Información Tecnológica
Vol. 27(4), 165-174 (2016)
doi: 10.4067/S0718-07642016000400018

Modelo Ontológico para Contextos de uso de Herramientas de Ingeniería Inversa

Martín E. Monroy[(1)], José L. Arciniegas[(2)] y Julio C. Rodríguez[(1)]
(1) Universidad de Cartagena, Facultad de Ingeniería, Programa de Ingeniería de Sistemas, Piedra de Bolívar, Bolívar – Colombia (e-mail: mmonroyr@unicartagena.edu.co, jrodriguezr@unicartagena.edu.co)
(2) Universidad del Cauca, Departamento de Telemática, Sector Tulcán, Cauca – Colombia (e-mail: jlarci@unicauca.edu.co)

Recibido Dic. 15, 2015; Aceptado Feb. 15, 2016; Versión final Feb. 19, 2016, Publicado Ago. 2016

Resumen

Se presenta un modelo ontológico para seleccionar las herramientas de ingeniería inversa más adecuadas ante un contexto de uso específico. El modelo ontológico se elaboró aplicando la propuesta metodológica de la Universidad de Stanford, establecida por Noy y McGuinness y se validó con 38 herramientas de ingeniería inversa. Se implementó la ontología utilizando la herramienta Protégé y se identificaron 32 características determinantes para describir los contextos de uso de las herramientas de ingeniería inversa, discriminando las que tienen condición de necesaria y las que son deseadas para cada contexto de uso. Se concluye que el modelo ontológico propuesto facilita la selección de herramientas de ingeniería inversa ante un contexto de uso determinado y contribuye a la disminución de los costos de mantenimiento de software, al reducir esfuerzos y optimizar sus procesos.

Palabras clave: herramientas de ingeniería inversa; modelo ontológico; contextos de uso; herramienta Protégé

Ontological Model for Contexts of use of Reverse Engineering Tools

Abstract

An ontological model to select the most appropriate reverse engineering tools for a specific context of use is presented. The ontological model was developed using the methodology of Stanford University, established by Noy and McGuinness and was validated with 38 reverse engineering tools. The ontology was implemented with the Protégé tool and 32 determinant characteristics were identified to describe the contexts of use of reverse engineering tools, discriminating those conditions that are necessary and desired for each context of use. It is concluded that the proposed ontological model facilitates the selection of reverse engineering tools for a particular context of use and contributes to lowering the costs of software maintenance, by reducing efforts and optimizing processes.

Keywords: reverse engineering tools; ontological model; contexts of use; Protégé tool

INTRODUCCIÓN

Desde su génesis la ingeniería inversa ha sido utilizada como técnica de mantenimiento de software (IEEE, 2012), sin embargo, a lo largo de su evolución ha servido para resolver varios problemas que afronta la ingeniería de software, tales como: la recuperación de arquitecturas y patrones de diseño, la re-documentación de programas y bases de datos, la identificación de activos reutilizables, la definición de la trazabilidad entre artefactos de software, la identificación del impacto de los cambios del producto software, la reestructuración de sistemas existentes, la renovación de interfaces de usuario, la migración hacia nuevas arquitecturas y plataformas, entre otros (Canfora et al, 2011). Esta característica ha hecho que los campos de aplicación de la ingeniería inversa se extiendan a distintos contextos, tales como: La producción de software, la seguridad informática y la educación, para cada uno de los cuales se describe a continuación la forma como es utilizada la ingeniería inversa.

1) Producción de software: Comprende todos los procesos que definen el ciclo de vida del software, establecidos en el estándar ISO/IEC 12207:2008. La ingeniería inversa se utiliza concretamente en los siguientes procesos:

i) Mantenimiento de software: El estándar para el mantenimiento de software IEEE – 1219 recomienda la ingeniería inversa como la principal estrategia para tratar sistemas cuya única representación fiable es el código fuente (Canfora et al, 2011), debido a que la comprensión del software consume una parte sustancial del esfuerzo de mantenimiento. Adicionalmente se utiliza para la identificación de errores, al permitir examinar el sistema con el fin de localizar posibles defectos que presente, facilitando su modificación y actualización al implementar cambios para mejorarlo o ajustarlo a las nuevas necesidades, convirtiéndose en un paso obligatorio de los procesos de reingeniería (Chikofsky y Cross, 1990).

ii) Documentación: La ingeniería inversa se convierte en la principal estrategia para reconstruir la documentación de un sistema software, al lograr una representación de más alto nivel que permita su comprensión al recuperar las vistas de diseño y arquitectura, los requerimientos e incluso los modelos del negocio (Van Geet et al, 2010). También se puede utilizar para mantener el control de versiones en sistemas legados a partir de sus implementaciones, porque facilita la recuperación de artefactos en sistemas que han sido desarrollados hace mucho tiempo y que son difíciles de actualizar, haciendo posible establecer diferencias entre las distintas versiones que ha presentado el producto software (El Boussaidi et al, 2012; Van Geet et al, 2010).

iii) Verificación de software: La ingeniería inversa es una excelente herramienta de control de calidad al momento de verificar la coherencia y trazabilidad entre el código fuente y: los modelos de diseño, la arquitectura, el modelo de negocio y los requisitos establecidos para del producto software. Desde la perspectiva del enfoque conocido como Arquitectura Dirigida por Modelos (MDA), se plantea la necesidad de mantener los modelos en tiempo de ejecución, lo cual consiste en sincronizar los modelos y el código fuente de tal forma que al modificarse uno, el otro se actualice respondiendo a dicho cambio, lo cual es posible gracias a la ingeniería inversa (Jouault et al, 2009).

iv) Reutilización de software: Este proceso consiste en un conjunto de actividades que soportan la habilidad que tiene la organización para adquirir, desarrollar y gestionar activos software reutilizables. La ingeniería inversa juega un papel importante al momento de identificar piezas de software que puedan ser utilizadas en nuevas soluciones a partir de desarrollos previos, evitando así la necesidad de volver a implementarlas (Vasconcelos y Werner, 2011). Por otra parte, cada vez es mayor la importancia que toma la ingeniería inversa en contextos de líneas de producción de software, sobre todo al momento de mantener la consistencia de la arquitectura de la familia de productos (Wu et al, 2011; Stoermer y O'Brien, 2001).

2) Seguridad informática: Como disciplina se encarga de definir normas, procedimientos, métodos y técnicas destinados a garantizar que un sistema de información sea seguro y confiable. En este campo se utiliza la ingeniería inversa con el fin de analizar y entender el código malicioso que pueda estar presente en el sistema, para facilitar la definición de estrategias a seguir en la solución del riesgo e inconvenientes que genera (Treude et al, 2011).

3) Educación: Para el Constructivismo Filosófico de Kant, el principal postulado indica que “el conocimiento humano no se recibe de forma pasiva sino que, más bien, es procesado y construido de una forma activa por el individuo que realiza la actividad de conocer y que, gracias a su aparato cognitivo, puede ir adaptando y modificando el objeto de estudio sobre el cual actúa, lo que permite al conocedor, (alumno o aprendiz, hablando en términos de aprendizaje) organizar su mundo, interactuar con él y registrar sus experiencias desde una perspectiva individual y vivencial” (Flórez, 1994, p.235, citado por Heredia 2012).

En este sentido, la ingeniería inversa se convierte en una herramienta didáctica que posibilita: 1) El aprendizaje a partir de éxitos y fallas reales, porque hace posible el uso de productos software para

mostrarle a los estudiantes el díselo y arquitectura de los casos de éxito con el fin de replicarlos, así como también los fracasos para evitarlos. 2) El desarrollo de la curiosidad, estimulando el interés de los estudiantes al evidenciar la importancia de los modelos en los procesos de desarrollo de software, tomando como referentes implementaciones reales. 3) La recuperación del conocimiento representado en productos software desarrollados, para facilitar la comprensión de conceptos en el campo de la ingeniería de software, como es el caso de la identificación de patrones aplicando técnicas de ingeniería inversa (Tonella et all, 2007). 4) El desarrollo de habilidades para modelar, programar y hacer mantenimiento a productos software a partir implementaciones reales y 5) la corrección de errores en el proceso de aprendizaje, mediante la utilización de herramientas de ingeniería inversa que permitan identificar posibles errores de diseño, o identificar inconsistencias entre los modelos de diseño y la implementación realizada (Ali, 2005).

Según el documento SWEBOK (2012) existen estudios que afirman que la mayor parte de los costos del mantenimiento de software están relacionados con actividades de apoyo, que comprenden tareas de re-documentación y entendimiento de sistema. Afortunadamente, durante las últimas dos décadas se han propuesto muchos métodos y herramientas de ingeniería inversa (Monroy et al, 2013), lo cual a su vez genera un nuevo problema: ¿Cómo seleccionar la herramienta de ingeniería inversa más adecuada ante un contexto de uso específico?. Al responder a esta pregunta se pretende facilitar la toma de decisión a los interesados en el uso de herramientas de ingeniería inversa, contribuir a la disminución de costos de trabajos como el mantenimiento de software, al reducir esfuerzos y optimizar sus procesos.

La revisión de la literatura permitió identificar la existencia de 160 herramientas de ingeniería inversa y establecer que existen varias propuestas para solucionar el interrogante planteado, las cuales se pueden clasificar en dos grupos: El primero corresponde a los estudios que evalúan herramientas de ingeniería inversa y el segundo a propuestas de caracterización de estas herramientas. En el primer grupo se resaltan los trabajos realizados por Rasool et al (2010), quienes elaboran una guía para comparar las características de las herramientas existentes, centrando la atención exclusivamente en herramientas para la recuperación de patrones de diseño, al igual que lo hacen Pettersson et al (2006) y Wang et al (2004). Algo similar realizan Pacione et al (2003), Kollmann et al. (2002), Systä et al (2001) y Storey et al (1996), al hacer evaluaciones comparativas entre herramientas de visualización.

Todas las propuesta mencionadas se limitan al proceso de evaluación de las herramientas, sin llegar a proponer estrategias o recomendaciones para realizar la selección, atendiendo al contexto de uso que se le quiera dar a la herramienta, por lo tanto, sólo dan respuesta parcial a la pregunta formulada, porque sirven de guía para evaluar las herramientas pero no para seleccionar la más adecuada. Con respecto al grupo de estudios que se centran en la caracterización de herramientas de ingeniería inversa, uno de los trabajos más detallados es el realizado por Bellay y Gall (1997), quienes basados en su experiencia definen cuatro categorías funcionales para evaluar herramientas de ingeniería inversa: Análisis, representación, edición/navegación y capacidades generales, cada una de las cuales a su vez está conformada por un conjunto de criterios específicos.

También se encuentra la propuesta de Guéhéneuc et al (2006) quienes plantean la necesidad de contar con un marco de referencia comparativo común bien definido y fácil de comprender, que haga menos complejo el trabajo de evaluación de herramientas de recuperación de diseño, para lo cual proponen un marco de referencia comparativo común, construido a partir de su experiencia, en calidad de extensión de otros marcos comparativos existente, analizando las siguientes características: contexto en el cual se aplica, intención, tipos de usuarios, tipos de entradas, técnicas utilizadas, tipos de salidas que provee, como está implementada y grado de madurez de la herramienta.

Estas dos propuestas fueron utilizadas por Monroy et al (2012) como referentes para definir un instrumento para la caracterización de herramientas de ingeniería inversa, en el cual proponen una estructura de caracterización basada en dos criterios: el aspecto estructural y las propiedades comunes entre ellas, a partir de la cual se puede hacer una valoración independiente y conjunta de cada uno de los componentes que conforman las herramientas de ingeniería inversa, sin embargo, ninguno de estos trabajos propone estrategias o recomendaciones para realizar la selección, atendiendo al contexto de uso que se le quiera dar a la herramienta.

Por otra parte, se encuentra la aplicación de un enfoque comparativo llamado BEFRIEND (BEnchmark For Reverse engInEering tools workiNg on source coDe), de Jeno Fülöp (2011), con el que se puede evaluar y comparar con facilidad y eficiencia las salidas de las herramientas de ingeniería inversa, tales como, herramientas de minería de patrones de diseño, detectores de duplicación de código y verificadores de violación de las reglas. BEFRIEND soporta diferentes tipos de familias de herramientas, lenguajes de programación y sistemas de software, permitiendo a los usuarios definir sus propios criterios de evaluación.

Adicionalmente, existen propuestas genéricas para evaluar, comparar y seleccionar sistemas hardware y software complejos, como la puntuación lógica de preferencias (Dujmović, 1966; Dujmović, 2007), que requieren de un proceso exhaustivo que incluye estudio de factibilidad, especificación de variables de desempeño, definición de criterios, especificación de la estructura de preferencias, solicitud de propuestas y preparación de propuestas, entre otras actividades. Para dar respuesta al interrogante planteado se realizó una investigación, estableciendo como objetivo la definición de un modelo ontológico para seleccionar las herramientas de ingeniería inversa más adecuadas ante un contexto de uso específico, obteniendo como resultado la definición de las características de los contextos de uso de las herramientas de ingeniería inversa y el modelo ontológico. A continuación se describe la metodología utilizada, se presenta la propuesta, se hace el análisis de los resultados, se proponen trabajos futuros y finalmente se presentan las conclusiones.

METODOLOGÍA

El modelo ontológico se elaboró aplicando la propuesta metodológica de la Universidad de Stanford, establecida por Noy y McGuinness (2001) porque además de ser simple es la más referenciada en la literatura. Esta metodología comprende los siguientes pasos: El primer paso consiste en determinar el dominio y el alcance de la ontología, lo que implica dar respuesta a preguntas como: ¿Cuál es el dominio que la ontología comprenderá?, ¿para qué se utilizará la ontología?, ¿para qué tipo de preguntas la ontología debe proveer respuesta? y ¿quién usará y mantendrá la ontología?.

El segundo paso implica considerar la reutilización de ontologías existentes con el propósito de evitar hacer más eficiente el trabajo. En el tercer paso se deben enumerar los términos importantes en la ontología, para lo cual se debe hacer una lista de todos los términos con los cuales se pueden hacer frases acerca del dominio del problema. Los pasos cuarto y quinto están estrechamente entrelazados y se hacen en forma paralela En el cuarto paso se definen las clases y su jerarquía utilizando estrategias top-down, bottom-up o una combinación de estas (Uschold and Gruninger, 1996), mientras que en el quinto paso se definen los atributos de las clases, que pueden ser intrínsecos, extrínsecos o representar relaciones entre ellas.

En el sexto paso se definen las restricciones de los atributos en términos de su dominio y el rango, su cardinalidad y el tipo (cadena, número, lógico, enumeración o instancia) y finalmente se crean las instancias de la jerarquía definiendo cada individuo que conforma la ontología. Para la creación de la ontología se utilizó la herramienta Protégé en su versión 5.0 desarrollado por la Universidad de Stanford (2015), por ser un reconocido editor de ontologías de código abierto fácil de usar. Para la validación del modelo ontológico se utilizaron los resultados de la caracterización obtenidos en estudios previos (Monroy et al, 2012), registrando como individuos en la ontología los valores correspondientes a 38 herramientas de ingeniería inversa y generando consultas en SPARQL, tomando como referente la definición de las características de los contextos de uso establecidas por este trabajo.

PROPUESTA

Como resultado de la aplicación de cada uno de los pasos establecidos en la metodología, los autores proponen un modelo ontológico que se describe a continuación. Además, se identificaron las características determinantes de los contextos de uso de las herramientas de ingeniería inversa.

Modelo Ontológico: El dominio de la ontología está determinado por los contextos de uso de las herramientas de ingeniería inversa en el ámbito de la ingeniería de software. Esta ontología se utiliza para facilitar las tareas de toma de decisión con respecto al uso de herramientas de ingeniería inversa, a partir de las circunstancias que describen el contexto de uso, por lo tanto está orientada para ser utilizada por ingenieros de software, docentes y estudiantes en este campo del conocimiento, además de los profesionales relacionados con los contextos de uso de la ingeniería inversa establecidos en el marco teórico de este artículo.

En consecuencia, la ontología ha sido diseñada principalmente para responder la pregunta ¿Qué herramientas de ingeniería inversa se pueden utilizar ante un contexto de uso determinado?. Además puede responder entre otras a las siguientes preguntas: ¿Qué herramientas cumplen con una función específica?, ¿Qué herramientas cumplen con una característica determinada?, ¿Qué funciones se tienen en cuenta en un contexto específico? ¿Qué características se tienen en cuenta en un contexto específico?. En la revisión bibliográfica que se realizó para establecer el estado del arte y el marco teórico, y en la búsqueda en los repositorios de ontologías que ofrece DAML (The DARPA Agent Markup Language) (sf) y la Universidad de Stanford a través de Ontolingua (sf), no se encontró ninguna ontología que pudiera ser reutilizada. Sin embargo, se encontraron documentos que clasifican y caracterizan herramientas de ingeniería inversa, que sirvieron como referentes para definir la ontología (Monroy et al, 2012; Fülöp et al, 2008; Guéhéneuc et al, 2006; Bellay y Harald, 1997).

Los resultados de la aplicación de los demás pasos propuestos por la metodología se sintetizan en la figura 1, donde se muestra el modelo ontológico por medio de la representación del modelo de dominio, usando la notación de UML, presentando las clases, su jerarquía y las relaciones entre ellas, manifestando la semántica descrita a continuación. La educación, la producción de software y la seguridad informática son contextos de uso que requieren de herramientas de ingeniería inversa, teniendo en cuenta que producción de software a su vez tiene procesos como el mantenimiento, la documentación, la verificación de software y la reutilización.

En su estructura, según la caracterización hecha por Monroy et al (2012), las herramientas de ingeniería inversa tienen dos elementos: el analizador, responsable de la conversión del código fuente a un nivel mayor de abstracción, y la interfaz que permite la interacción entre el usuario y la herramienta. Una función corresponde a las tareas que realiza cada elemento de la arquitectura, por ejemplo el analizador cumple con tareas de entrada, proceso y salida. El aspecto indica un matiz o rasgo diferenciador para cada función que cumplen los elementos. La característica, entendida como una cualidad que sirve para distinguir a cada uno de los aspectos identificados para las funciones que cumplen los elementos, y los factores que distinguen las propiedades comunes a cualquier producto software, como es el caso de las herramientas de ingeniería inversa.

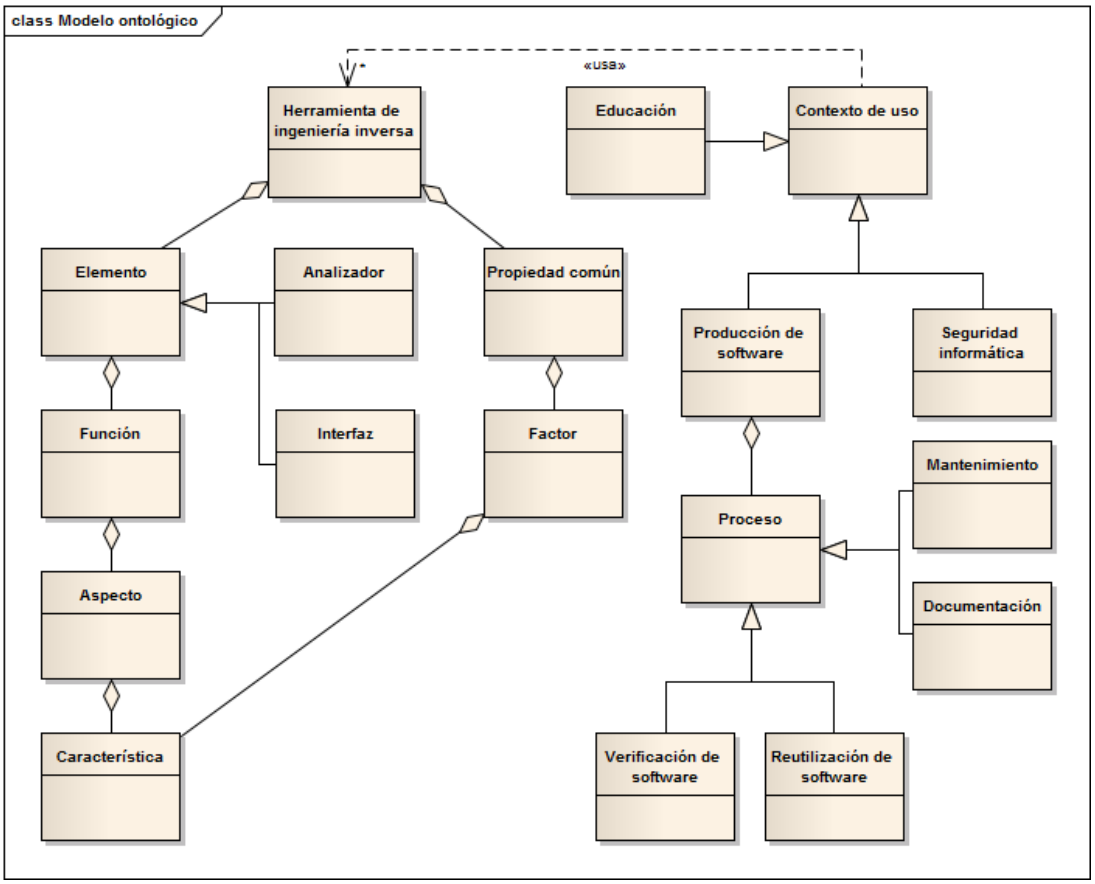

Fig. 1: Modelo ontológico

Características de los contextos de uso: Se establecen a partir de la propuesta de caracterización de herramientas de ingeniería inversa (Monroy et al, 2012), según la cual se definen 82 características clasificadas teniendo en cuenta el elemento, la función y el aspecto. Debido a que no todas las características son determinantes del contexto de uso, para cada una se evaluó su nivel de pertinencia para cada contexto de uso (M: Mantenimiento, V: Verificación de software, Dc: Documentación, R: Reutilización, S: Seguridad informática y E: Educación), obtenido los resultados relacionados en la tabla 1, donde se omiten las características valoradas como no determinantes.

Esto se logró estableciendo la siguiente escala de valoración: La característica es obligatoria (O) para el contexto de uso si al no cumplirse no se le logra el propósito del contexto de uso; la característica es necesaria (N), si contribuye al logro del propósito, pero si no se cumple no implica que no se logre el objetivo del contexto de uso; la característica es deseada (D) si ayuda a mejorar los resultados del propósito del contexto de uso; y finalmente, la característica se valora como no determinante (ND) si no afecta para nada el cumplimiento del propósito del contexto de uso.

Tabla 1: Características de los contextos de uso

Características	*M*	*V*	*Dc*	*R*	*S*	*E*
Explicación de la técnica	D	D	D	ND	ND	N
Tratamiento de aspectos indefinidos	D	ND	D	ND	N	ND
Tipo de analizador estático	N	N	N	D	N	N
Tipo de analizador dinámico	D	D	D	D	D	D
Tipo de analizador Híbrido	D	D	D	D	D	D
Tipo de analizador Histórico	D	ND	D	D	ND	D
Reconocimiento de Patrones	N	N	D	O	D	D
Capacidad para analizar Funciones/Variables Externas	D	ND	ND	D	O	ND
Capacidad para trabajar con información incompleta	D	D	D	D	O	D
Capacidad para trabajar con caja negra	D	D	D	D	D	D
Identificación de clonación	N	D	ND	N	D	D
Ingeniería Inversa de documentos WSDL en UML	D	D	D	ND	ND	D
Legibilidad del reporte	N	N	O	N	N	O
Calidad del reporte	N	N	N	N	D	N
Trazabilidad Horizontal	O	O	D	ND	N	D
Trazabilidad Vertical	O	O	N	D	N	D
Visualización Estática	N	N	N	N	N	N
Visualización Dinámica	N	N	N	D	N	N
Niveles de Abstracción: Código	O	O	D	N	N	O
Niveles de Abstracción: Diseño	O	O	O	N	N	O
Niveles de Abstracción: Arquitectura	N	N	N	D	N	D
Niveles de Abstracción: Requerimientos	D	D	D	D	D	D
Niveles de Abstracción: Modelo del Negocio	D	D	D	D	D	D
Capacidades de Hipertexto	N	N	N	D	D	N
Capacidad para analizar diferencias	N	N	D	D	ND	D
Mecanismos de consulta	N	N	D	D	D	D
Capacidad para agrupar cambios relacionados	N	N	N	D	ND	N
Anotaciones de ayuda	D	D	D	ND	D	D
Permite organizar el informe	N	N	N	ND	D	N
Generación automática de documentación	N	N	O	ND	D	N
Función de Búsqueda	N	N	N	N	N	N
Capacidad para seleccionar entradas representativas	N	N	N	D	D	D

ANÁLISIS DE LOS RESULTADOS

Para la creación de la ontología se utilizó la herramienta Protégé en su versión 5.0 (Stanford University, 2015). Se definieron los conceptos establecidos en el modelo ontológico como clases y en calidad de propiedades de los objetos se asignaron las relaciones. Para cada clase se establecieron las clases disjuntas, su jerarquía, sus equivalentes y sus miembros en caso de tenerlos. De igual forma se hizo con las propiedades de objetos, registrando si corresponde a una propiedad funcional, o una función inversa, transitiva, simétrica, asimétrica, reflexiva o irreflexiva. Finalmente se registraron como individuos los contextos de uso explicados en la introducción y las 38 herramientas de ingeniería inversa caracterizadas en estudios previos (Monroy et al, 2012).

La utilidad de la ontología se evidencia al aplicar las preguntas para las cuales fue diseñada, siendo para este proyecto la más importante: ¿Qué herramientas de ingeniería inversa se pueden utilizar ante un contexto de uso determinado?. Esta pregunta se formula con el lenguaje SPARQL y haciendo uso del entorno Protégé (Stanford University, 2015), como se indica en la figura 2, donde se consulta a la ontología sobre las herramientas de ingeniería inversa que se pueden utilizar en un contexto de documentación de software, teniendo en cuenta que las características obligatorias que deben cumplir para dicho contexto son: Legibilidad del reporte, diseño como nivel mínimo de abstracción y capacidad de generar automáticamente documentación.

De esta manera se pueden formular preguntas a la ontología, resaltando aquellas características que la persona considere pertinentes según sea necesario, facilitando la tarea de selección de la herramienta más adecuada, disminuyendo los tiempos destinados a actividades complementarias y mejorando los procesos al permitir centrar la atención en el propósito planteado según el contexto de uso, ya sea el mantenimiento de software con sus diferentes actividades, algún problema de seguridad informática tendiente a comprender software malicioso, o alguna actividad de aprendizaje.

A diferencia de las propuestas anteriores, que se limitan a comparar herramientas de ingeniería inversa, por medio de la ontología definida se pueden identificar y seleccionar herramientas que realicen una función específica o tengan una característica determinada, al mismo tiempo que se pueden identificar las funciones y características que se tienen en cuenta en un contexto específico de uso. Sin embargo, es pertinente aclarar que el alcance de la ontología se limita a determinar los contextos de uso de las herramientas de ingeniería inversa que se caractericen como individuos, teniendo en cuenta que en la revisión bibliográfica hecha por los autores, aunque se identificaron 160 herramientas, no todas cuentan con la documentación suficiente para caracterizarlas en la ontología.

Por otra parte, se debe hacer más fácil la forma de plantear las preguntas a la ontología, generando un herramienta que a través de una interfaz amigable convierta cada pregunta en la respectiva consulta SPARQL. De igual forma, se hace necesario simplificar la tarea de actualización y edición de los individuos que conforman la ontología, por eso se propone como trabajo futuro la realización de un software que cumpla con estas funciones.

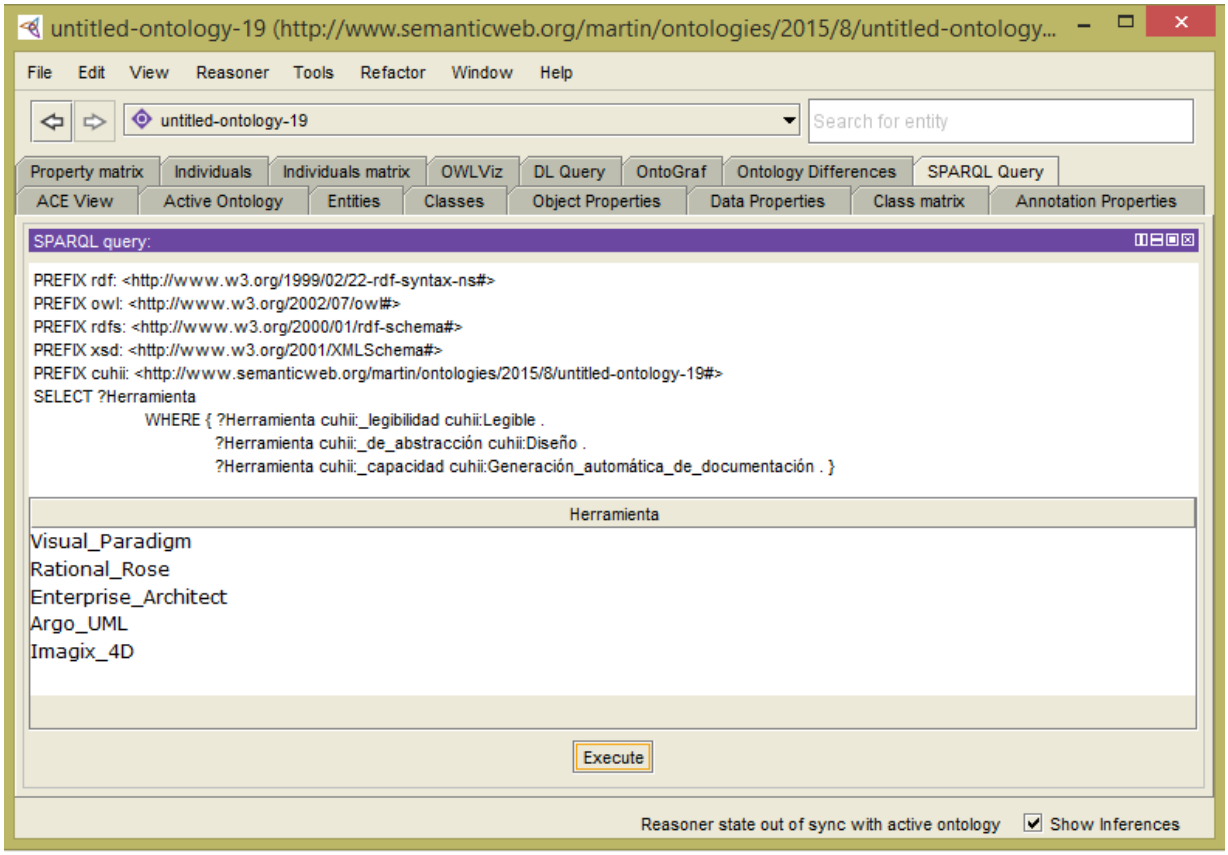

Fig. 2: Consulta a la ontología

CONCLUSIONES

Se definió un modelo ontológico para facilitar la selección de herramientas de ingeniería inversa ante los posibles contextos de uso en el campo de la ingeniería de software, la seguridad informática y la educación, estableciendo las características necesarias y deseadas para cada contexto de uso, tomando como referente la propuesta de caracterización de herramientas de ingeniería inversa presentada por Monroy et al (2012). Estos resultados permiten concluir que el uso de la ontología definida facilita la selección de herramientas de ingeniería inversa, teniendo en cuenta la necesidad concreta de un contexto de uso determinado, lo que contribuye a la disminución de los costos de trabajos como el mantenimiento de software, al reducir esfuerzos y optimizar sus procesos.

AGRADECIMIENTOS

Este trabajo es parte de los resultados parciales de la tesis doctoral titulada Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento, desarrollada por Martín Monroy Ríos en el programa de Doctorado de la Universidad del Cauca. Agradecemos a la Red Latinoamericana de Coordinación y Cooperación para unificar buenas prácticas de desarrollo de Software - Proyecto SPU CoCoNet-LA por su apoyo.

REFERENCIAS

Ali, M. R. Why Teach Reverse Engineering?, ACM SIGSOFT Software Engineering Notes, vol. 30, n.º 4, pp. 1-4, jul. (2005)

Bellay B. y G. Harald, A Comparison of four Reverse Engineering Tools, 4th Working Conference on Reverse Engineering, Amsterdam, Holanda, 2 – 11, 6 al 8 de octubre (1997)

Canfora G., Di Penta M. y L. Cerulo. Achievements and Challenges in Software Reverse Engineering, communications of the ACM 54(4) 142–151. (2011)

Chikofsky E. y J. Cross. Reverse Engineering and Design Recovery: a Taxonomy. IEEE Software, vol. 7, no. 1, pp. 13–17, (1990)

DARPA's Information Exploitation Office, Ontologies by class, http://www.daml.org/ontologies/class.html, acceso: 10 de enero (2015)

Dujmović, J. J. A method for evaluation and selection of complex hardware and software systems. *CMG 96 Proceedings*. (1996)

Dujmović, J. J. Continuous preference logic for system evaluation. *Fuzzy Systems, IEEE Transactions on*: 1082-1099, 15.6 (2007)

El Boussaidi G., Belle A., Vaucher S. y H. Mili. Reconstructing Architectural Views from Legacy Systems, 19th Working Conference on Reverse Engineering, Kingston, ON, pp. 345 – 354. (2012)

Fülöp, L.J., Hegedus, P., Ferenc, R. y T. Gyimothy, Towards a Benchmark for Evaluating Reverse Engineering Tools. 15th Working Conference on Reverse Engineering. 335 – 336, Antwerp, Bélgica, 15 al 18 de octubre (2008)

Guéhéneuc Y., Mens K. y R. Wuyts, A Comparative Framework for Design Recovery Tools, Proceedings of the 10th European Conference on Software Maintenance and Reengineering, 134 – 143, Bari, Italia, 22 al 24 de marzo (2006)

Heredia, Y. y Sánchez, A. Teorías del Aprendizaje en el Contexto Educativo. Editorial Digital, Tecnológico de Monterrey, México. (2012)

IEEE. SWEBOK Guide to the Software Engine Body of Knowledge. Version 3. (2012)

Jouault F., Bézivin J. y M. Barbero. Towards an advanced Model Driven Engineering Toolbox. Innovations in Systems and Software Engineering, vol. 5, no. 1, pp. 5–12, mar. (2009)

Kollmann R., Selonon P., Stroulia E., Systä T. and A. Zündorf. "A Study on the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering", Proceedings of the 9th Working Conference on Reverse Engineering, IEEE CS Press, Los Alamitos, CA, pp. 22- 33. (2002)

Monroy, M., Arciniegas J. L. y J. Rodríguez, Caracterización de herramientas de ingeniería inversa. Información Tecnológica, 23(6), 31-42 (2012)

Monroy, M., Arciniegas J. L., y J. C. Rodríguez. Propuesta Metodológica para Caracterizar y Seleccionar Métodos de Ingeniería Inversa. Información tecnológica, 24(5), 23-30 (2013)

Noy, N. F. y McGuinness, D. L. Ontology development 101: A guide to creating your first ontology. (2001)

Pacione, M.J. "A Review and Evaluation of Dynamic Visualisation Tools", Technical Report EFoCS-50. (2003)

Pettersson, N., Löwe, W., y Nivre, J., On Evaluation of Accuracy in Pattern Detection. (2006)

Rasool G., Maeder P., y I. Philippow, Evaluation of design pattern recovery tools, ScienceDirect Procedia Computer Science, 3, 813 – 819 (2010)

Stanford University, Protégé, http://protege.stanford.edu/products.php#desktop-protege, acceso: 13 de enero (2015)

Stanford University, Ontolingua, http://www.ksl.stanford.edu/software/ontolingua/, acceso: 18 de enero (2015)

Stoermer, C., y O'Brien, L., MAP-mining architectures for product line evaluations. In Software Architecture, Proceedings. Working IEEE/IFIP Conference on (pp. 35-44). IEEE. (2001)

Storey, M.-A., Wong, K., Fong, P., Hooper, D., Hopkins, K., y Müller, H. On Designing an Experiment to Evaluate a Reverse Engineering Tool. Proceedings of the 3rd Working Conference on Reverse Engineering, 31-40. (1996)

Systä T., Koskimies K. y H. Müller, "Shimba – an Environment for Reverse Engineering Java Software Systems", Software – Practice and Experience 31(4), Wiley, New York, NY, pp. 371-394. (2001)

Treude C., Figueira F. y M. Storey. An Exploratory Study of Software Reverse Engineering in a Security Context. 18th Working Conference on Reverse Engineering, Limerick, Ireland, pp. 184 – 188. (2011)

Tonella P., Torchiano M., Du Bois B, y T. Systä. Empirical studies in reverse engineering: state of the art and future trends. Empirical Software Engineering, vol. 12, no. 5, pp. 551 – 571, Oct (2007)

Van Geet J., Ebraert P. y S. Demeyer. Redocumentation of a legacy banking system: an experience report. International Workshop on Principles of Software Evolution, Antwerp, Belgium, pp. 33–41, (2010)

Vasconcelos A. y C. Werner. Evaluating reuse and program understanding in ArchMine architecture recovery approach. Journal Information Sciences: an International Journal, vol. 181, no. 13, pp. 2761–2786, Jul. (2011)

Wang, W., y Tzerpos, V., DPVK - An Eclipse Plug-in to Detect Design Patterns in Eiffel Systems, Electronic Notes in Theoretical Computer Science (ENTCS), Volume 107, pp. 71-86, (2004)

Wu, Y., Yang, Y., Peng, X., Qiu, C., y Zhao, W. Recovering object-oriented framework for software product line reengineering. Top Productivity through Software Reuse. Springer Berlin Heidelberg, 119-134. (2011)