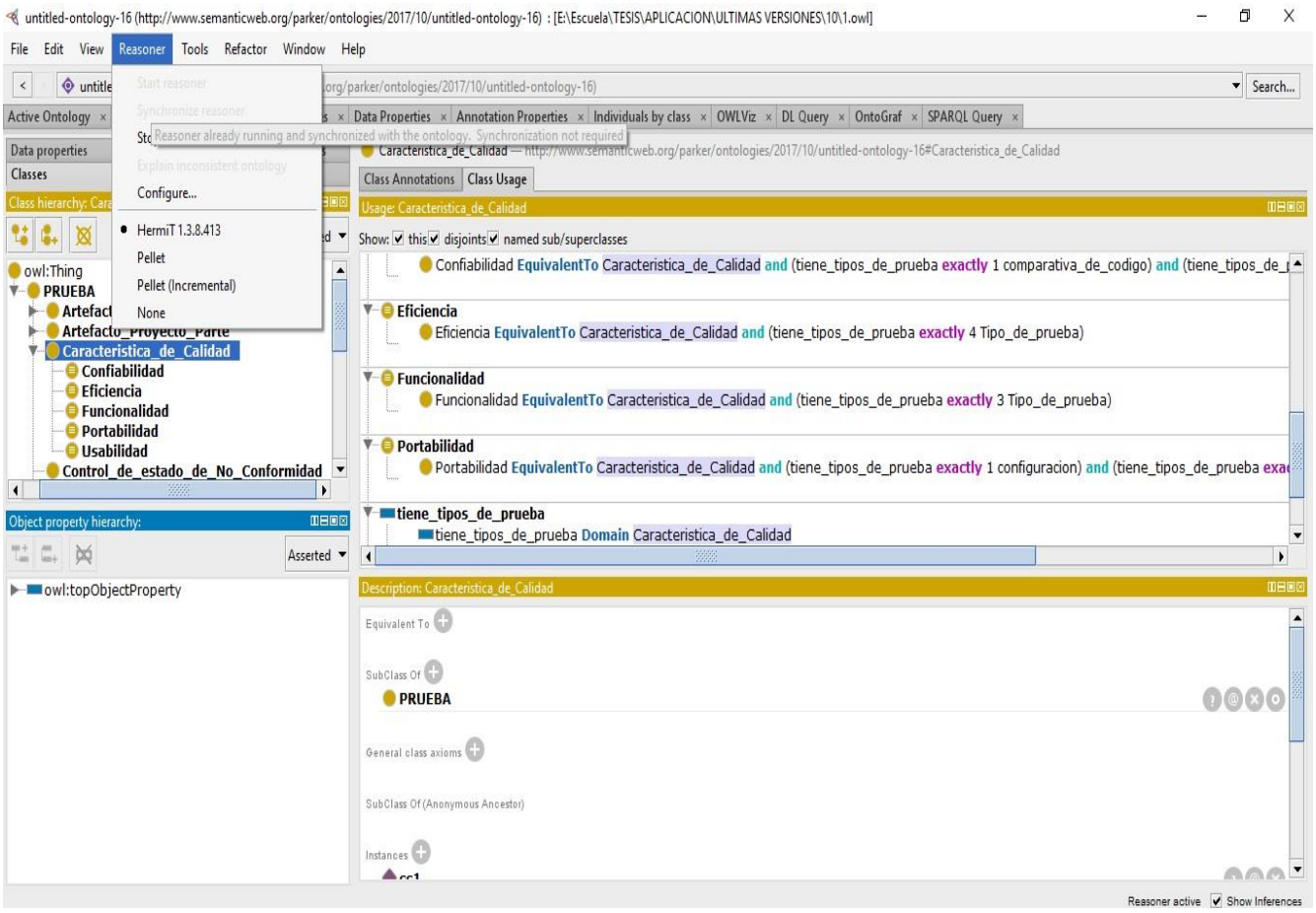


Figura 12.Marco de chequeo del Protégé-OWL.

Fase 2. Exactitud de la estructura taxonómica.

La evaluación taxonómica considera que las actividades que se llevan a cabo para incrementar la calidad de la ontología del modelo multidimensional son: completitud de los conceptos, identificación de inconsistencias y existencia de redundancia en las clases y relaciones, por lo que se realizó una comprobación manual de cada concepto y de las relaciones entre estos. El resultado de aplicar esta fase confirmó que no existen inconsistencias tales como: conceptos que no pertenecen a una clase en particular, ausencia de conceptos relevantes del dominio y clases e instancias con diferentes nombres, pero definiciones similares.

Fase 3. Validez del vocabulario.

En esta fase se chequea que los términos codificados en la ontología existan y sean significativos en otras fuentes de conocimiento independientes. Esto se realiza mediante el cálculo de la precisión y la exhaustividad, medidas numéricas de cubrimiento para determinar la completitud y validez del vocabulario.

El cálculo de la precisión ofrece el porcentaje de los términos que aparecen en el corpus con relación a la cantidad total de términos de la ontología, utilizando la siguiente fórmula:

$$Precision = \frac{CO\_C}{COnto} \quad \text{..... (1)}$$

Donde:

CO_C: es la cantidad de términos que se solapan entre la ontología y el corpus,

COnto: es la cantidad total de términos que tiene la ontología.

$$Precision = \frac{57}{57} = 1$$

Esto nos demuestra que el 100% de los términos existentes en la ontología se encuentran en el corpus del dominio.

Por su parte el cálculo de la exhaustividad proporciona el porcentaje de términos del corpus que aparecen en la ontología con relación al total de términos en el corpus, utilizando la siguiente expresión:

$$Exhaustividad = \frac{CO\_C}{CCorp} \quad \text{..... (2)}$$

Donde:

CCorp: representa la cantidad total de términos del corpus.

$$Exhaustividad = \frac{57}{57} = 1$$

Esto nos demuestra que el 100% de los términos del corpus del dominio aparecen en la ontología.

Fase 4. Adecuación a los requerimientos.

En esta fase se debe cumplir con la metodología y con las preguntas de competencia. Se cumplieron todos los pasos planteados en la metodología Methontology.

A continuación, se muestran los pasos a seguir para conocer la respuesta de la herramienta a una pregunta de competencia. (Ver figura 13)

1. Dar clic en la pestaña individuals by class

2. Seleccionar el individuo que se relaciona directamente en la pregunta, en este caso específico seria cc4 que es la característica de calidad de la que se quiere conocer la clasificación acorde a los tipos de prueba que corresponden a ella.
3. Activar el razonador.
4. comprobar la respuesta.

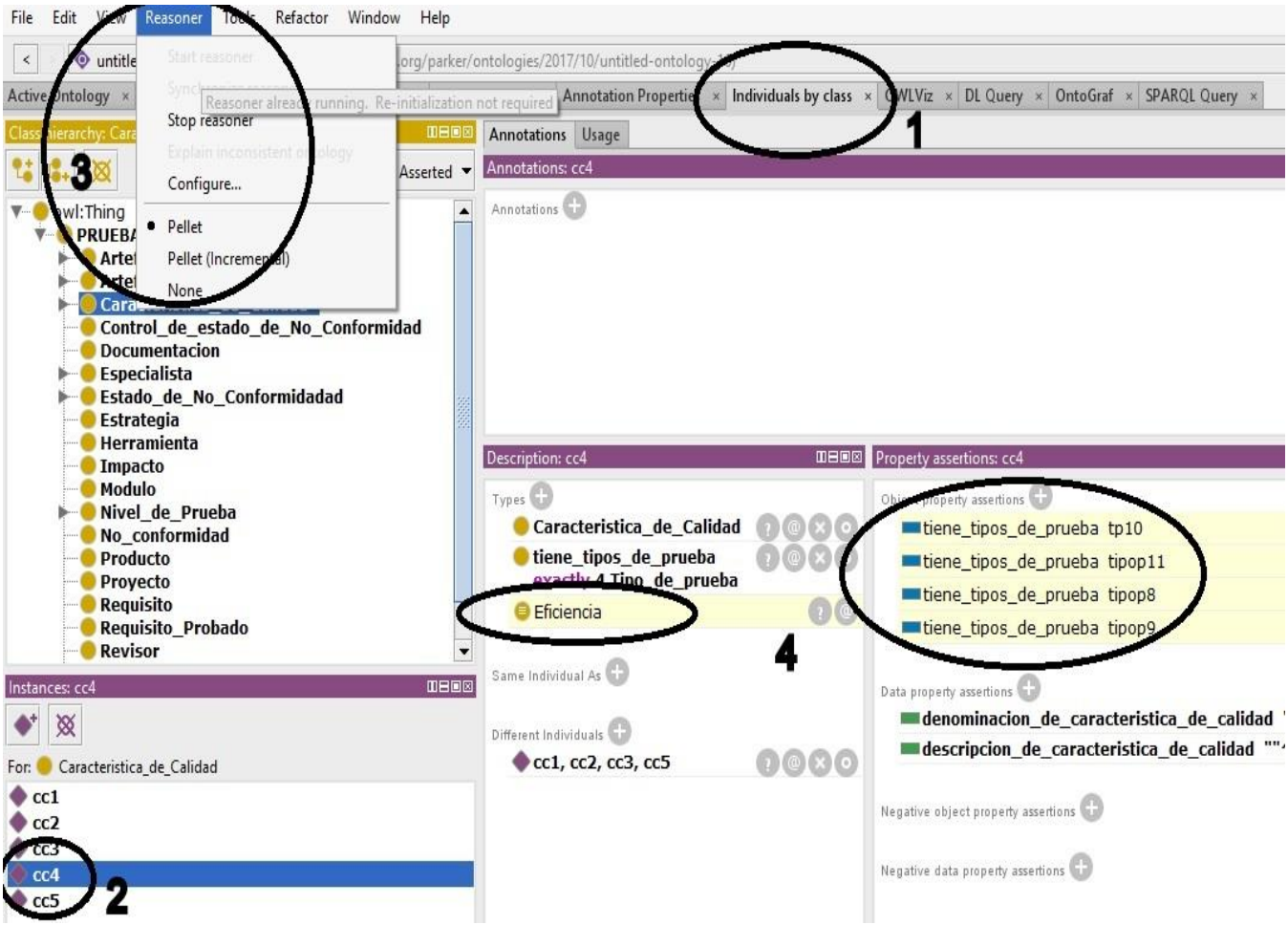


Figura 13 Pasos a seguir para conocer la respuesta a una determinada pregunta de competencia.

Los pasos antes descritos se realizan de la misma forma para todas las preguntas con la condición de que se seleccione en el paso 2 el individuo al que se hace referencia en la pregunta.

A continuación, se muestra la respuesta de la herramienta Protégé a algunas de las preguntas de competencia (Ver anexo 1 para más preguntas)

1 ¿Qué herramientas manipula el revisor a la hora de realizar la detección de las no conformidades?

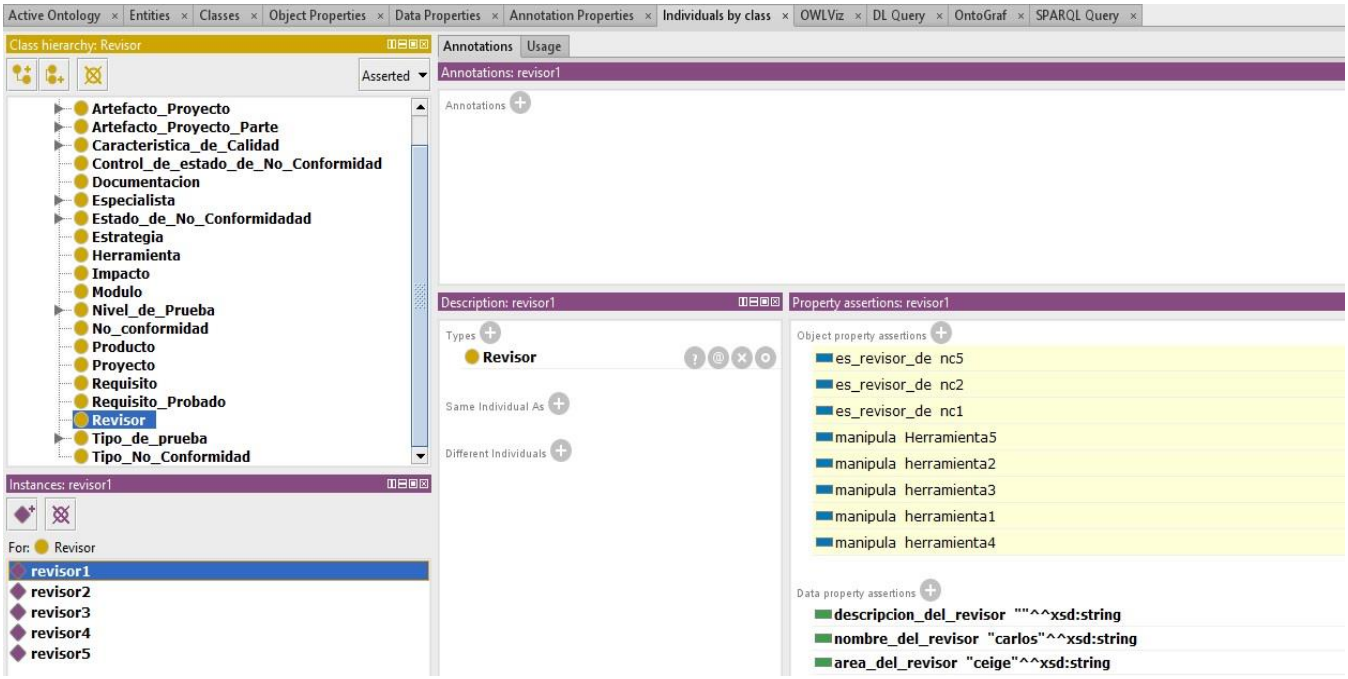


Figura 14. Interfaz 1. Respuestas

2 ¿Qué tipos de prueba corresponden a una determinada característica de calidad?

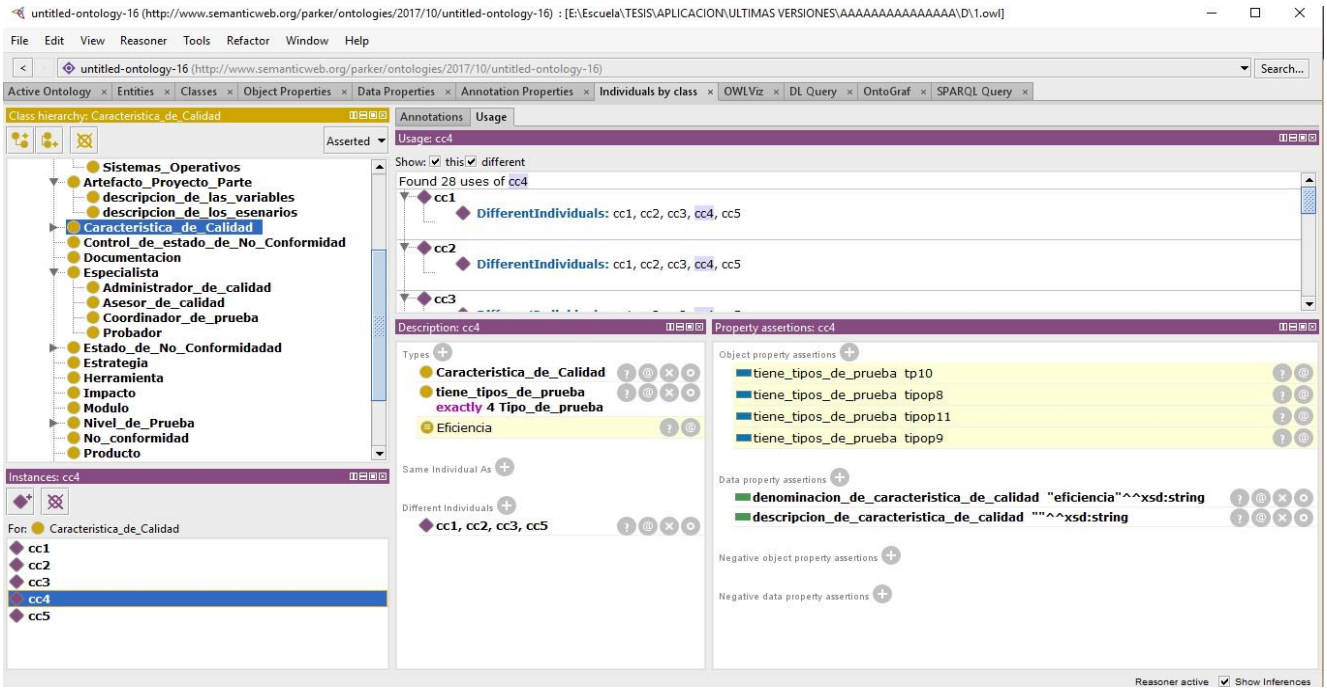


Figura 15.Interfaz 2. Respuestas

3 ¿De qué tipo es una determinada no conformidad?

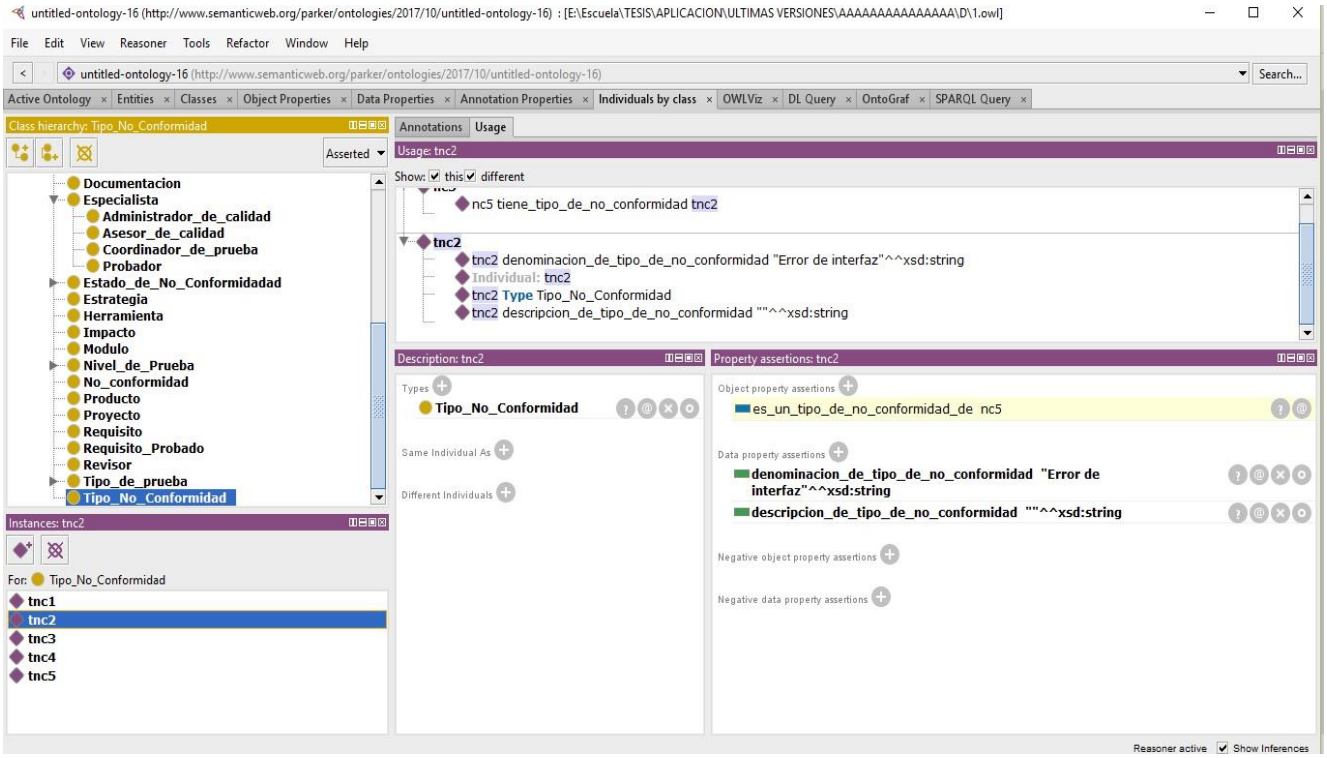


Figura 16.Interfaz 3 Respuestas

4 ¿Qué no conformidades presento un determinado artefacto de proyecto y cuál es su clasificación?

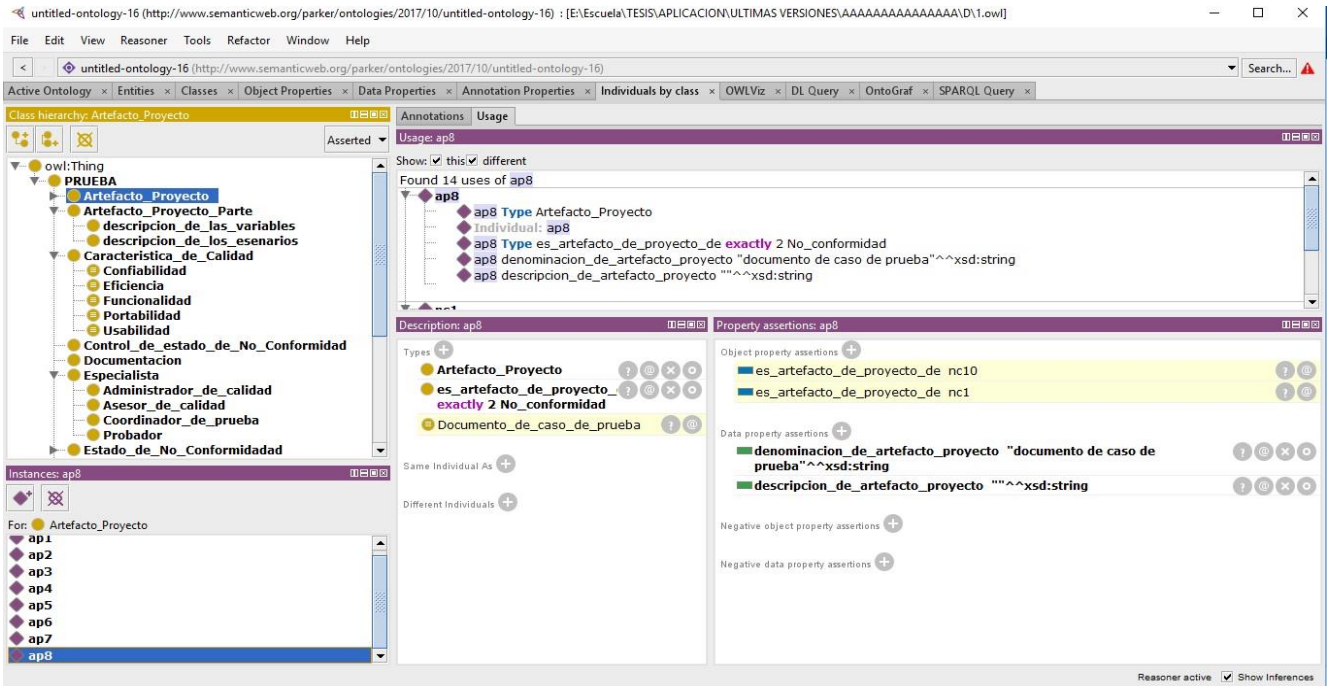


Figura 17Interfaz 4. Respuestas

5 ¿Qué impacto tiene una determinada no conformidad?

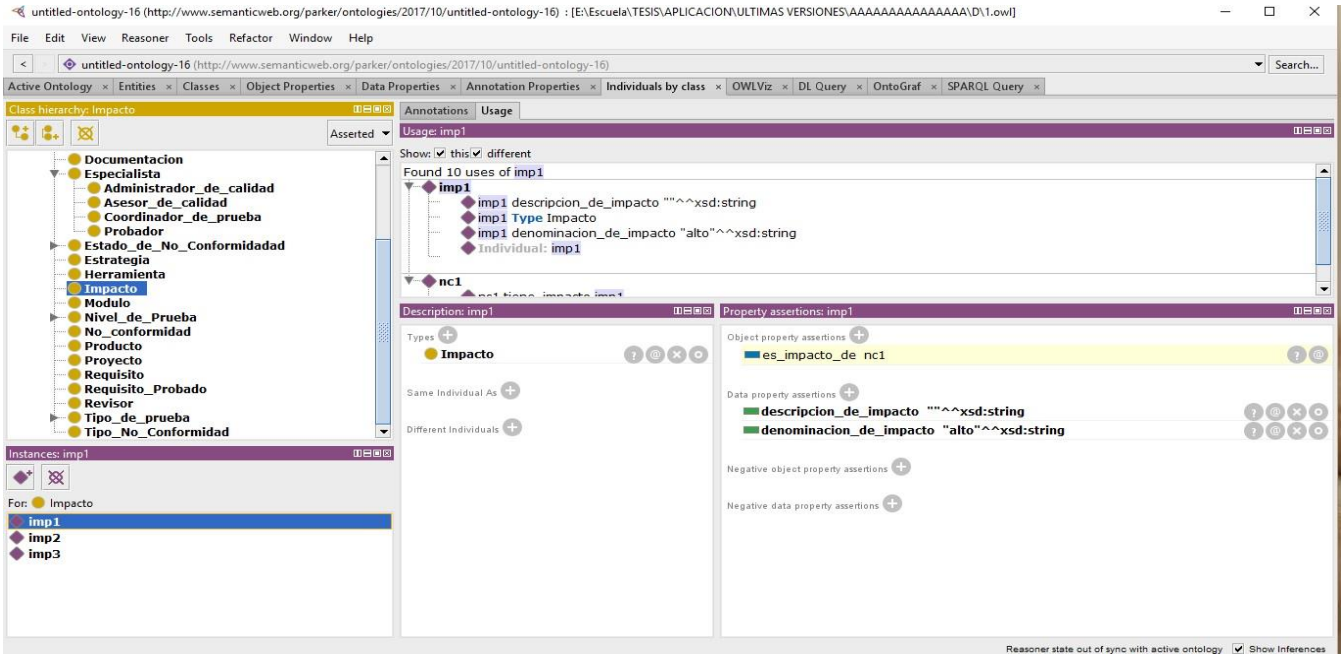


Figura 18.Interfaz 5. Respuestas

6 ¿Que módulos tiene un determinado producto?

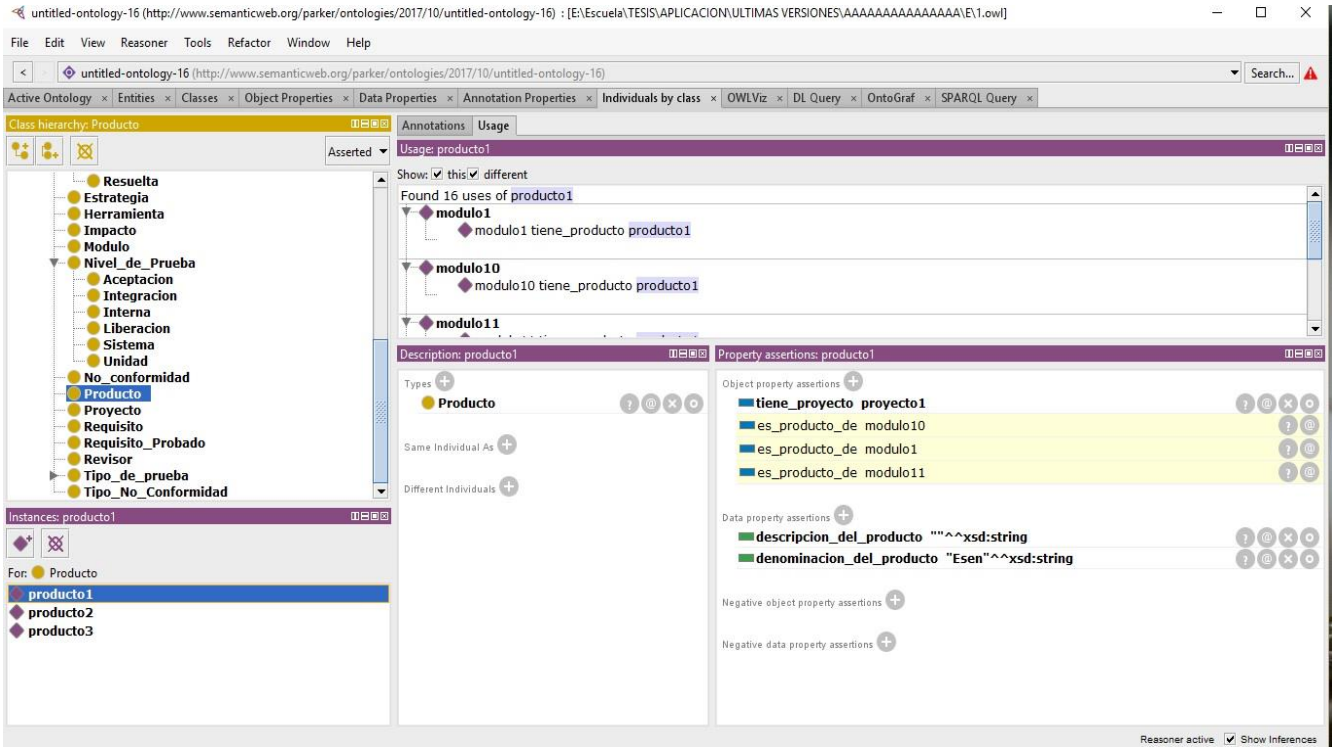


Figura 19.Interfaz 6. Respuestas

7 ¿Que módulos pertenecen a un determinado requisito?

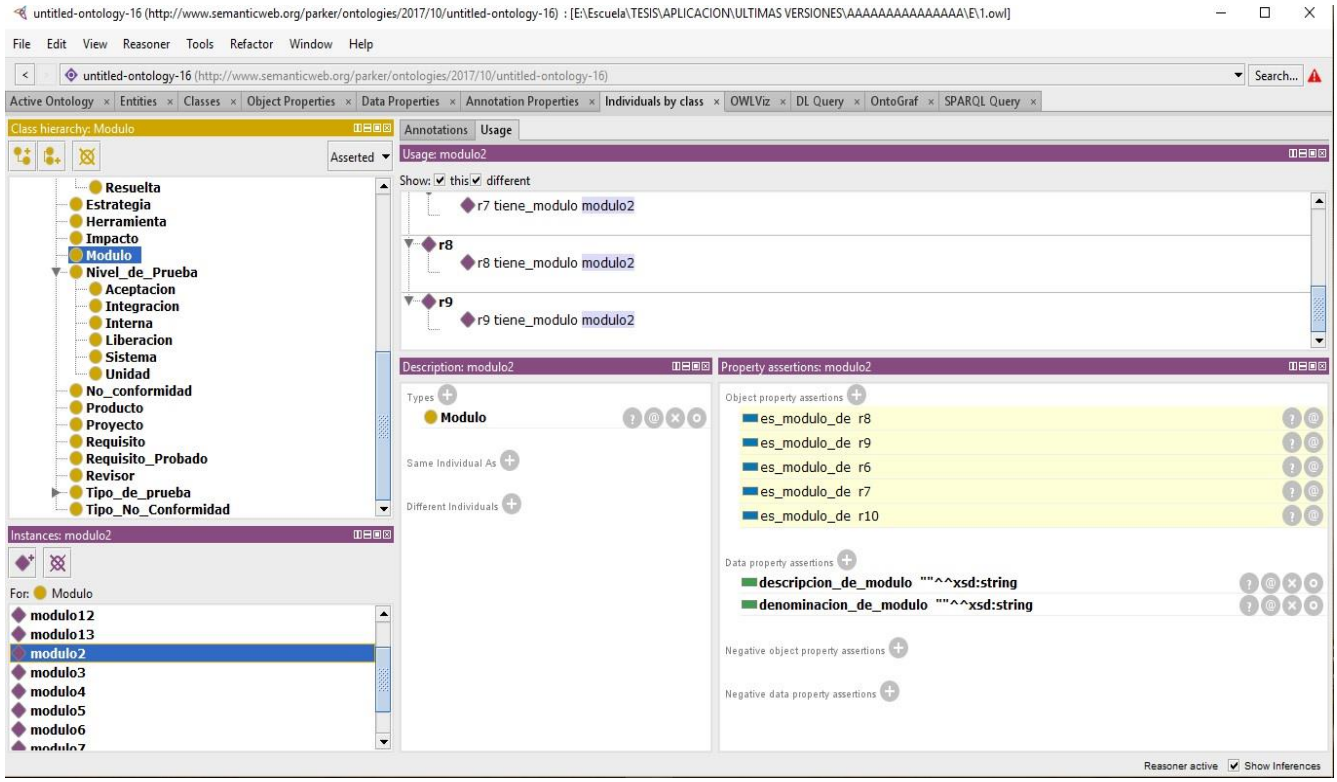


Figura 20.Interfaz 7. Respuestas

8 ¿Qué no conformidad detectó un determinado revisor?

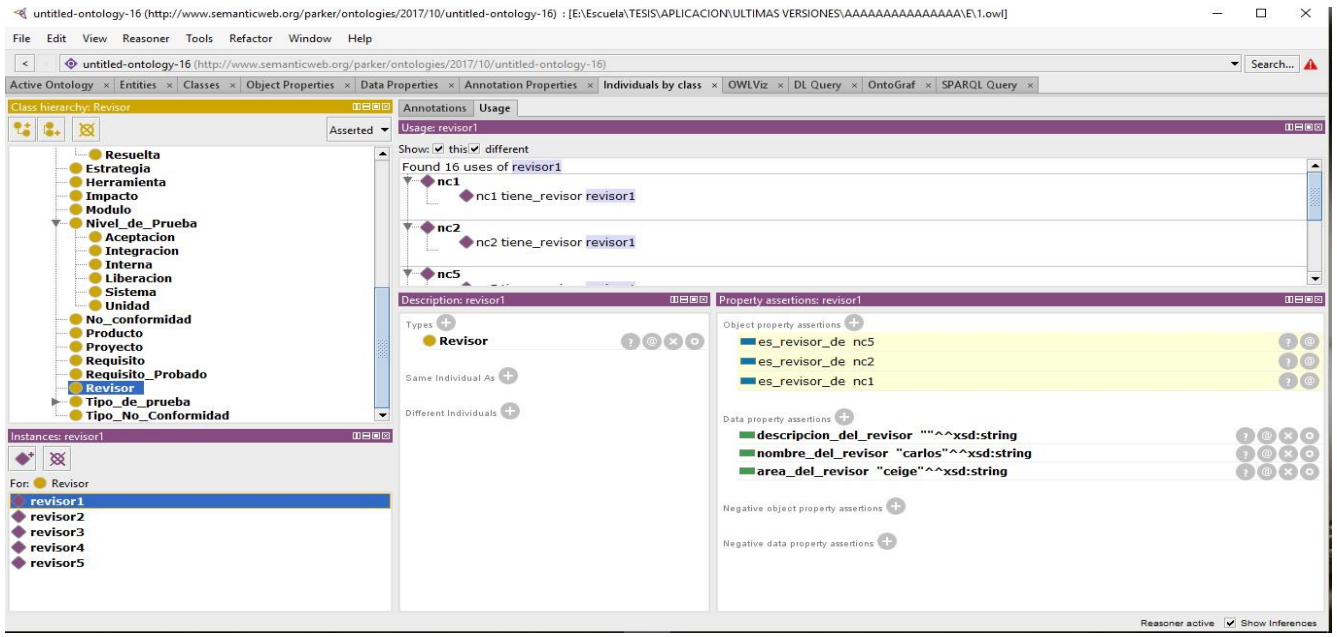


Figura 21.Interfaz 8. Respuestas

9 ¿Cuál es el estado de una no conformidad?

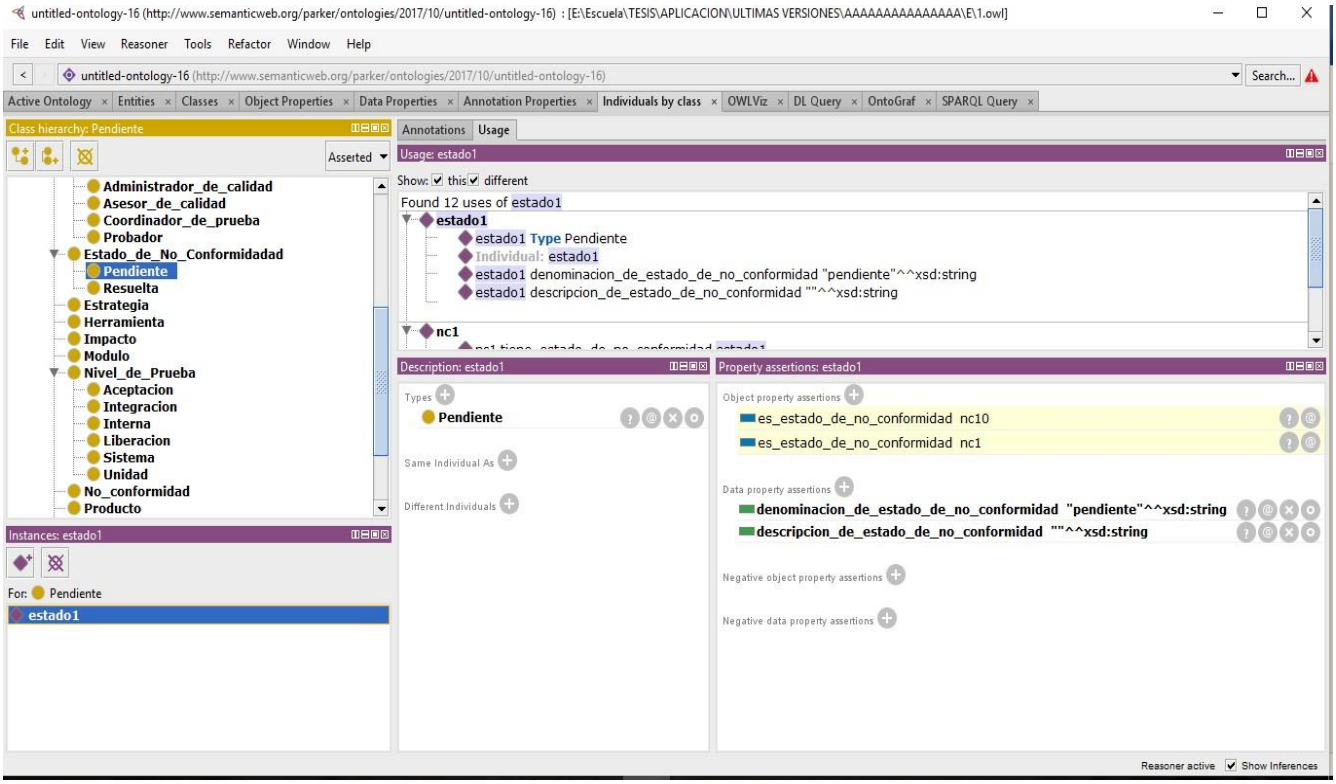


Figura 22.Interfaz 9. Respuestas

10 ¿Qué documentación requiere un determinado artefacto de proyecto para ser analizado?

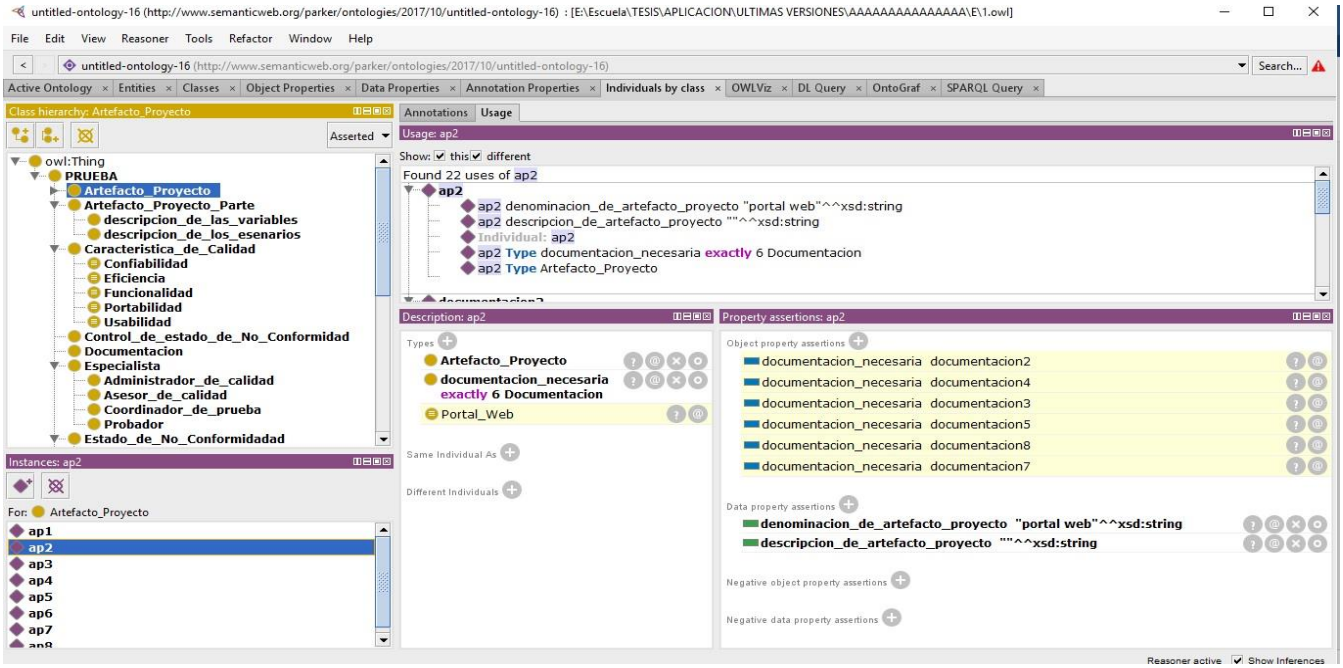


Figura 23.Interfaz 10. Respuestas

11 ¿A qué proyecto pertenece un determinado especialista?

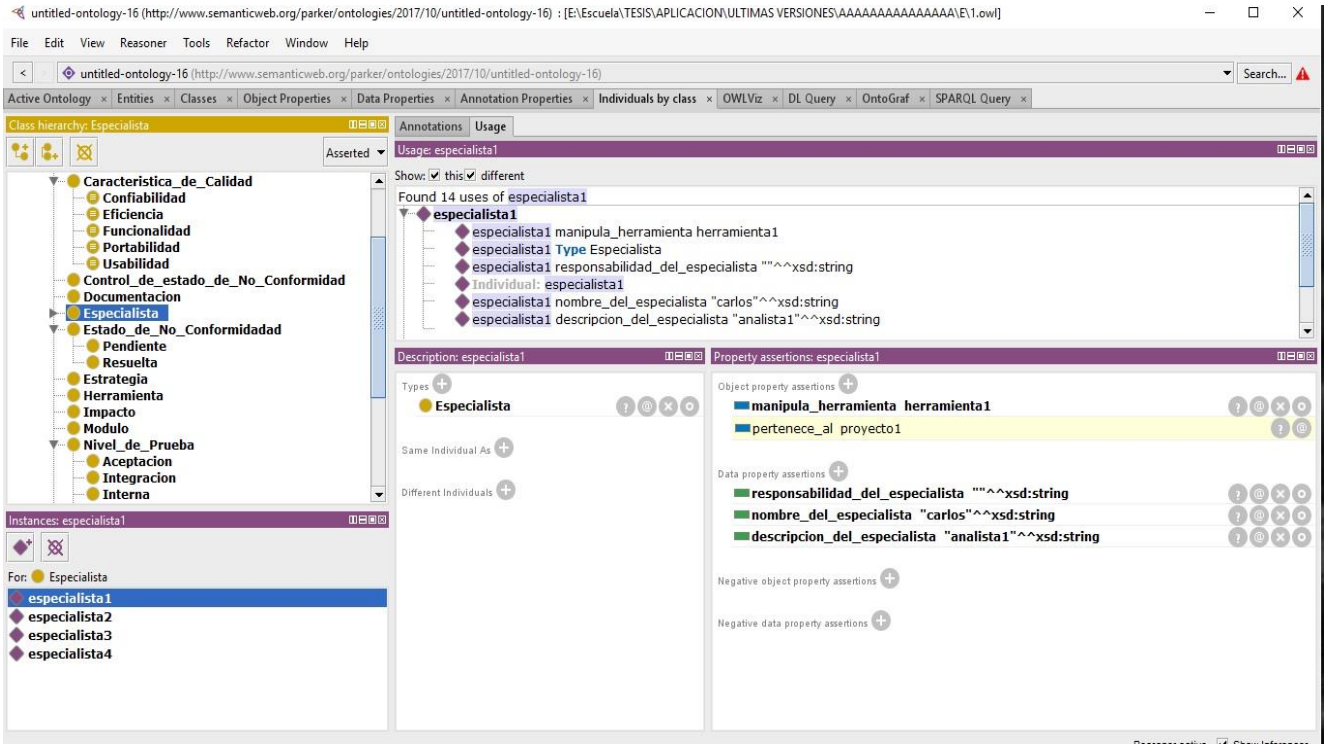


Figura 24.Interfaz 11. Respuestas

Durante el desarrollo de la investigación, el profesor que se encuentra al frente de la asignatura PID 2 en la UCI se mantuvo chequeando la realización de la misma hasta su culminación. Avalando de esta forma la aplicación de la propuesta ontológica en el dominio que le refiere.

Conclusiones del capitulo

Una vez realizada la validación de la solución al problema planteado se concluye lo siguiente:

- La validación mediante el “Esquema para evaluar ontologías únicas para un dominio de conocimiento”, quedó demostrado el uso correcto del lenguaje y que la estructura taxonómica no contiene inconsistencias ni redundancias.
- Las preguntas de competencia definidas y aplicadas permitieron determinar el alcance de la ontología.
- Una vez validada la propuesta ontológica queda evidenciada que la misma sirve de apoyo al subproceso de prueba en la dirección de calidad, permitiendo la organización y comunicación del conocimiento generado en dicho subproceso.

Conclusiones Generales

Los resultados obtenidos durante el desarrollo del presente trabajo permiten llegar a las siguientes conclusiones:

- El subproceso de pruebas de software juega un papel fundamental en las empresas que desarrollan software. Este garantiza que el producto final tenga la calidad requerida.
- Se demostró la necesidad de desarrollar una ontología que apoye el subproceso de pruebas de software en la Dirección de Calidad de la UCI, pues en el análisis del estado del arte se determinó que las ontologías identificadas son generales e incompletas al no abordar todos los términos y sus relaciones.
- Las herramientas, metodología y tecnologías seleccionadas contribuyeron al desarrollo de una propuesta ontológica que encierra el conocimiento que está presente en el dominio de las pruebas de software, permitiendo de esta forma la organización y comunicación del conocimiento generado.
- La ontología propuesta garantiza la organización y comunicación del conocimiento generado, apoyando el subproceso de pruebas de software en la Dirección de Calidad de la UCI. Esto se demostró a través del esquema de validación realizado, que evalúa la elaboración correcta de la ontología, el uso correcto del lenguaje utilizado para la codificación, la no existencia de inconsistencias y la capacidad de inferencia.

Recomendaciones

- Incluir en la modelación del conocimiento las pruebas desde un enfoque estático dígase: revisiones técnicas formales, revisiones a la documentación y auditorías.
- Poblar la ontología a partir de los datos existentes en la herramienta Gespro mediante un proceso automatizado.

Referencias Bibliográfica

1. Barité, Mario. 2014. *El control de vocabulario en la era digital.* [En línea] 2014 [Citado el: 20 de marzo del 2018]. Disponible en: http://www.ibersid.eu/ojs/index.php/scire/article/view/4196

2. Canals, Agustí. 2003. *La gestión del conocimiento.* [En línea] 2003 [Citado el: 10 de noviembre del 2017]. Disponible en: http://www.uoc.edu/dt/20251/

3. Carrasco, Selin. 2010. *Ontologías y su utilidad en ingeniería de software.* [En línea] 7 de diciembre de 2010 [Citado el: 10 de enero del 2018]. Disponible en: https://es.slideshare.net/selincarrasco/ontologas-y-su-utilidad-en-ingeniera-de-software

4. Casamayor, Carlos Carrascosa. 2005. Herramienta ontoedit. *Herramienta de creacíon de ontologías ontoedit.* [En línea] 14 de febrero de 2005 [Citado el: 29 de enero del 2018].. Disponible en: http://personales.upv.es/ccarrasc/doc/2003-2004/ontoedit/presentacion_htm.

5. Centelles, Miquel. 2005. *Taxonomías para la categorización y la organización de la información en sitios web.* Barcelona : s.n., [En línea] 2005 [Citado el: 16 de febrero del 2018]. Disponible en:

6. Deborah L. Mcguinness, Frank van Harmelen. 2004. W3C. *W3C.* [En línea] 10 de FEBRERO de 2004 [Citado el: 23 de febrero del 2018]. Disponible en: https://www.w3.org/TR/2004/REC-owl-features-20040210/.

7. Esmeralda Ramos, Haydemar Núñez, Roberto Casañas. 2011. *Diseño de un sistema basado en agentes para apoyar el diagnóstico de la calidad del semen humano.* [En línea] octubre de 2011. Vol. 26 [Citado el: 10 de marzo del 2018]. Disponible en: http://revistadelafacultaddeingenieria.com/index.php/ingenieria/article/view/279

8. Felipe, Rohandy Alfonso. 2011. *Implementación de una ontología para el proceso de ingeniería de requisitos del modelo cubano de calidad para el desarrollo de aplicaciones informáticas en la industria cubana de software.* UCI. La habana : s.n., [En línea] 2011. Trabajo de diploma [Citado el: 9 de diciembre del 2017]. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04196_11/1/TD_04196_11.pdf

9. Figueroa, Liliana y Palavecino, Rosa. 2006. Aproximación a la diferencia entre Gestión de la Información y la Gestión del Conocimiento. [En línea] 2006 [Citado el: 13 de enero del 2018]. Disponible en: Http://www.cibersociedad.net/congres2006/gts/comunicacio.php?Id=618&llengua=es.

10. Gruber, T. R. 1993. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing". Stanford Knowledge Systems Laboratory.* [En línea] 1993 [Citado el: 10 de diciembre del 2017]. Disponible en: http://archive.itee.uq.edu.au/~infs3101/_Readings/OntoEng.pdf

11. Guarino, Nicola. 1995. *Formal Ontology, Conceptual Analysis and Kwnoledge Representation.* [En línea] 1995. Págs. 625-640. Vol. 43. 5-6 [Citado el: 11 de febrero del 2018]. Disponible en: https://www.sciencedirect.com/science/article/pii/S107158198571066X

12. Hendler, Berners-Lee y Miller. 2012. Diseño de Páginas Web, Sitios de Internet y Posicionamiento SEO akus.net. *Web Semántica, definición, historia y características.* [En línea] 2012 [Citado el: 5 de enero del 2018]. Disponible en: Https://disenowebakus.net/semantica-web.php.

13. Hernández, Anisleiby Fernández. 2015. *Modelo Ontológico de recuperación de información para la toma de decisiones en Gestión de Proyectos.* Granada : s.n., [En línea] 2015. [Citado el: 3 de enero del 2018] Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/9046/2/Anisleiby%20Fern%c3%a1ndez%20Hern%c3%a1ndez.pdf

14. IEEE. 1991. Nueva York : s.n., [En línea] 1991 [Citado el: 20 de diciembre del 2017]

15. Katia Cristina Duarte, Ricardo de Almeida Falbo. 2000. *Uma Ontologia de Qualidade de Software.* [En línea] 2000 [Citado el: 2 de mayo del 2018]. Disponible en: https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/uma_ontologia_de_qualidade_de_software_2000.pdf

16. Krsarmiento. 2011. Semantizando la Web. *Semantizando la Web.* [En línea] 7 de noviembre de 2011 [Citado el: 2 de diciembre del 2017]. Disponible en: Https://semantizandolaweb.wordpress.com/2011/11/07/que-es-rdf-y-para-que-es-bueno/.

17. Lamas, Maria Isabel. 2006. *Lenguajes de consulta para documentos RDF.* [En línea] 9 de enero de 2006. [Citado el: 13 de diciembre del 2017] Disponible en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/640/1/38449tfc.pdf

18. Lapuente, María Jesús Lamarca. 2013. *Hipertexto, el nuevo concepto de documento en la cultura de la imagen.* Madrid : s.n., [En línea] 2013 [Citado el: 15 de diciembre del 2018]. Disponible en: http://www.hipertexto.info/documentos/ontologias.htm

19. Laura grela, elena sauri, alicia sellés. 2002. *Ontologías en documentación.* [En línea] 2002 [Citado el: 15 de febrero del 2018]. Disponible en: http://personales.upv.es/ccarrasc/doc/2001-2002/ontologias/INICIO.htm

20. Leonard L. Tripp, Chair. 2014. *Swebok.* [En línea] 2014 . Disponible en: http://www.cc.uah.es/drg/b/HispaSWEBOK.Borrador.pdf

21. Luna, Jaime Alberto Guzmán. 2012. *Metodologías y métodos para la construcción de ontologías.* [En línea] abril de 2012. 50. Disponible en: http://www.redalyc.org/comocitar.oa?id=84923878033

22. María Aurora Soto Balbón, Norma M. Barrios Fernández. 2006. Gestión del conocimento. Parte I. Revisión crítica del estado del arte. *Gestión del conocimento.* [En línea] 30 de marzo de 2006. [Citado el: 5 de diciembre del 2017] Disponible en: Http://bvs.sld.cu/revistas/aci/vol14_2_06/aci04206.htm.

23. Mateus Ferreira, Félix García, Francisco Ruiz, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Mario Piatti, Beatriz Mora. 2006. *Medición del Software Ontología y Metamodelo.*

[En línea] 2006 [Citado el: 2 de mayo del 2018]. Disponible en: https://previa.uclm.es/dep/tsi/pdf/UCLM-TSI-001.pdf

24. Montoya, Edgar Serna. 2013. *Acercamiento ontológico a la gestión del conocimiento en el mantenimiento del software.* [En línea] 2013 [Citado el: 10 de enero del 2018]. Revista Facultad de Ingeniería Universidad de Antioquia. Disponible en: http://www.redalyc.org/articulo.oa?id=43019328019

25. Perez, Delvis Echeverría. 2011. *Desarrollo de una ontología de apoyo al procedimiento del Departamento de Pruebas de Software de Calisoft.* La habana : s.n., [En línea] diciembre de 2011. [Citado el: 18 de diciembre del 2017]

26. Pressman, R.S, Mc Graw Hill. 2009. *Ingeniería del Software. Un enfoque práctico.* [En línea] 2009. Vol. 6. [Citado el: 30 de octubre del 2017]

27. Pressman, Roger S. 2010. *Software Engineering.* S.l. : Higher Education, [En línea] 2010. 978-0-07-337597-7 [Citado el: 3 de febrero del 2018].

28. Ramos, Esmeralda. 2009. *Esquema para evaluar ontologías únicas para un dominio de conocimiento.* [En línea] 2009 [Citado el: 25 de abril del 2018]. Disponible en: https://doaj.org/article/2e52e18a85de4c94b7009f89c2d2c697

29. Research, Stanford. 2016. Protege.stanford.edu. *Protege.stanford.edu.* [En línea] 2016. [Citado el: 20 de noviembre del 2017]. Disponible en: Https://protege.stanford.edu/.

30. Samper Zapater, José Javier. 2005. *Ontologías para Servicios Web Semánticos de Información de Tráfico: Descripción y Herramientas de Explotación.* [En línea] 2005 [Citado el: 15 de diciembre del 2017].. Disponible en: http://www.tesisenxarxa.net/TESIS_UV/AVAILABLE/TDX-0628106-085805//SAMPER.pdf

31. Sánchez, Javier Zapata. 2013. Pruebas de software:Ingenieria de Software con énfasis en pruebas. [En línea] 21 de enero de 2013 [Citado el: 3 de febrero del 2018].. Disponible en: Https://pruebasdelsoftware.wordpress.com/.

32. Significados.com. 2017. Taxonomía. *Significados.com.* [En línea] 7 de noviembre de 2017. [Citado el: 19 de mayo del 2018].Disponible en: Https://www.significados.com/taxonomia/

33. Simón, Alfredo, y otros. 2006. *GECOSOFT: UN Entorno Colaborativo para la Gestión del Conocimiento con Mapas Conceptuales.* [En línea] 2006 [Citado el: 27 de diciembre del 2017].. Disponible en: http://cmc.ihmc.us/cmc2006Papers/cmc2006-p156.pdf

34. Souza, E. F. 2013. *Using Ontology Patterns for Building a Reference Sofware Testing Ontology.* [En línea] 2013 [Citado el: 5 de febrero del 2018].. Disponible en: http://mtc-m21b.sid.inpe.br/col/sid.inpe.br/mtc-m21b/2014/01.21.15.32/doc/Using%20Ontology%20Patterns%20for%20Building%20a%20Reference%20Software%202013.pdf?metadatarepository=&mirror=iconet.com.br/banon/2006/11.26.21.31

35. Torcoroma Velásquez Pérez, Mauricio Andrés Puentes Velásquez, Alberto Jaime Guzmán Luna. 2011, [En línea] 8 de julio de 2011 [Citado el: 25 de abril del 2018]., Revista Avances en Sistemas e Informática, Vol. 8. Disponible en: http://www.redalyc.org/articulo.oa?id=13311986702

36. Torre, Pablo de la. 2010. *Almacenes de Datos para la Web Sem¶antica.* Cadiz (españa) : s.n., [En línea] 2010 [Citado el: 5 de enero del 2018].. Disponible en: http://www.lsi.us.es/docs/doctorado/memorias/Torre-Moreno-Pablo-Memoria-Investigacion.pdf

37. University, Stanford. 2016. Ksl.stanford.edu. *Ksl.stanford.edu.* [En línea] 2016 [Citado el: 20 de noviembre del 2017].. Disponible en: Http://www.ksl.stanford.edu/software/ontolingua/ontology-server-projects.html.

38. Valencia, Rafael. 2010. La siguiente versión de la web, la web semántica. *La siguiente versión de la web, la web semántica.* [En línea] 2010 [Citado el: 24 de octubre del 2017].. Disponible en: Http://www.ciimurcia.es/informas/ene05/articulos/La_siguiente_version_de_la_web_la_web_semantica.pdf.

39. Vasco, Sociedad informática del Gobierno. 2009. *Ejie.eus.* [En línea] 2009 [Citado el: 20 de diciembre del 2017].. Disponible en: http://www.ejie.eus/contenidos/informacion/anexos_pbt_ejie/eu_0214/adjuntos/Estandares%20de%20calidad%20de%20producto%20software%20v1.4.pdf.

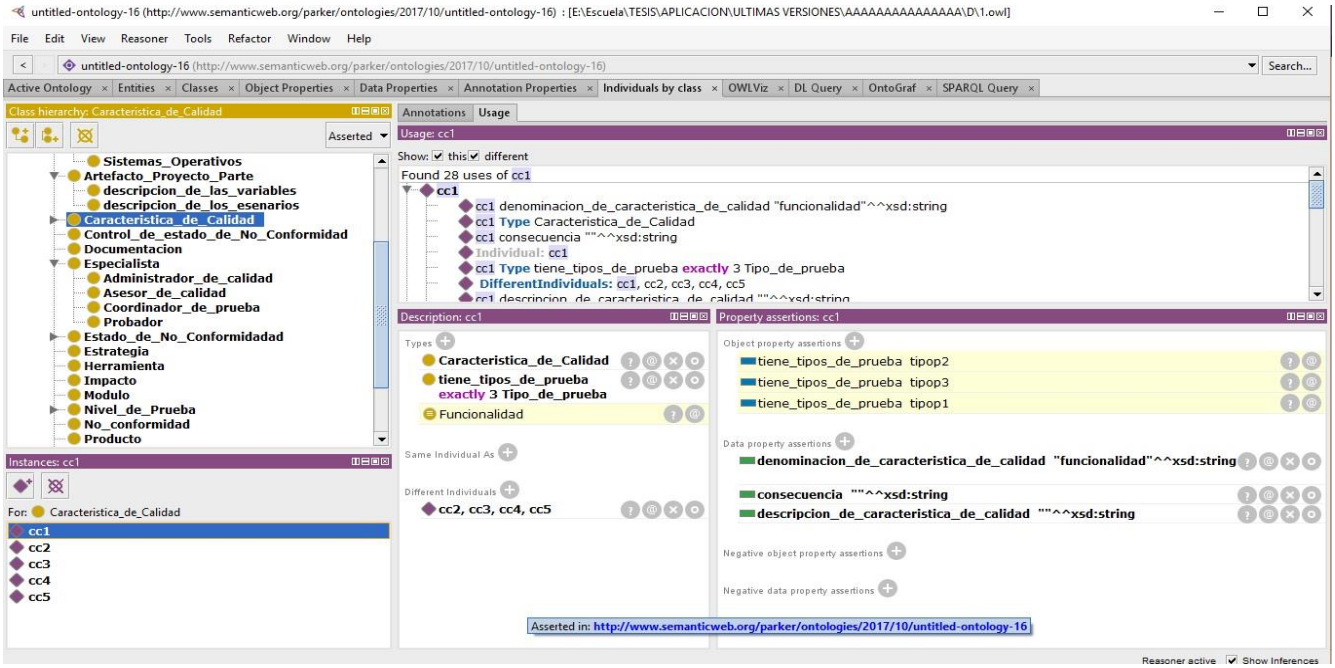
40. Vitelli, Iván Flores. 2011. *Aplicación de Methontology para la Construcción de una Ontología en el Domino de la Microbiología.* Caracas, Venezuela : s.n., [En línea] 2011 [Citado el: 5 de enero del 2018]..

41. Weigand, Hans. 1997. *A Multilingual Ontology-based Lexicon for News Filtering.* [En línea] 1997 [Citado el: 7 de marzo del 2018].. Disponible en: https://pure.uvt.nl/portal/files/234417/ijcai3.pdf

Anexos

Anexo 1: Preguntas de competencia

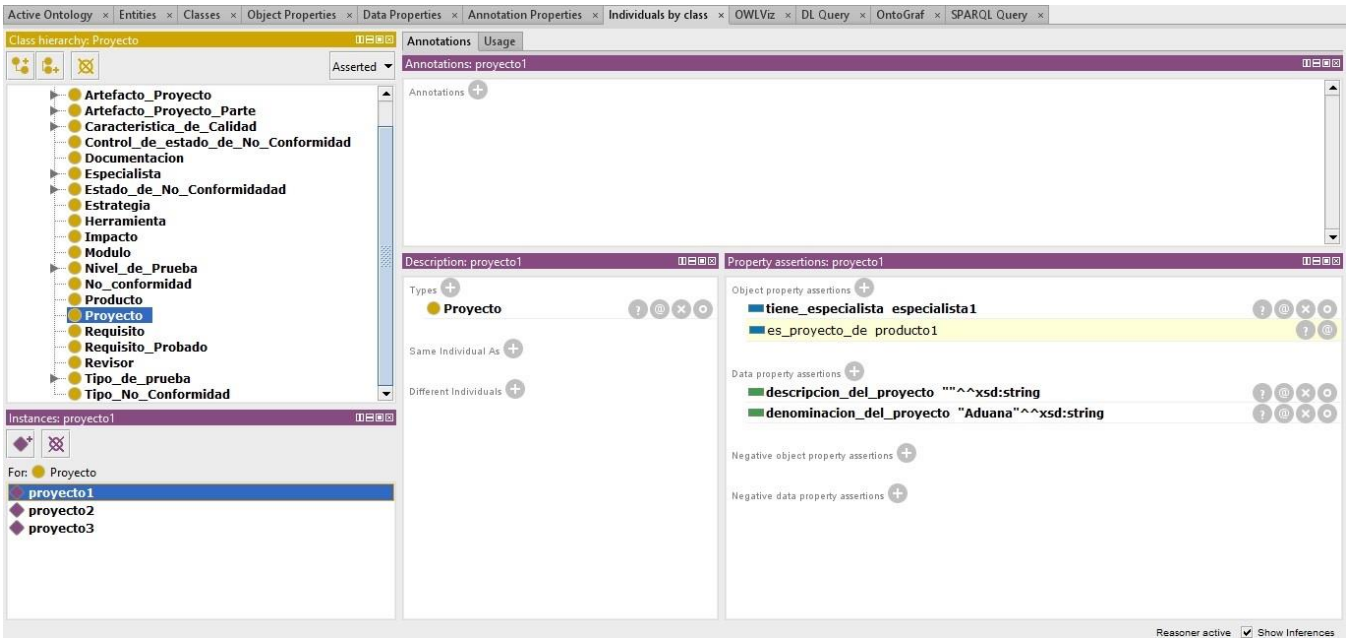
12. ¿Cuál es la clasificación de una determinada característica de calidad acorde a los tipos de prueba?

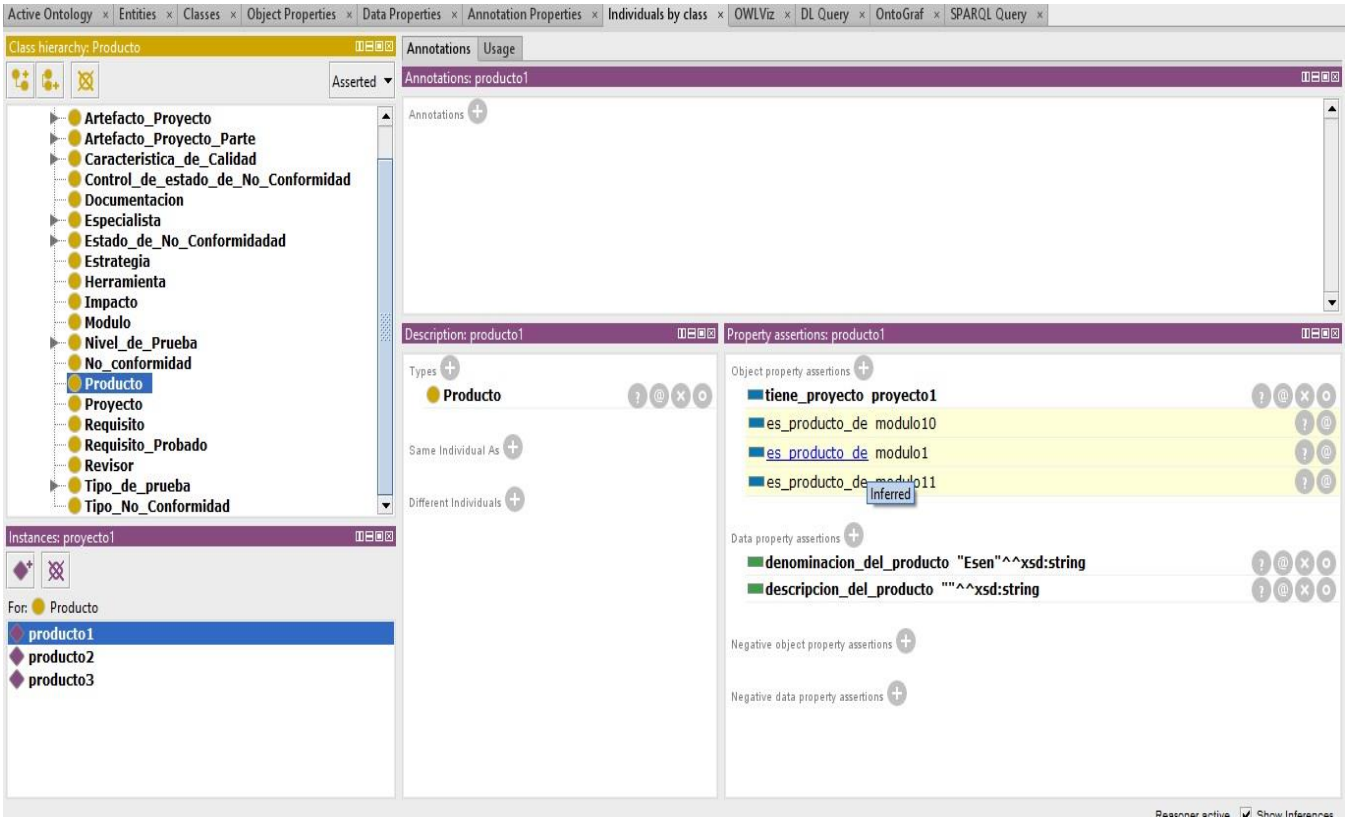


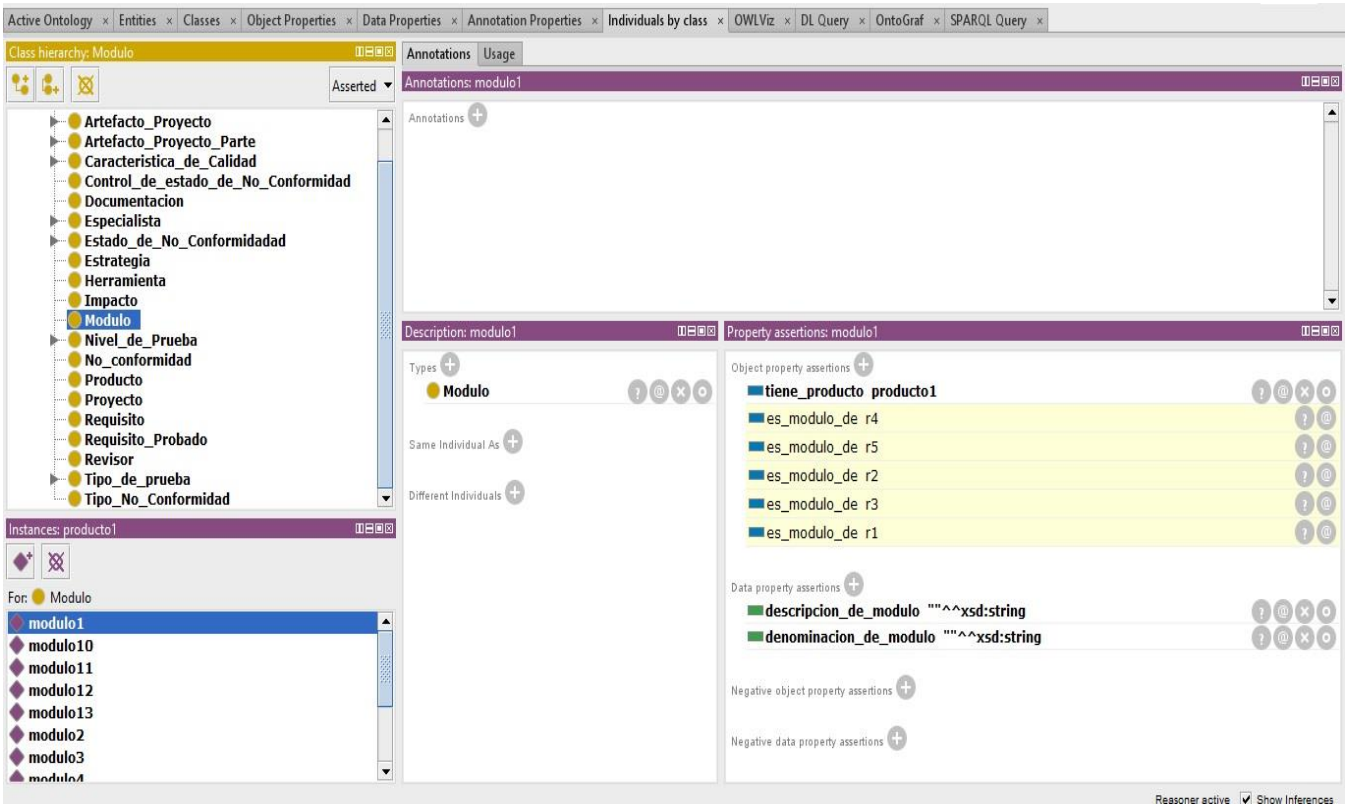
13. ¿Qué no conformidades presento un proyecto?

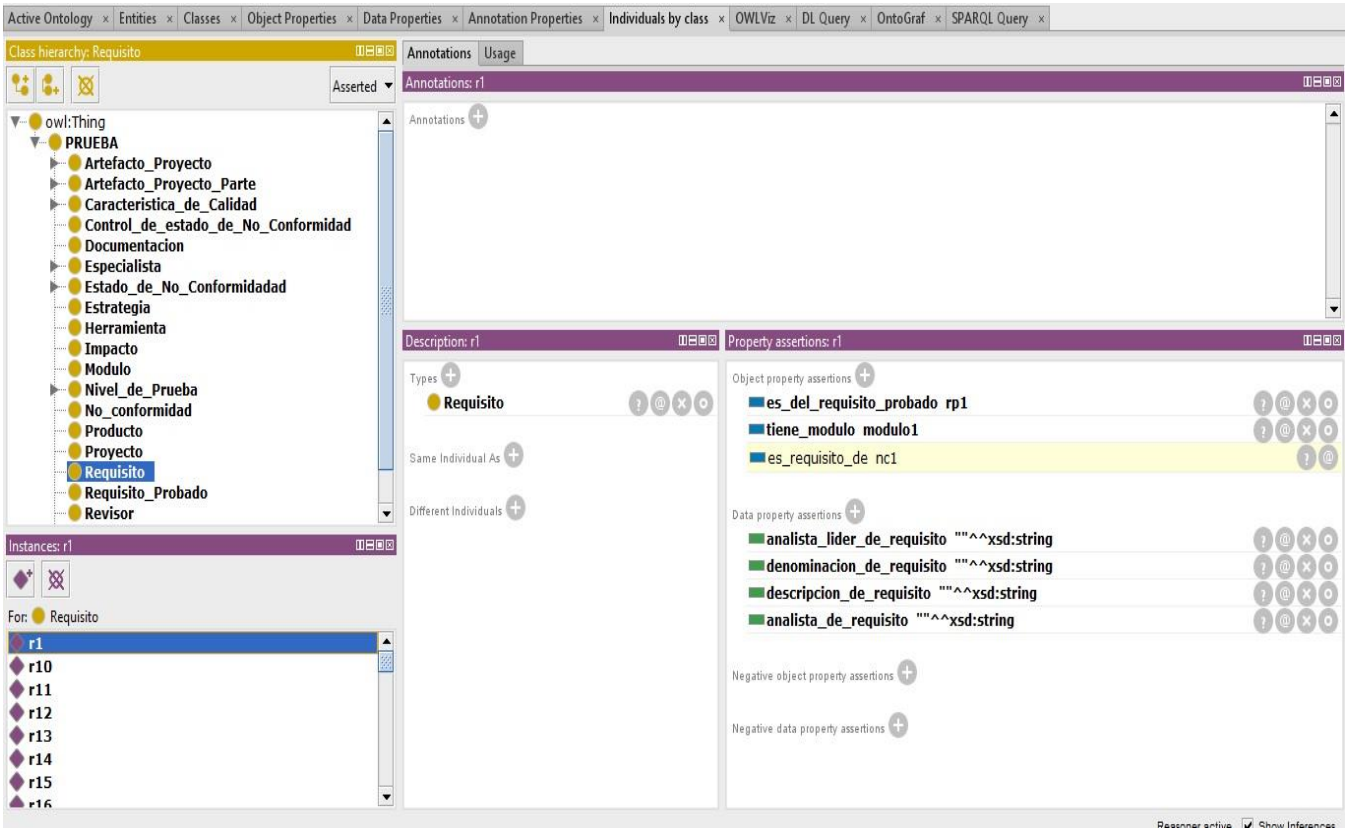
Para dar respuesta a esta pregunta se deben seguir una serie de pasos, teniendo en cuenta que para la detección de las no conformidades de un proyecto hay que conocer el producto al que pertenece dicho proyecto, los módulos que tiene este producto, los requisitos a los que responde dicho módulo y luego de analizarse el cumplimiento de estos requisitos es que pueden ser detectadas las no conformidades.

A continuación, se presentan una serie de imágenes con el cumplimiento de cada uno de estos pasos mediante el uso de la herramienta Protégé, dando de esta forma respuesta a la pregunta.


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals by class
OWLViz
DL Query
OntoGraf
SPARQL Query
Class hierarchy: Proyecto
Asserted
Artefacto_Proyecto
Artefacto_Proyecto_Parte
Caracteristica_de_Calidad
Control_de_estado_de_No_Conformidad
Documentacion
Especialista
Estado_de_No_Conformidadad
Estrategia
Herramienta
Impacto
Modulo
Nivel_de_Prueba
No_conformidad
Producto
Proyecto
Requisito
Requisito_Probado
Revisor
Tipo_de_prueba
Tipo_No_Conformidad
Instances: proyecto1
For: Proyecto
proyecto1
proyecto2
proyecto3
Annotations
Usage
Annotations: proyecto1
Description: proyecto1
Types
Proyecto
Same Individual As
Different Individuals
Property assertions: proyecto1
Object property assertions
tiene_especialista especialista1
es_proyecto_de producto1
Data property assertions
descripcion_del_proyecto ""^^xsd:string
denominacion_del_proyecto "Aduana"^^xsd:string
Negative object property assertions
Negative data property assertions
Reasoner active
Show Inferences


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals by class
OWLViz
DL Query
OntoGraf
SPARQL Query
Class hierarchy: Producto
Asserted
Artefacto_Proyecto
Artefacto_Proyecto_Parte
Caracteristica_de_Calidad
Control_de_estado_de_No_Conformidad
Documentacion
Especialista
Estado_de_No_Conformidadad
Estrategia
Herramienta
Impacto
Modulo
Nivel_de_Prueba
No_conformidad
Producto
Proyecto
Requisito
Requisito_Probado
Revisor
Tipo_de_prueba
Tipo_No_Conformidad
Instances: proyecto1
For: Producto
producto1
producto2
producto3
Annotations
Usage
Annotations: producto1
Description: producto1
Types
Producto
Same Individual As
Different Individuals
Property assertions: producto1
Object property assertions
tiene_proyecto proyecto1
es_producto_de modulo10
es_producto_de modulo1
es_producto_de modulo11
Inferred
Data property assertions
denominacion_del_producto "Esen"^^xsd:string
descripcion_del_producto ""^^xsd:string
Negative object property assertions
Negative data property assertions

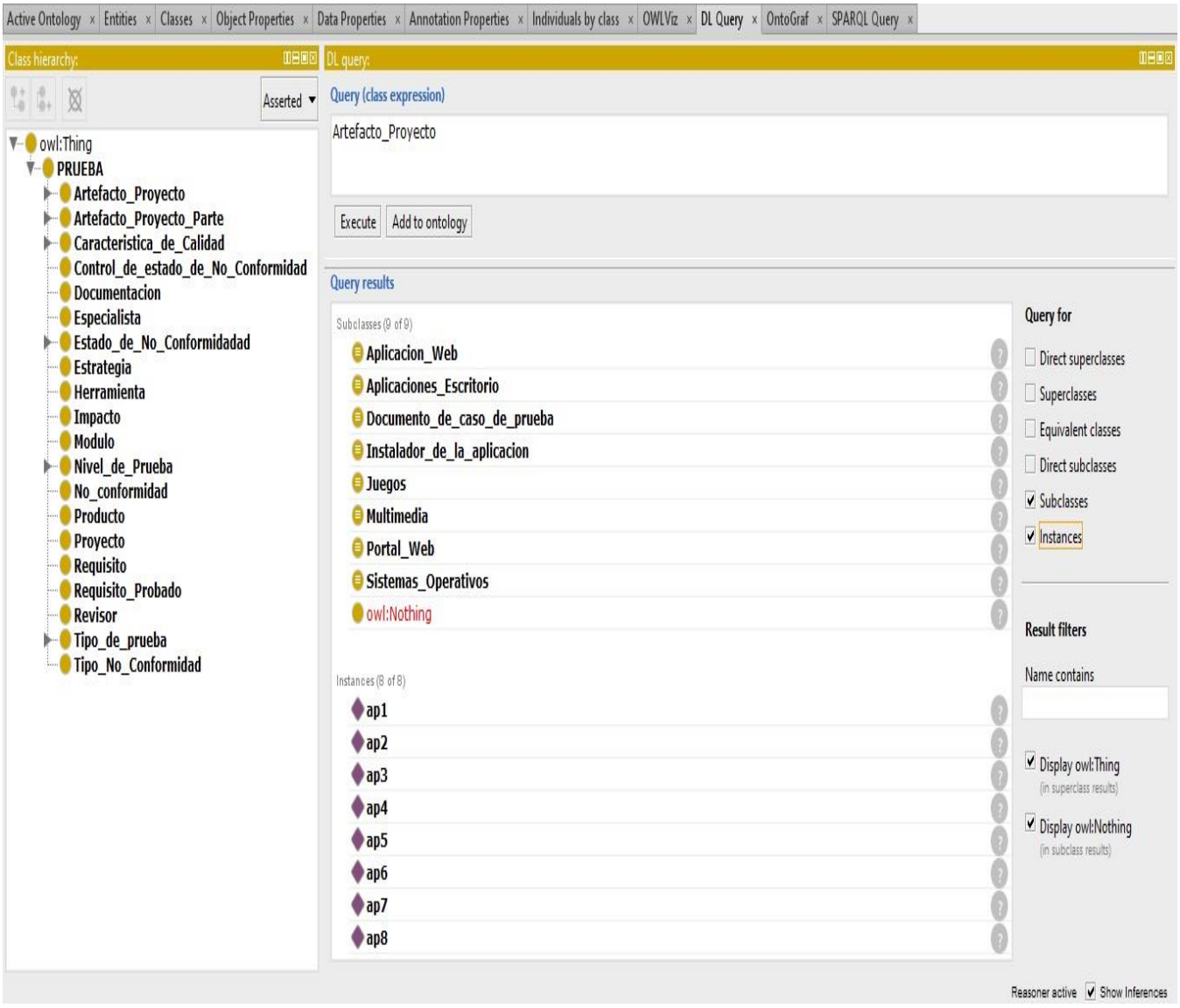

Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals by class
OWLViz
DL Query
OntoGraf
SPARQL Query
Class hierarchy: Modulo
Asserted
Artefacto_Proyecto
Artefacto_Proyecto_Parte
Caracteristica_de_Calidad
Control_de_estado_de_No_Conformidad
Documentacion
Especialista
Estado_de_No_Conformidadad
Estrategia
Herramienta
Impacto
Modulo
Nivel_de_Prueba
No_conformidad
Producto
Proyecto
Requisito
Requisito_Probado
Revisor
Tipo_de_prueba
Tipo_No_Conformidad
Instances: producto1
For: Modulo
modulo1
modulo10
modulo11
modulo12
modulo13
modulo2
modulo3
Annotations
Usage
Annotations: modulo1
Description: modulo1
Types
Modulo
Same Individual As
Different Individuals
Property assertions: modulo1
Object property assertions
tiene_producto producto1
es_modulo_de r4
es_modulo_de r5
es_modulo_de r2
es_modulo_de r3
es_modulo_de r1
Data property assertions
descripcion_de_modulo ""^^xsd:string
denominacion_de_modulo ""^^xsd:string
Negative object property assertions
Negative data property assertions
Reasoner active
Show Inferences


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals by class
OWLViz
DL Query
OntoGraf
SPARQL Query
Class hierarchy: Requisito
Asserted
owl:Thing
PRUEBA
Artefacto_Proyecto
Artefacto_Proyecto_Parte
Caracteristica_de_Calidad
Control_de_estado_de_No_Conformidad
Documentacion
Especialista
Estado_de_No_Conformidadad
Estrategia
Herramienta
Impacto
Modulo
Nivel_de_Prueba
No_conformidad
Producto
Proyecto
Requisito
Requisito_Probado
Revisor
Instances: r1
For: Requisito
r1
r10
r11
r12
r13
r14
r15
Annotations
Usage
Annotations: r1
Description: r1
Types
Requisito
Same Individual As
Different Individuals
Property assertions: r1
Object property assertions
es_del_requisito_probado rp1
tiene_modulo modulo1
es_requisito_de nc1
Data property assertions
analista_lider_de_requisito ""^^xsd:string
denominacion_de_requisito ""^^xsd:string
descripcion_de_requisito ""^^xsd:string
analista_de_requisito ""^^xsd:string
Negative object property assertions
Negative data property assertions

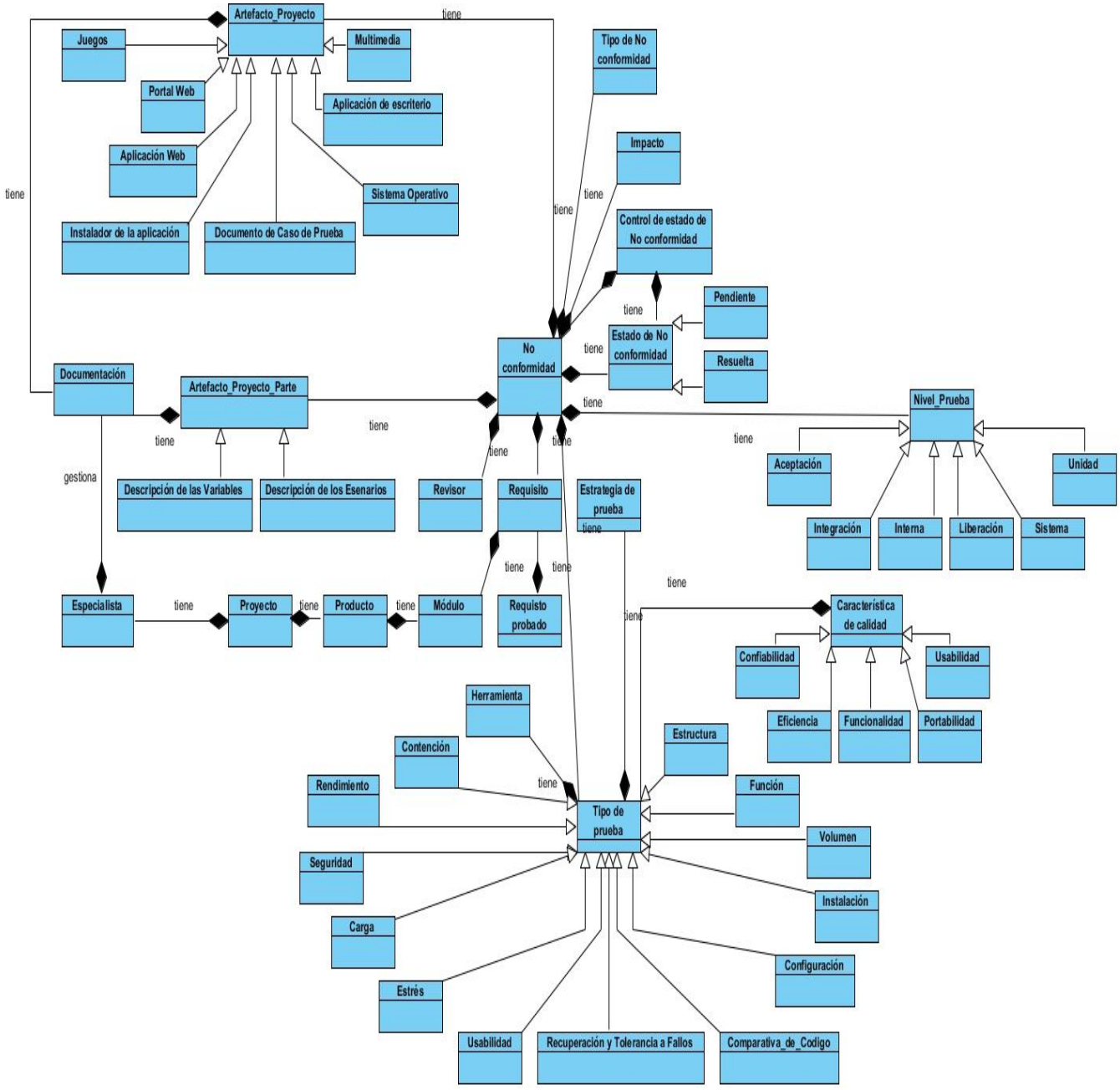
Anexo 2: Consultas a la ontología utilizando el plugins DL Query

El plugin DL Query proporciona una manera de realizar consultas a la ontología destinadas a conocer varios elementos de las clases, dígase subclases, instancias, superclases entre otros.

La siguiente imagen muestra cada una de las subclases e instancias pertenecientes a la clase Artefacto_Proyecto.



Anexo 3: Taxonomía de conceptos



Anexo 4: Diseño de la entrevista

El objetivo de la entrevista fue detallar el conocimiento existente sobre los términos relacionados con el subproceso de prueba de software en las diferentes entidades, su funcionamiento, así como tratar aspectos como la ambigüedad de la información.

Las preguntas realizadas en la entrevista fueron las siguientes:

1. ¿Puede detallar las características del subproceso?
2. ¿Puede describir los términos utilizados para llevar a cabo el subproceso de prueba?
3. ¿Dónde se almacena la información generada en el subproceso de prueba?
4. ¿Qué técnicas de verificación se utilizan?
5. ¿Cuáles son los sistemas informáticos que apoyan a dicho subproceso?
6. ¿Qué estándares se utilizan para el subproceso de prueba de productos de software?
7. ¿Qué herramientas, automatizadas y manuales se utilizan para llevar a cabo el subproceso de prueba de software?
8. ¿Cómo clasifican las no conformidades?
9. ¿Otro comentario que usted quisiera realizar?

Anexo 6: Expertos entrevistados

- MSc. Anaivys Vázquez Abascal: Especialista de la empresa XETID. Vicepresidente del subcomité 7 de la ISO para cuba.
- MSc. Aymara Marin Díaz: Miembro del subcomité 7 de la ISO para cuba.
- MSc. Surima Gé Pérez: Jefa del laboratorio de pruebas en Centro Nacional de Calidad de Software (CALISOFT).