UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Santa Fe

Doctorado en Ingeniería

Mención en Ingeniería en Sistemas de Información

Tesis Doctoral

"Marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías"

Ing. Emiliano Reynares

Director Dra. María Rosa Galli

Codirector Dra. María Laura Caliusco

Santa Fe, Argentina. Febrero de 2015

Reynares, Emiliano
Marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías - 1a ed. - Santa Fe : el autor, 2015.
224 p. ; 29x21 cm.

ISBN 978-987-33-7432-6

1. Ontología. I. Título
CDD 111

Fecha de catalogación: 06/02/2015

UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Santa Fe

Comisión de Posgrado

Se presenta esta Tesis en cumplimiento de los requisitos exigidos por la Universidad Tecnológica Nacional para la obtención del grado académico de Doctor en Ingeniería, mención Sistemas de Información

"Marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías"

por

Ing. Emiliano Reynares

Director Dra. María Rosa Galli

Codirector Dra. María Laura Caliusco

Jurados de Tesis

Dr. Andrés Díaz Pace

Dra. Alicia Díaz

Dra. Marcela Vegetti

Santa Fe, Argentina. Febrero de 2015

A todos los que me acompañaron en esta increíble aventura

Índice General

Índice de Tablas

Índice de Figuras

Publicaciones relacionadas

Los avances parciales de la tesis que se presenta a consideración, han sido divulgados mediante las publicaciones que a continuación se detallan.

Artículos:

- REYNARES, E.; CALIUSCO, M.L. y GALLI, M.R. (2015) «A set of ontology design patterns for reengineering SBVR statements into OWL/SWRL ontologies». *Expert Systems with Applications*, **42(5)**, pp. 2680-2690. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2014.11.012. http://www.sciencedirect.com/science/article/pii/S0957417414006988. Source Normalized Impact per Paper (SNIP): 2.362 - SCImago Journal Rank (SJR): 1.487 - Impact Factor: 1.965 - 5 Year Impact Factor: 2.254

- REYNARES, E.; CALIUSCO, M.L. y GALLI, M.R. (2014) «Evaluación empírica del mapeo de reglas de negocio para el desarrollo de ontologías». *Revista Ibérica de Sistemas y Tecnologías de Información (RISTI)*, **14**, pp. 83-99. ISSN 1646-9895. doi: http://dx.doi.org/10.17013/risti.14.83-99. http://www.aisti.eu/risti/risti14.pdf. SCImago Journal Rank (SJR): 0.14

- REYNARES, E.; CALIUSCO, M.L. y GALLI, M.R. (2014) «Approaching the feasibility of SBVR as modeling language for ontology development: An exploratory experiment». *Expert Systems with Applications*, **41(4, Part 2)**, pp. 1576-1583. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2013.08.054. http://www.sciencedirect.com/science/article/pii/S0957417413006751.

Source Normalized Impact per Paper (SNIP): 2.362 - SCImago Journal Rank (SJR): 1.487 - Impact Factor: 1.965 - 5 Year Impact Factor: 2.254

- REYNARES, E.; CALIUSCO, M.L. y GALLI, M.R. (2014) «SBVR to OWL 2 Mappings: An Automatable and Structural-Rooted Approach». *CLEI Electronic Journal (CLEIej)*, **17(3)**. http://www.clei.cl/cleiej/index.html. En prensa.

- AYUB, M.C.; CIAN, A.N.; CALIUSCO, M.L. y REYNARES, E. (2014) «Developing an Ontology-Based Team Recommender System using EDON Method: An Experience Report». *SADIO Electronic Journal of Informatic and Operation Research*, **13(1)**, pp. 1-13. http://www.sadio.org.ar/wp-content/uploads/2014/06/1-Caliusco.pdf.

Trabajos en eventos científico-tecnológicos:

- MARTÍNEZ, A.; SOSA, S.; REYNARES, E. y CALIUSCO, M.L. (2014) «Implementación de Sistemas de Información Basados en Ontologías: Análisis de Tecnologías». Anales de 2do. Congreso Nacional de Ingeniería Informática/Sistemas de Información (CoNaIISI) - Simposio de Ingeniería en Sistemas y de Software, San Luis, Argentina. pp. 1114-1118

- REYNARES, E.; CALIUSCO, M.L. y GALLI, M.R. (2013) «An Automatable Approach for SBVR to OWL 2 Mappings». XVI Conferencia Ibero-Americana en Ingeniería de Software (CIbSE 2013), Montevideo, Uruguay. http://cibse2013.ort.edu.uy/pdf/SE-ISBN-978-9974-8379-1-1.pdf.

- GASPOZ, C.; BERTOSSI, V., REYNARES, E. y CALIUSCO, M.L. (2013) «Applying EDON Methodology and SBVR2OWL Mappings for Building an Ontology-Aware Software». XIX Congreso Argentino de Ciencias de la Computación CACIC 2013 - V Workshop Innovación en Sistemas de Software, Mar del Plata, Argentina. http://sedici.unlp.edu.ar/handle/10915/31610.

- AYUB, M.C.; CIAN, A.N.; CALIUSCO, M.L. y REYNARES, E. (2013) «An Experience Report on using the EDON Method

for Building a Team Recommender System». 42 JAIIO Jornadas Argentinas de Informática - ASSE 2013 Simposio Argentino de Ingeniería de Software (JAIIO - ASSE 2013), Córdoba, Argentina. http://www.42jaiio.org.ar/proceedings/simposios/trabajos/ASSE/02.pdf.

- REYNARES, E.; CALIUSCO, M.L. y GALLI, M.R. (2012) «EDON: A Method for Building an Ontology as Software Artefact». 41 JAIIO Jornadas Argentinas de Informática - ASSE 2013 Simposio Argentino de Ingeniería de Software (JAIIO - ASSE 2012), La Plata, Buenos Aires, Argentina. http://www.41jaiio.org.ar/sites/default/files/372_ASSE_2012.pdf.

Resumen

En los últimos años se ha incrementado el uso de tecnologías semánticas en el desarrollo de sistemas de información organizacionales. Entre dichas tecnologías se destacan las ontologías, las cuales han demostrado brindar beneficios importantes en un amplio espectro de contextos y aplicaciones. Una de las aplicaciones más prometedoras consiste en la utilización de ontologías como medio de encapsular la especificación declarativa del conocimiento del negocio en los sistemas de información, permitiendo la representación sin ambigüedades del conocimiento y la gestión eficiente de entornos reales de alto dinamismo conceptual y procedimental.

Por otra parte, las metodologías para el desarrollo de ontologías aún constituyen un campo abierto de investigación. La mayoría de ellas no proveen suficientes detalles sobre las técnicas y actividades involucradas en su aplicación, ni proveen recomendaciones detalladas acerca de la efectiva utilización de las nociones de reusabilidad y reingeniería, ni presentan métodos o técnicas eficientes para la identificación de los conceptos ontológicos. Además, el proceso de desarrollo de una ontología con las características antes mencionadas debe ser considerado en el contexto de un proceso de desarrollo de software, a fin de facilitar las actividades de evaluación e integración de la ontología y el código de programa.

Probablemente reste cierto tiempo para que la incorporación de ontologías en los sistemas de información se convierta en un enfoque ampliamente reconocido, aceptado e implementado. Los aportes presentados en esta tesis constituyen un avance importante en tal sentido. En primer lugar, se describe un conjunto de patrones de diseño para la obtención de una ontología OWL/SWRL mediante la aplicación de reglas de transformación de meta-modelos sobre la especificación SBVR del dominio del negocio. Las transformaciones se encuentran basadas en la especificación estructural de ambos estándares, proveyendo un conjunto

de mapeos fácilmente utilizable por expertos del negocio o desarrolladores de software. Luego se define un marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías. En este contexto, las ontologías encapsulan el conocimiento del dominio y son utilizadas en tiempo de ejecución por el sistema de software. El marco de trabajo se denomina ***PATRON*** y se encuentra basado en la utilización de los patrones de diseño de ontologías definidos.

Capítulo 1

Introducción

Este capítulo describe el contexto en el cual se enmarca la tesis (Sección 1.1), presenta los antecedentes y motivaciones que originaron la propuesta (Sección 1.2), y expone el objetivo general y los aportes realizados en tal sentido (Sección 1.3). Finalmente, se presenta la organización del resto de la tesis (Sección 1.4).

1.1. Sistemas de información basados en ontologías

La ingeniería de software se encuentra en una búsqueda constante de mejores técnicas de soporte al proceso de desarrollo de sistemas de información. En tal sentido, las líneas de investigación más frecuentemente exploradas son aquellas tendientes a (1) elevar el nivel de abstracción considerado e (2) incrementar la utilización de métodos formales. La conjunción de ambos enfoques permite mejorar aspectos del proceso relacionados a la velocidad de desarrollo, el reuso, la confiabilidad y la reducción de los esfuerzos de mantenimiento. Sin embargo, la elevación del nivel de abstracción y/o el incremento en la utilización de métodos formales no genera beneficios por sí mismos: los modelos formales no siempre resultan apropiados y las abstracciones demasiado elevadas resultan poco utilizables en determinados contextos (Uschold, 2008).

La utilización de ontologías como modelos fundamentales para el desarrollo de sistemas de software representa una evolución en la fusión de ambas líneas de investigación, dando origen a los *Sistemas de Información Dirigidos por*

Ontologías (ODIS, del inglés Ontology-Driven Information Systems). Las bases fundacionales de estos sistemas pueden encontrarse en los primeros trabajos publicados en las áreas de Inteligencia Artificial e Ingeniería de Software, cuyas ideas principales consisten en el modelado del dominio de una aplicación, el razonamiento automatizado, la programación automatizada y las especificaciones ejecutables, el desarrollo de software dirigido por modelos, y las tecnologías semánticas.

El *modelado del dominio* consiste en la definición de modelos explícitos del dominio para el cual una aplicación fue concebida (Iscoe, 1991)(Keller y otros, 1994)(Panti y otros, 1995), constituyendo uno de los primeros intentos de vincular modelos explícitos de conocimiento con sistemas de software.

El *razonamiento automatizado* (Wos y otros, 1984)(Robinson y Voronkov, 2001) puede utilizarse para soportar la mayor parte del ciclo de vida de la ingeniería de software: desde la elicitación y análisis de requerimientos (Alrajeh y otros, 2009)(Meth y otros, 2013), el chequeo de consistencia de los diseños (Farré y otros, 2013)(Nicholson y otros, 2014) e incluso para probar la correctitud del código de programa (Catelani y otros, 2011)(Anand y otros, 2013).

La *programación automatizada* de las aplicaciones de software ha adoptado distintos enfoques. Mientras la evolución de la programación desde el código de máquina hasta los actuales lenguajes de alto nivel constituye un avance en este sentido; la mayor parte de las propuestas se basan en la creación y reuso de modelos formales en etapas tempranas del proceso de desarrollo de software, donde el código ejecutable es obtenido luego de una serie de refinamientos y transformaciones semi-automatizadas (Romli y otros, 2010)(Reiss, 2012)(Arcuri y Yao, 2014).

Las *especificaciones ejecutables* permiten generar programas ejecutables en la forma de modelos de dominio o reglas expresadas en lenguaje natural. Si bien se encuentran estrechamente relacionadas a la propuesta anterior, en este caso la especificación es ejecutada directamente resultando innecesario el paso intermedio de generación de código (Fuchs, 1992)(Farahbod y otros, 2014).

El *Desarrollo Dirigido por Modelos (MDD, del inglés Model-Driven Development)* consiste en la utilización de modelos para la generación de software. La independencia del dominio de estos modelos diferencia este enfoque de las especificaciones ejecutables antes mencionadas. Su utilización incrementa el

nivel de abstracción del proceso en general y simplifica las actividades de desarrollo. Sin embargo, la *Arquitectura Dirigida por Modelos (MDA, del inglés Model-Driven Architecture)* es un enfoque más conocido y utilizado (OMG, 2014b). MDA define un *Modelo Independiente de la Plataforma (PIM, del inglés Platform-Independent Model)* que especifica la funcionalidad requerida de la aplicación. Luego, el PIM se transforma en uno o más *Modelos Específicos de la Plataforma (PSM, del inglés Platform-Specific Model)*. Para la especificación de los modelos generalmente se utiliza el *Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Language)*, lo cual permite su transformación parcial en código ejecutable (OMG, 2011). Aunque esta transformación directa implica una conexión explícita entre los modelos y el software resultante, la mayor parte de la lógica del negocio es implementada manualmente por los desarrolladores y se encuentra desvinculada de los modelos originales (Hailpern y Tarr, 2006)(Whittle y otros, 2013).

Por otra parte, los investigadores y desarrolladores interesados en las tecnologías semánticas focalizaron su atención en el desarrollo de ontologías como núcleo fundamental de la Web Semántica. Aunque esa tendencia surgió en forma independiente y casi sin puntos de contacto con el área de Ingeniería de Software, los beneficios obtenidos en la interacción de ambos campos ha resultado en la definición del *Metamodelo de Definición de Ontología (ODM, del inglés Ontology Definition Metamodel)* (OMG, 2009). ODM es la evolución natural del desarrollo de software dirigido por modelos y representa la base formal para representar, interoperar y automatizar la semántica del negocio por medio de una ontología formalizada. A continuación se describen los beneficios obtenidos de la aplicación de las tecnologías semánticas en el desarrollo de sistemas de software. Una presentación más exhaustiva puede encontrarse en Uschold (2008).

- *Reducción de la brecha conceptual*, dado que los desarrolladores interactúan con las herramientas en una forma que resulta más natural a su modo de concebir el mundo.

- *Reutilización* de nociones abstractas en la instanciación de nociones concretas.

- *Automatización* de las actividades, producto del uso de estructuras formales capaces de ser sometidas a procedimientos de razonamiento automatizado.

- *Reducción en los tiempos de desarrollo* gracias a la combinación de las ventajas mencionadas en los ítems anteriores.

- *Incremento en la confiabilidad*, dado que los constructores formales y la automatización de los procesos reduce la probabilidad de errores humanos.

- *Reducción de los costos de mantenimiento.* La transformación automática de modelos en código ejecutable reduce la probabilidad de aparición de errores. En consecuencia, la existencia de un vínculo formal entre los modelos y el código genera aplicaciones de software más simples de comprender y mantener.

- *Incremento en la flexibilidad del sistema resultante*, dado que el proceso de introducción de modificaciones en un modelo resulta más simple y confiable que la modificación de código de programa.

El reconocimiento de estos beneficios ha incrementado la magnitud de los avances realizados en el área. Sin embargo, los ODIS se encuentran dando los primeros pasos en el proceso de ser una realidad práctica, y probablemente reste cierto tiempo para que se conviertan en un enfoque ampliamente reconocido, aceptado e implementado. Los aportes presentados en esta tesis constituyen un avance importante en tal sentido.

1.2. Descripción del problema

Esta sección expone las razones que motivaron el desarrollo de la tesis, presentando el estado del arte en (1) el campo de las metodologías para el desarrollo de ontologías (Sección 1.2.1), y (2) las áreas relacionadas a la expresión del conocimiento del dominio mediante constructores ontológicos (Sección 1.2.2).

1.2.1. Estado del arte: metodologías para el desarrollo de ontologías

El conocimiento del dominio en el cual una organización opera generalmente se expresa mediante *reglas de negocio*, las cuales permiten caracterizar las entidades del mundo real describiendo políticas, normas, operaciones, definiciones y restricciones de la organización. La gestión eficiente de tales reglas resulta de vital importancia para alcanzar los objetivos estratégicos de la organización (Zacharias, 2008).

La naturaleza dinámica del contexto actual en el cual operan las organizaciones resulta en modificaciones continuas en sus reglas de negocio. Tales reglas usualmente se encuentran embebidas en el código de programa de las diferentes aplicaciones de software que conforman el sistema de información de la organización. En consecuencia, la gestión de las reglas de negocio implica la ejecución de tareas de mantenimiento de software que involucran la búsqueda y modificación de las porciones de código de programa que resultan relevantes a cada uno de los cambios producidos en dichas reglas. Los contextos de negocio rápidamente cambiantes que enfrentan las organizaciones actuales impone la necesidad de contar con tecnologías capaces de soportar una rápida propagación de las nuevas reglas de negocio en las aplicaciones de software pre-existentes. Esto permitiría mejorar la flexibilidad, extensibilidad y facilidad de mantenimiento de los sistemas, incrementando la adaptabilidad de las organizaciones a entornos altamente dinámicos.

Encapsular las reglas de negocio es una manera posible de afrontar la cuestión antes mencionada. Aunque las bases de reglas pueden ser utilizadas a tal fin, (1) el uso de diferentes atributos para representar la misma entidad y (2) la definición de reglas basadas en nociones incompatibles de un mismo concepto presentan complicaciones que terminan afectando la interacción entre las reglas definidas (Zacharias, 2008).

Las ontologías podrían ser utilizadas para tratar ambas cuestiones. Como artefactos destinados a encapsular las reglas de negocio permitirían mejorar la eficiencia en la gestión de las modificaciones producidas. Como artefacto concebido para representar explícitamente la semántica de los datos permitiría evitar la ambigüedad en la definición de los conceptos (Vieira y otros,

2004)(Ruotsalo, 2010). Sin embargo, el proceso de desarrollo de una ontología con las características antes mencionadas debe ser considerado en el contexto de un proceso de desarrollo de software, a fin de posibilitar las actividades de evaluación e integración de la ontología y el código de programa.

Las metodologías para el desarrollo de ontologías aún constituyen un campo abierto de investigación. La mayoría de las metodologías no proveen suficientes detalles sobre las técnicas y actividades involucradas en su ejecución, ni proveen recomendaciones detalladas acerca de la efectiva aplicación de los conceptos de reusabilidad y reingeniería, y en general optan por estrategias convencionales para la identificación de los conceptos ontológicos (Iqbal y otros, 2013).

Las metodologías existentes para el desarrollo de ontologías pueden agruparse en dos grandes categorías. La primera involucra las mejores prácticas del campo de la Ingeniería del Conocimiento (Gómez-Pérez y otros, 2004). Tales prácticas generalmente no forman parte del conjunto de técnicas involucradas en el desarrollo de las aplicaciones de software y su ejecución por parte de los desarrolladores implica un proceso de aprendizaje adicional.

A fin de evitar este inconveniente, un segundo grupo de propuestas metodológicas se basa en estándares ampliamente reconocidos en el campo de la Ingeniería de Software. En general, estas metodologías utilizan UML (OMG, 2011) y el *Lenguaje de Restricción de Objetos (OCL, del inglés Object Constraint Language)* (OMG, 2012) para el modelado conceptual de la ontología (Wang y Chan, 2001)(Guizzardi y otros, 2002)(de Nicola y otros, 2009). La amplia aceptación de UML en la comunidad de ingeniería de software, su representación gráfica estandarizada, la gran disponibilidad de herramientas que lo soportan, y la naturaleza extensible del lenguaje conforman las principales ventajas para ser considerado en un proceso de desarrollo de ontologías. Sin embargo, la falta de una semántica precisa de conjuntos y la ausencia de una teoría de modelos impide la ejecución de razonadores automatizados sobre modelos UML. Tampoco OCL posee una teoría formal de modelos ni una teoría formal de prueba de modelos, por lo que no resulta un candidato posible para la ejecución de procesos automatizados de razonamiento (OMG, 2009). ODM surge con la finalidad de superar estas deficiencias (OMG, 2009)(Saripalle y Blechner., 2014). ODM es una familia de meta-modelos *MOF (del inglés Meta-Object Facility)* (OMG, 2014a), mapeos entre dichos meta-modelos, y mapeos desde y hacia el

lenguaje UML, además de un conjunto de perfiles que permiten el modelado de ontologías mediante herramientas basadas en UML. Los meta-modelos reflejan la sintaxis abstracta de varios lenguajes estándares utilizados en el modelado conceptual y la representación de conocimiento. Sin embargo, lo que hace a un buen modelado de software orientado a objetos no necesariamente hace a una buena ontología: una vez que un modelo particular ha sido transformado en una ontología, es necesario tomar precauciones que permitan asegurar que el modelo resultante soportará las aserciones requeridas. A menudo son necesarias re-estructuraciones significativas de la ontología para satisfacer tal punto. ODM provee dos maneras de relacionar los modelos a fin de superar el problema: (1) mediante perfiles UML y (2) mediante mapeos entre los modelos. Aunque los perfiles proporcionan ciertas facilidades para que los usuarios puedan utilizar UML como base para el desarrollo de ontologías, no facilitan una transformación completa a través del conjunto de paradigmas de representación incluidos en los meta-modelos ODM. Para ello se han definido los mapeos de un meta-modelo a otro. Sin embargo, la naturaleza altamente estructurada de estas propuestas y la ejecución de actividades destinadas a la generación de múltiples artefactos intermedios implica que la dinámica conceptual de los dominios actuales permanezca como un problema de difícil tratamiento (Hepp, 2007). En respuesta a este inconveniente, un grupo posterior de trabajos propone la aplicación de los principios de la *Programación Extrema (XP, del inglés Extreme Programming)* (Beck, 2000) - una metodología ágil para el desarrollo de software ampliamente conocida -, resaltando la importancia de aplicar los cambios requeridos en el momento que sea necesario (Hristozova y Sterling, 2002)(Knublauch, 2002)(Auer y Herre, 2007)(Sharifloo y Shamsfard, 2008). La propuesta más interesante en este sentido es *eXtreme Design (XD)*, un método colaborativo, iterativo e incremental para el diseño de ontologías por medio de patrones (Presutti y otros, 2009). La propuesta se inspira en los principios de la metodología ágil XP para gestionar el proceso, utilizando los patrones de contenido - un tipo particular de patrón de diseño de ontología - para identificar la "mejor" solución de modelado. Sin embargo, la utilización de este tipo de patrones no provee un medio de identificar los conceptos relevantes del dominio ni un procedimiento sistemático para transformar el conocimiento del negocio en constructores ontológicos. Por otro lado, si bien estas propuestas permiten abordar la cuestión de la dinámica

conceptual, la falta de un modelo de fases dificulta la evolución desde un modelo conceptual hasta una ontología final implementada en un lenguaje computable (Reynares y otros, 2012).

En forma opuesta a la notación gráfica de las propuestas basadas en UML, trabajos recientes han remarcado que la definición ontológica de conceptos en un lenguaje formal es muy similar a la descripción de dichos conceptos por medio del lenguaje natural: en ambos casos una expresión es construida mediante la combinación de símbolos de acuerdo a reglas gramaticales (Pinker, 2007)(Hoekstra, 2009).

Siguiendo este enfoque lingüístico, en Reynares y otros (2012) se presenta *EDON (del inglés Evolutionary Development of Ontologies)*, un método para el desarrollo de ontologías concebidas para encapsular la especificación declarativa de los conceptos y las reglas de negocio de una organización. EDON requiere del trabajo colaborativo de expertos del dominio - quienes interactúan diariamente con el dominio del problema y poseen el conocimiento a ser modelado - e ingenieros de conocimiento - quienes poseen el conocimiento técnico y metodológico para representar la realidad -. Luego, las actividades de modelado y formalización de la ontología son realizadas en un alto nivel de abstracción a fin de posibilitar una adecuada comunicación entre ambos perfiles. Para ello se utiliza el *Léxico Extendido del Lenguaje (LEL)* (Sampaio do Prado Leite y Franco, 1993) y un procedimiento heurístico para la transformación en constructores ontológicos del léxico originalmente definido (Breitman y do Prado Leite, 2004). En Ayub y otros (2013) y Ayub y otros (2014) pueden encontrarse reportes de experiencia en la utilización de EDON para el desarrollo de un *Sistema Recomendador de Equipos de Trabajo (TRS, del inglés Team Recommender System)* (Keim, 2007). Tales reportes han permitido reconocer que (1) es necesario un formalismo de poder expresivo superior a LEL para la especificación de reglas de negocio complejas, y (2) la heurística utilizada en la generación de la ontología es incompleta.

1.2.2. Estado del arte: mapeo de reglas de negocio en constructores ontológicos

Varios autores han propuesto el mapeo de reglas de negocio a constructores ontológicos como una técnica para el desarrollo de ontologías (Ceravolo y otros, 2007)(Alberts y Franconi, 2012)(Franconi y Mosca, 2012)(Reynares y otros, 2013)(Kendall y Linehan, 2013)(Reynares y otros, 2014c)(Karpovic y otros, 2014). El enfoque lingüístico adoptado permite la expresión del conocimiento del negocio mediante sentencias en lugar de diagramas, respondiendo a la idea de que los diagramas resultan útiles para la descripción estructural de los conceptos de la organización pero no resultan prácticos como medio principal para la definición de un vocabulario y la expresión de reglas de negocio.

Los trabajos antes mencionados se encuentran basados en dos lenguajes. El primero, denominado *Semántica de Vocabulario de Negocio y Reglas de Negocio (SBVR, del inglés Semantics of Business Vocabulary and Business Rules)*, brinda a los expertos del negocio un medio lingüístico para describir semánticamente los conceptos del dominio y especificar las reglas de negocio en forma independiente del diseño de un sistema de información (OMG, 2013). SBVR se basa en lógica de predicados de primer orden con algunas restricciones en lógicas de orden superior y ciertas extensiones en lógica modal. Esta sólida fundamentación en lógicas formales resulta una característica clave en contextos donde el razonamiento automatizado resulta necesario, y constituye una clara ventaja con respecto al uso de modelos UML/OCL. El segundo de los lenguajes, denominado *Lenguaje de Ontologías Web (OWL, del inglés Ontology Web Language)*, constituye el lenguaje objetivo de las transformaciones debido a su evolución como estándar de facto para la implementación de ontologías destinadas a ser utilizadas en un amplio espectro de aplicaciones (W3C, 2009a). A continuación se describen brevemente los aportes realizados por cada una de las propuestas mencionadas.

En Ceravolo y otros (2007) se explora la combinación de la primera versión de OWL (W3C, 2004a) con expresiones *SWRL (del inglés Semantic Web Rule Language)*. El trabajo adopta los principios de MDA (OMG, 2014b) para obtener un modelo PIM (OMG, 2014b) a partir de vocabularios y reglas de negocios expresadas mediante SBVR. El objetivo es alcanzado mediante un proceso de transformación intermedia entre SBVR y OWL/SWRL, a fin de

realizar chequeos de consistencia y expandir el conocimiento por medio de procedimientos de inferencia. Sin embargo, los autores exploran la factibilidad de los mapeos propuestos entre las versiones iniciales de SBVR y OWL ilustrándolos mediante un ejemplo. En consecuencia, las transformaciones no se encuentran explícitamente formalizadas y no pueden ser generalizadas a otras situaciones. Alberts y Franconi (2012) y Franconi y Mosca (2012) proponen un conjunto de transformaciones SBVR a la segunda versión de OWL (OWL 2) haciendo uso del lenguaje de modelado conceptual denominado *Modelado Objeto-Rol (ORM, del inglés Object Role Modeling)*[1] , el cual constituye el núcleo fundacional de SBVR. Alberts y Franconi (2012) propone la definición y aplicación de un método integrado que utiliza ORM para generar un mecanismo de consulta basado en ontologías. Franconi y Mosca (2012) introduce una semántica de conjuntos para ORM y los mapeos formalmente definidos entre dicho lenguaje y el fragmento *ALCQI* de la *Lógica Descriptiva (DL, del inglés Description Logics)*(Baader y otros, 2003). Este último trabajo también describe una herramienta que implementa las transformaciones de un conjunto de restricciones ORM 2 en una ontología OWL 2. Estos trabajos siguen un enfoque formal al establecer los fundamentos lógicos de las transformaciones entre las teorías subyacentes de SBVR y OWL 2, presentando mapeos de ORM a *Lógica de Primer Orden (FOL, del inglés First Order Logic)*. Pero SBVR no es completamente equivalente a ORM y OWL 2 se encuentra basado en un fragmento de FOL. En consecuencia, las transformaciones de ORM a FOL no pueden ser utilizadas directamente para el mapeo de expresiones SBVR en sentencias OWL 2. Kendall y Linehan (2013) proponen un mapeo reversible entre SBVR y OWL 2 con el objetivo de proveer una forma de intercambiar vocabularios SBVR entre diferentes herramientas informáticas sin pérdida de información semántica. Sin embargo, la propuesta considera un subconjunto muy reducido de los constructores generalmente involucrados en la expresión de reglas de negocio complejas. En Karpovic y otros (2014) se presenta un estudio que analiza la capacidad de un subconjunto del meta-modelo SBVR para la representación de ontologías OWL 2. El análisis sigue la dirección opuesta (de OWL a SBVR) en las transformaciones propuestas por esta tesis, por lo que las características específicas del meta-modelo origen son pasadas por alto. En primer lugar, el estudio no toma en cuenta

[1] http://orm.net/)

que la transformación de diversas expresiones SBVR en sentencias OWL 2 debe ser hecha en función del tipo de las entidades involucradas. Además, tampoco define una manera de llenar la brecha semántica entre SBVR y OWL 2. En Reynares y otros (2013) y Reynares y otros (2014c) se presenta un conjunto de transformaciones basadas en aspectos estructurales de ambos lenguajes, lo cual permite la generación automatizable de una ontología OWL 2 desde las especificaciones SBVR del dominio del negocio. Las transformaciones están basadas en la especificación estructural de ambos estándares en lugar de consideraciones teóricas sobre los lenguajes, proveyendo un conjunto de mapeos utilizables por los expertos del negocio o desarrolladores de software. Reynares y otros (2014a) y Reynares y otros (2014b) describen dos experimentos destinados a obtener evidencia empírica sobre la factibilidad de esta última propuesta. La descripción completa de ambos experimentos puede encontrarse en el Apéndice B y el Apéndice C, respectivamente. En Gaspoz y otros (2013) se presenta un reporte de experiencia en el desarrollo de un sistema de información utilizando una combinación *ad-hoc* del método EDON (Reynares y otros, 2012) y los mapeos de SBVR a OWL 2 presentados en Reynares y otros (2013) y Reynares y otros (2014c), subrayando el potencial de esta combinación como técnica de desarrollo de ontologías. La publicación reciente de una versión mejorada de SBVR y la evidencia empírica recogida por los experimentos antes mencionados ha motivado el desarrollo del trabajo presentado en esta tesis y descripto brevemente en la siguiente sección.

1.3. Objetivo y Aportes

El objetivo de esta tesis consiste en proveer un entorno completo para el desarrollo integrado de sistemas de software basados en ontologías, donde las ontologías encapsulan la especificación declarativa de la lógica del negocio, facilitan la representación del conocimiento sin ambigüedades, brindan servicios de razonamiento al sistema de software, y mejoran la gestión de entornos de alto dinamismo conceptual. Los aportes que a continuación se describen permiten alcanzar este objetivo, constituyendo un paso adelante en el proceso de convertir tales sistemas en una realidad práctica.

En primer lugar, esta tesis presenta un conjunto de patrones de diseño para

la obtención de ontologías OWL/SWRL mediante la aplicación de reglas de transformación de meta-modelos sobre especificaciones SBVR de dominios de negocios. Las transformaciones se encuentran basadas en la especificación estructural de los meta-modelos en lugar de consideraciones teóricas de los lenguajes, proveyendo un conjunto de mapeos fácilmente utilizable por expertos del negocio y/o desarrolladores de software. Luego, se define un marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías. Las ontologías encapsulan el conocimiento del dominio y son utilizadas en tiempo de ejecución por el sistema de software. El marco de trabajo se denomina ***PATRON*** y se encuentra basado en la utilización de los patrones de diseño de ontologías antes definidos.

1.4. Organización de la tesis

El resto de esta tesis está organizado de la siguiente manera. El Capítulo 2 describe un conjunto de patrones de diseño para la obtención de ontologías OWL/SWRL desde la especificación SBVR de un dominio de negocio. El Capítulo 3 presenta ***PATRON***, un marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías. El Capítulo 4 ilustra ***PATRON*** mediante la presentación de los artefactos resultantes de su aplicación en la resolución de un caso de estudio. Las conclusiones y trabajos futuros se presentan en el Capítulo 5.

Capítulo **2**

Patrones de diseño para la reingeniería de sentencias SBVR en ontologías OWL/SWRL

Este capítulo describe un conjunto de patrones de diseño de ontologías para la obtención de una ontología OWL/SWRL mediante la aplicación de reglas de transformación de meta-modelos sobre la especificación SBVR del dominio del negocio. Los patrones se basan en la especificación estructural de los metamodelos considerados, proveyendo un conjunto de mapeos fácilmente utilizables por expertos del negocio o desarrolladores de una aplicación de software.
La Sección 2.1 presenta las nociones conceptuales fundamentales del enfoque de diseño basado en la utilización de patrones, y su utilización en el contexto de la ingeniería ontológica. Descripciones generales de las especificaciones más recientes de los lenguajes SBVR, OWL y SWRL se presentan en las secciones 2.2, 2.3 y 2.4, respectivamente. Los patrones propuestos se describen en la Sección 2.5, mientras que ciertas consideraciones de importancia sobre los mismos se presentan en la Sección 2.6. Finalmente, las conclusiones del capítulo pueden encontrarse en la Sección 2.7.

2.1. Patrones de diseño de ontologías

El término *patrones de diseño* fue concebido originalmente en el campo de la Arquitectura como un conjunto de guías compartidas para la resolución de problemas de diseño. Cada patrón describe un problema recurrente en un contexto dado y presenta la solución a dicho problema (Alexander y otros,

1977)(Alexander, 1979). Actualmente, los patrones de diseño son ampliamente aceptados en la Ingeniería de Software (Gamma y otros, 1994) y su aplicación se ha extendido al campo de la Ingeniería Ontológica (Clark y otros, 2004)(Svatek, 2004)(Hoekstra, 2009)(Gangemi y Presutti, 2009). Luego, la siguiente definición puede plantearse de acuerdo a los trabajos de Gangemi y Presutti (2009) y Blomqvist y otros (2009):

Un ***Patrón de Diseño de Ontología (ODP, del inglés Ontology Design Pattern)*** *es una solución de modelado destinada a resolver un problema recurrente de diseño ontológico, proveyendo (1) beneficios de reuso, (2) beneficios referentes a la guía que proporcionan y (3) beneficios de comunicación de las intenciones de diseño.*

El reuso de ODPs es considerado el beneficio fundamental, refiriéndose a la actividad de utilizar los patrones disponibles en la solución de diferentes problemas de modelado durante el desarrollo de nuevas ontologías (Villazón-Terrazas, 2011). La naturaleza reusable de los patrones facilita el modelado de problemas recurrentes mediante la aplicación de soluciones probadas conocidas en la jerga como "mejores prácticas". Se han realizado diversos experimentos mostrando que el reuso de los patrones facilita el proceso de desarrollo y mejora la calidad de la ontología resultante (Blomqvist y otros, 2009). Gangemi y Presutti (2009) plantea la agrupación de los patrones de acuerdo a las seis familias que se describen a continuación.

1. *ODPs Estructurales.*
   Incluyen a los ODP Lógicos y los ODP Arquitectónicos.
   Los ODP Lógicos son composiciones de constructores lógicos que resuelven un problema de expresividad, ayudando a resolver problemas de diseño donde las primitivas del lenguaje no les brindan un soporte directo. Son independientes de un dominio específico, pero dependen de la expresividad del formalismo lógico que es utilizado para su representación.
   Los ODP Arquitectónicos afectan la "forma" general de la ontología ya sea interna o externamente, debido a decisiones de diseño motivadas por una necesidad específica.

2. *ODPs de Razonamiento.*
   Consisten en la aplicación de los ODP Lógicos a los fines de obtener

determinados resultados del proceso de razonamiento y basándose en el comportamiento implementado en un motor de inferencia: clasificación, subsunción, herencia, etc.

3. *ODPs de Correspondencia.*
   Incluyen los ODPs de Reingeniería y los ODPs de Mapeo.
   Los ODPs de Reingeniería proveen una manera de obtener una ontología mediante la aplicación de un conjunto de reglas de transformación de metamodelos sobre un modelo conceptual. Se pueden distinguir dos tipos de ODPs de Reingeniería. El primer tipo - denominado Patrón de Reingeniería de Esquemas - consiste en reglas para la transformación de un modelo a otro. El segundo tipo - llamado Patrón de Refactorización - provee reglas para cambiar el tipo de los elementos ontológicos.
   Los ODPs de Mapeo son utilizados en la creación de asociaciones semánticas entre dos ontologías preexistentes, proveyendo una forma de relacionarlas sin cambiar los elementos ontológicos involucrados.

4. *ODPs de Presentación.*
   Representan buenas prácticas destinadas a mejorar la usabilidad y legibilidad de las ontologías desde el punto de vista del usuario humano, soportando un mejor reuso al facilitar su evaluación y selección.

5. *ODPs Léxico-Sintácticos.*
   Consisten en estructuras lingüísticas que permiten extraer conclusiones acerca del significado que expresan. Resultan útiles en la asociación de ODP Lógicos y de Contenido con sentencias en lenguaje natural.

6. *ODPs de Contenido.*
   Consisten en patrones conceptuales dependientes del dominio a modelar, proponiendo soluciones a los problemas de diseño referentes a las clases y propiedades que conforman la ontología. Su propósito consiste en actuar como interfaz entre los casos de uso y las soluciones de diseño, abordando un conjunto específico de preguntas de competencia que representan el problema para el cual brindan una solución probada. Los ODPs de Contenido pueden ser vistos como instanciaciones o composiciones de ODPs Lógicos.

Además, los patrones de diseño de ontologías pueden ser clasificados en *lógicos* y *conceptuales*. Mientras los patrones lógicos se encuentran concebidos para la resolución de problemas de diseño independientemente de cualquier conceptualización particular, los patrones conceptuales permiten resolver cuestiones concretas de un dominio específico. La mayor parte del trabajo en este campo ha sido realizado en el grupo *conceptual*, proponiendo un amplio espectro de patrones de contenido para la resolución de cuestiones de modelado en diversos dominios[1] .

En Hoekstra (2009) y Villazón-Terrazas (2011) se presentan importantes contribuciones en el grupo lógico. En el primer trabajo se describen las representaciones OWL 2 de tres patrones de diseño y su aplicación en diversos dominios. En el segundo trabajo se presenta un modelo, un método y una tecnología para el reuso y reingeniería de recursos no ontológicos, para el desarrollo de ontologías mediante la utilización de patrones.

Este capítulo especifica un conjunto de *Patrones de Reingeniería de Esquema (SRP, del inglés Schema Reengineering Pattern)*, los cuales pertenecen a la familia de Correspondencia antes mencionada. Un SRP provee una forma de obtener una ontología mediante la aplicación de un conjunto de reglas de transformación de meta-modelos sobre un modelo conceptual. El modelo conceptual original puede ser una ontología, un tesauro, un patrón de modelo de datos, un modelo UML, una estructura lingüística, etc (Gangemi y Presutti, 2009). Los SRPs presentados en este trabajo definen un conjunto de reglas para la generación de una ontología OWL/SWRL de la especificación SBVR del dominio del negocio. Las siguientes secciones describen los metamodelos involucrados.

2.2. SBVR

El lenguaje denominado *Semántica de Vocabulario de Negocio y Reglas de Negocio* (SBVR 1.1) es la última versión de un lenguaje orientado a los hechos para el modelado del dominio del negocio (OMG, 2013). Por medio de un vocabulario controlado fácilmente comprensible por los expertos del dominio, define el vocabulario y la gramática para la documentación de la semántica de los vocabularios, los hechos, y las reglas del negocio. SBVR posee un

[1] Una lista de ODPs disponibles puede encontrarse en http://ontologydesignpattern.org

sólido fundamento teórico en lógica formal: se encuentra basado en predicados de lógica de primer orden con extensiones en lógica modal. Específicamente, presenta constructores deónticos para la expresión de obligaciones y prohibiciones, junto a contructores aléticos que se utilizan para la expresión de necesidades y posibilidades.

El enfoque orientado a los hechos de SBVR se basa en el *Manifiesto de Reglas de Negocio* [2] , el cual establece que los términos representan conceptos del negocio, los hechos expresan aserciones sobre tales conceptos, y las reglas restringen y soportan tales hechos. De esta manera, el núcleo de SBVR se encuentra compuesto de *conceptos sustantivos* y *conceptos verbales* correspondiendo respectivamente a las nociones de términos y hechos, además de las *reglas* que establecen restricciones.

Un concepto sustantivo representa un sustantivo o una frase sustantiva, y se encuentra especializado en las siguientes categorías:

1. los *conceptos generales* clasifican entidades en base a sus propiedades comunes.

2. los *conceptos individuales* representan objetos individuales.

3. los *roles* representan entidades que asumen una función o son utilizadas en una situación determinada. Además, los *conceptos verbales de rol* se definen como aquellos roles que específicamente caracterizan sus instancias por su utilización en el contexto de un concepto verbal particular.

Un *concepto verbal* representa una frase verbal que involucra uno o más conceptos sustantivos. Puede ser utilizado para modelar relaciones unarias (denominadas *características/conceptos verbales unarios*) o binarias (denominadas *conceptos verbales binarios*). La Figura 2.1 muestra la organización estructural de los conceptos antes mencionados.

Finalmente, una regla SBVR es un elemento de guía que introduce una obligación o necesidad, la cual es construida mediante la imposición de restricciones sobre conceptos verbales por medio de expresiones modales, cuantificadores y operadores lógicos. Cada regla posee un operador modal asociado, el cual puede ser establecido explícita o implícitamente: se asume un operador modal alético

[2] http://www.businessrulesgroup.org/brmanifesto.htm

de necesidad cuando ninguna modalidad ha sido explícitamente especificada. En consecuencia, pueden distinguirse dos categorías fundamentales de reglas:

1. las *reglas estructurales* describen la manera en que el negocio organiza las entidades con las cuales opera.

2. las *reglas operativas* gobiernan las actividades del negocio. A diferencia de la categoría estructural antes mencionada, existe la posibilidad que las reglas operativas sean "rotas" en un determinado "momento" del negocio.

La Tabla 2.1 muestra los operadores modales aléticos y deónticos que pueden ser utilizados en la formulación de una regla SBVR. La Figura 2.2 y la Figura 2.3 presentan los cuantificadores y los operadores lógicos, respectivamente.

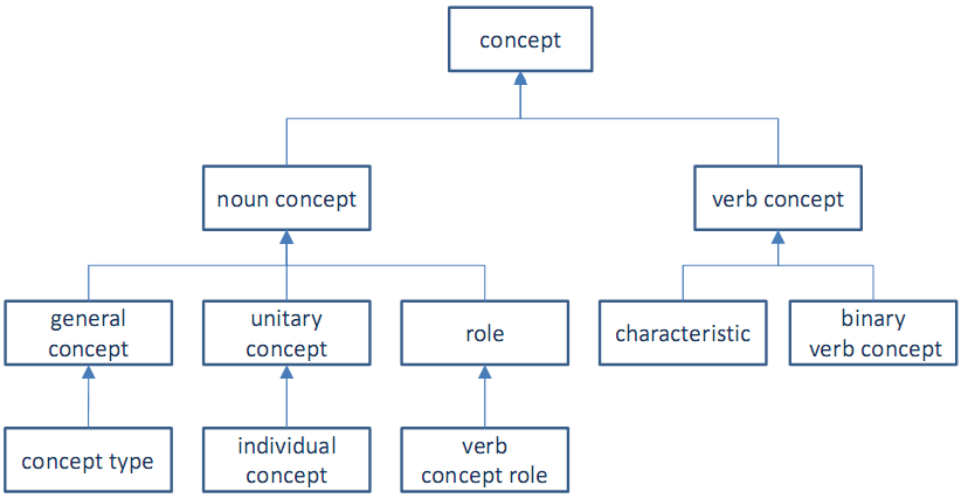

Figura 2.1: Conceptos centrales de SBVR

Modalidades y significados		
aléticos	*necessity*	Es necesario que... p
	non-necessity	No es necesario que... p
	possibility	Es posible que... p
	impossibility	Es imposible que... p
	contingency	Es posible pero no necesario que... p
deónticos	*obligation*	Es obligatorio que... p
	non-obligation	No es obligatorio que... p
	permission	Está permitido que...p
	prohibition	Está prohibido que... p
	optionality	Está permitido pero no es obligatorio que... p

Tabla 2.1: Operadores modales de SBVR, donde p representa una proposición lógica.

2.3. OWL 2

El *Lenguaje de Ontologías Web* (OWL 2) (W3C, 2009a) es la última versión de un lenguaje de implementación ontológica originalmente concebido por el World Wide Web Consortium (W3C), para ser utilizado en el contexto de la Web Semántica (Berners-Lee y otros, 2001). Sin embargo, ha evolucionado como estándar de facto para la implementación de ontologías destinadas a ser utilizadas en un amplio espectro de aplicaciones.
Una ontología OWL 2 constituye una descripción formal de un dominio interpretado bajo una semántica estandarizada, permitiendo ejecutar procedimientos de inferencia sobre el conocimiento explícitamente representado. A continuación se describen brevemente las tres categorías sintácticas en las cuales se basa una ontología OWL 2.

1. Las *entidades* constituyen los elementos básicos de una ontología, entre las cuales se pueden mencionar las *clases*, las *propiedades* y los *individuos*. Por ejemplo, una clase *Persona* representaría el conjunto de individuos de la

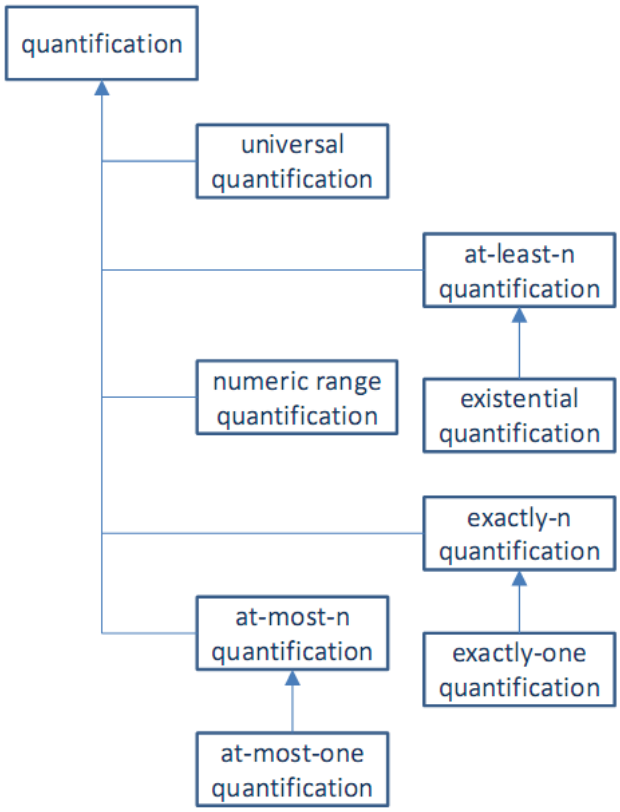

Figura 2.2: Cuantificadores de SBVR

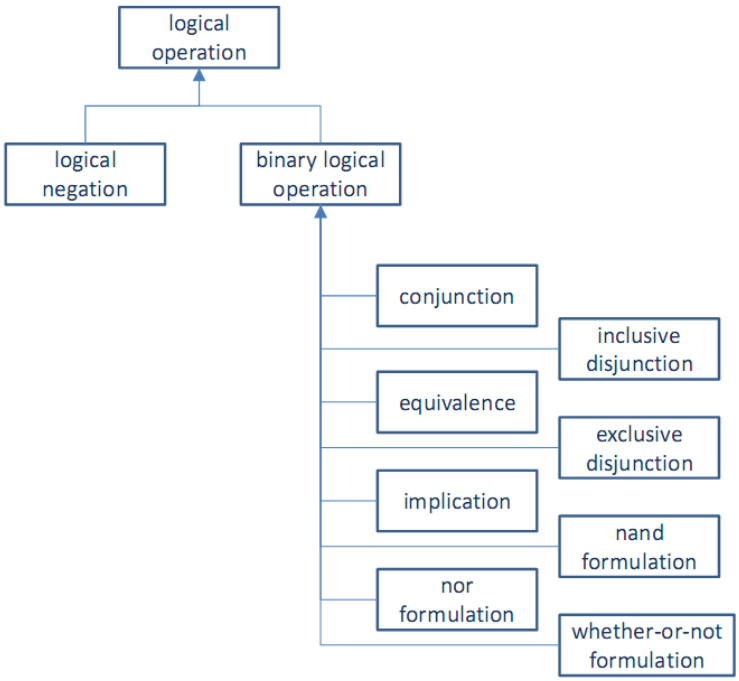

Figura 2.3: Operaciones lógicas de SBVR

raza humana, la propiedad *padre-de* podría representar la relación entre padres e hijos, y el individuo *Juan* modelaría una persona particular con dicho nombre. La Figura 2.4 describe la especificación estructural de las entidades.

2. Las *expresiones* representan nociones complejas del dominio siendo descripto. Por ejemplo, una *expresión de clase* describe un conjunto de individuos en función de las características y restricciones que los representan. La Figura 2.5 describe la especificación estructural de las expresiones.

3. Los *axiomas* son sentencias siempre verdaderas del dominio. Por ejemplo, un axioma de clasificación establece que la clase *Niño* es una especialización de la clase *Persona*. La Figura 2.6 describe la especificación estructural de los axiomas.

OWL 2 define varias sintaxis concretas para serializar e intercambiar las

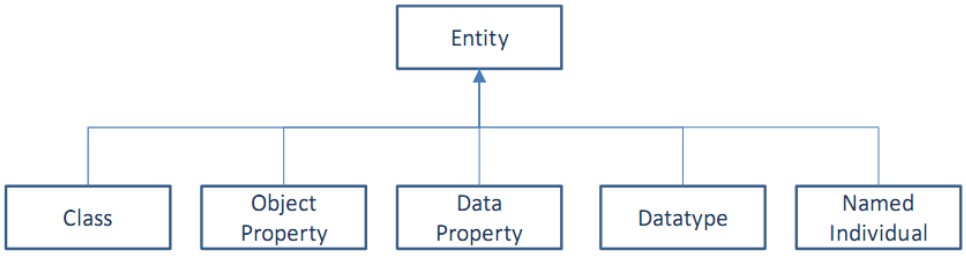

Figura 2.4: Entidades de OWL 2

ontologías implementadas. Entre ellas, la sintaxis de estilo funcional es utilizada en la especificación estructural del lenguaje para establecer la semántica de sus constructores (W3C, 2009b). Respecto a la semántica del lenguaje, OWL 2 define dos alternativas: (1) la Semántica Directa (W3C, 2012a) y (2) la Semántica basada en RDF (W3C, 2012b).

La Semántica Directa es compatible con la semántica del fragmento SROIQ de la Lógica Descriptiva (Gómez-Pérez y otros, 2004). En tal caso, los procedimientos de razonamiento aplicados sobre la ontología resultan *sólidos* - las consultas sólo obtienen respuestas correctas - y *completos* - la totalidad de las respuestas correctas son obtenidas -. “OWL DL” es el término utilizado comúnmente para

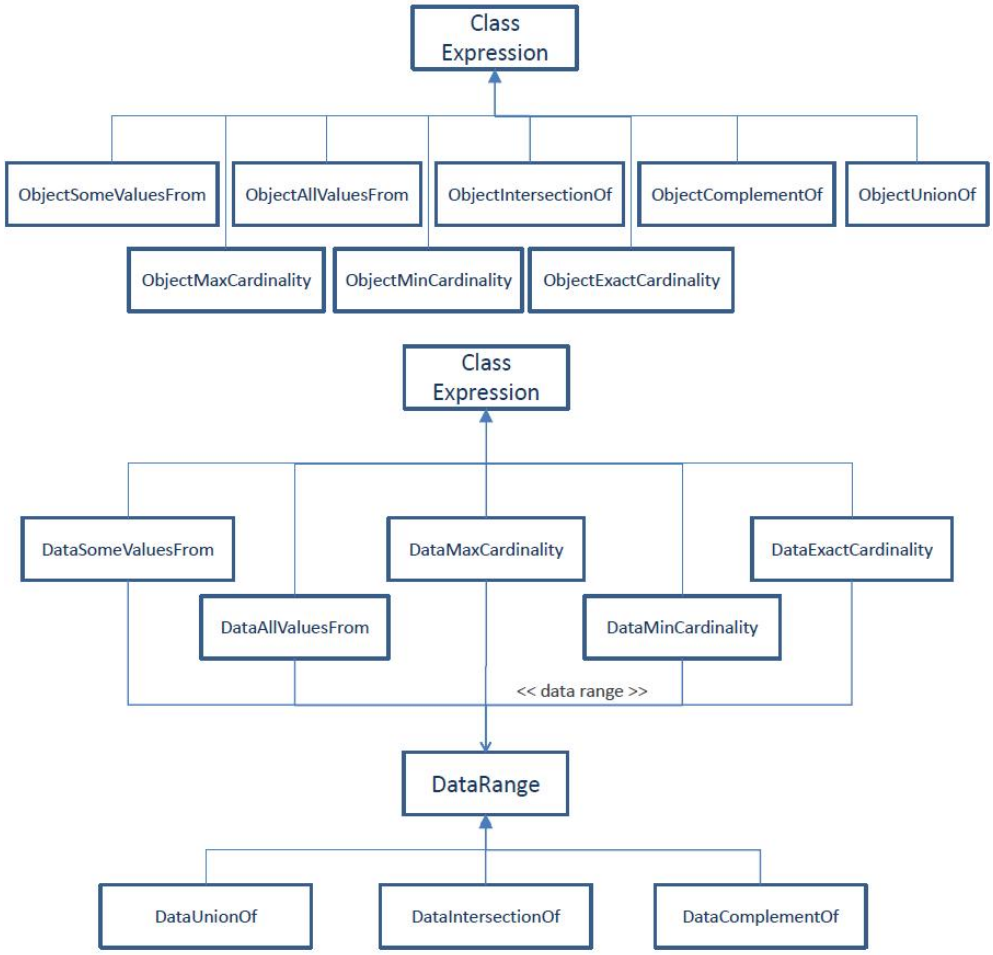

Figura 2.5: Expresiones de OWL 2

referirse a ontologías OWL 2 que satisfacen los condiciones sintácticas y son interpretadas por medio de la Semántica Directa.

La Semántica basada en RDF puede ser aplicada a una ontología OWL 2 sin restricciones. Las ontologías interpretadas de esta forma se denominan comúnmente "OWL Full". Sin embargo, un procedimiento de razonamiento aplicado sobre una ontología interpretada bajo esta semántica podría no ser completo: es decir, no podría asegurarse que sean obtenidas la totalidad de las respuestas correctas a una consulta dada.

El resto de este capítulo sigue la especificación estructural del estándar para la especificación de ontologías OWL 2: las sentencias son definidas mediante el sintaxis de estilo funcional y adquieren significado por medio de la Semántica Directa.

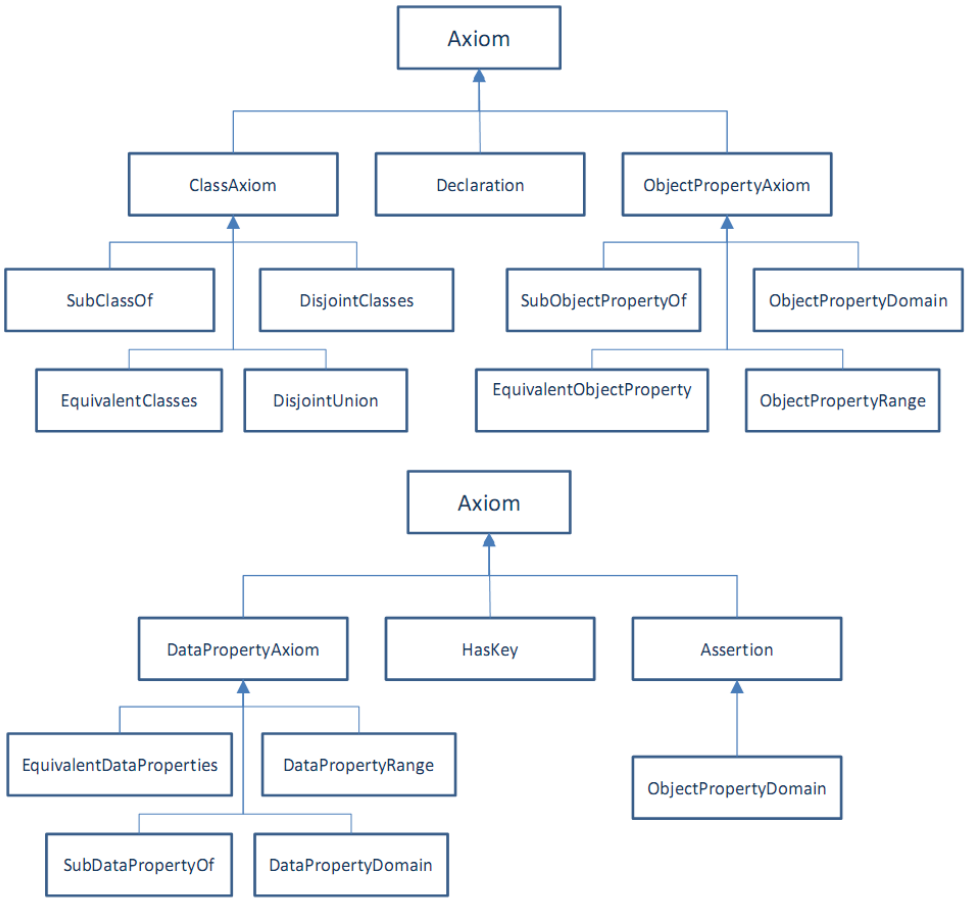

Figura 2.6: Axiomas de OWL 2

2.4. SWRL

El *Lenguaje de Reglas de la Web Semántica* (SWRL) ha sido concebido para extender el conjunto de axiomas OWL 2 a fin de incluir *Clásulas de Horn* (W3C, 2004b), definiendo un modelo teórico que provee una semántica formal para las ontologías que incluyan tales reglas.

Las reglas SWRL adoptan la forma más ampliamente difundida: una implicación entre un antecedente (o cuerpo) y un consecuente (o cabeza). Una expresión como la anterior es interpretada de la siguiente manera: siempre que las condiciones especificadas en el cuerpo de la regla se mantengan, luego las condiciones especificadas en la cabeza deben satisfacerse. Tanto la cabeza como el cuerpo de la regla consisten de cero o más átomos. Un antecedente vacío se considera trivialmente verdadero, luego el consecuente también debe ser verdadero. Un consecuente vacío se considera trivialmente falso, luego el antecedente debe

ser insatisfecho. Múltiples átomos son siempre tratados como una conjunción lógica. Las reglas con conjunciones en sus consecuentes pueden ser fácilmente transformados en múltiples reglas con un consecuente atómico. Los átomos pueden ser de la forma *'C(x)'*, *'P(x,y)'*, *'sameAs(x,y)'* o *'differentFrom(x,y)'* donde:

- *'C'* es una descripción OWL 2
- *'P'* es una propiedad OWL 2
- *'x'* e *'y'* son variables, individuos OWL 2, o valores atómicos
- *'sameAs(x,y)'* y *'differentFrom(x,y)'* representan equivalencia y disyunción lógica, respectivamente.

2.5. Patrones de reingeniería de esquema

Las siguientes secciones describen un conjunto de *patrones de reingeniería de esquemas* para la generación automatizable de una ontología implementada en OWL 2, partiendo de las especificaciones del dominio expresadas mediante sentencias SBVR. Los beneficios brindados por SBVR y OWL 2 han sido mencionados en secciones anteriores, motivando su selección respectiva como lenguajes origen y destino de los patrones propuestos. Cada SRP define una regla de transformación de un modelo origen a un modelo objetivo.
La especificación formal de los patrones propuestos puede encontrarse en el Apéndice A. En las siguientes secciones se los presenta e ilustra mediante un ejemplo, el cual describe la organización estructural de una compañía ficticia (Figura 2.7). Aún cuando el ejemplo es bastante simple, resulta lo suficientemente completo a fin de describir la utilidad de los patrones propuestos. La compañía se encuentra organizada en departamentos y adhiere a una jerarquía de mando tradicional: cada empleado se encuentra a cargo de un único superior y cada superior se encuentra al mando de 25 empleados como máximo. Tanto *empleado* como *superior* son roles jugados por los recursos humanos de la compañía, la cual emplea personal de ambos sexos. El personal posee un identificador único y es descrito mediante la enumeración de sus habilidades y la posible existencia de

un nivel superior de experiencia. Los pasantes constituyen una clase especial de recurso humano.

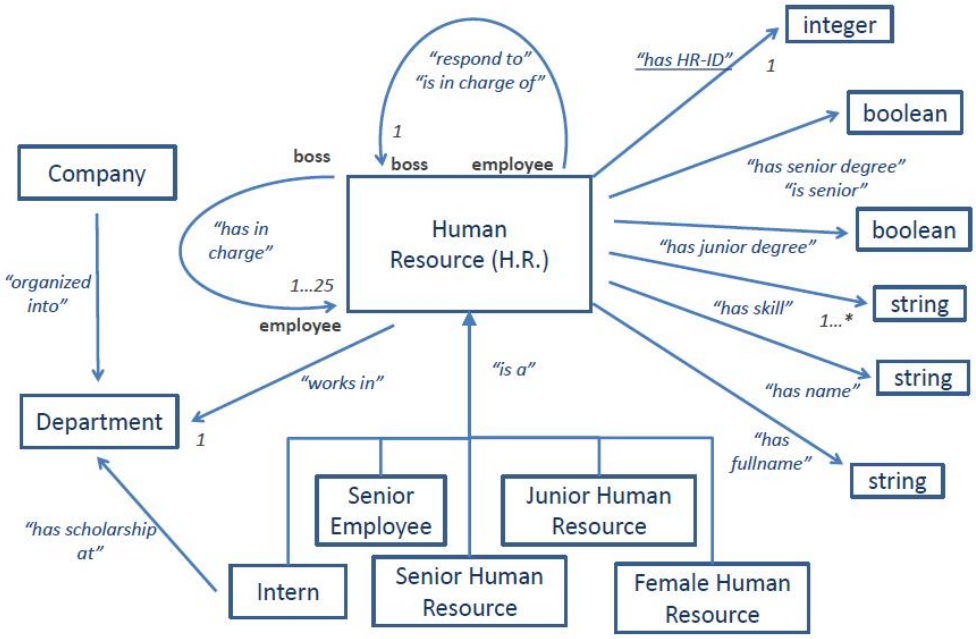

Figura 2.7: Estructura general de la compañía

2.5.1. Patrones fundamentales

Esta sección describe los patrones para la transformación de los conceptos que constituyen el núcleo del metamodelo SBVR, los cuales resultan indispensables para la definición de los restantes patrones.

2.5.1.1. Concepto General (General Concept)

SBVR define *concepto general* como *"concepto sustantivo que clasifica las cosas en función de sus propiedades comunes"*. Luego, un concepto general se transforma en una clase OWL 2, la cual es entendida como un conjunto de individuos.

Siguiendo el ejemplo antes presentado, *'Human Resource'* - el cual representa las personas trabajando en la compañía - se transforma en una clase OWL 2 homónima.

La especificación formal de este patrón puede encontrarse en el Apéndice A.1.

2.5.1.2. Concepto Unitario (Unitary Concept)

SBVR define *concepto unitario* como *"concepto sustantivo que posee una instancia como máximo"*. Aunque la definición corresponde a la noción ampliamente conocida como *singleton* en el campo de la Ingeniería de Software, OWL 2 no define un constructor para modelar tales conceptos. Sin embargo, es posible definir una sentencia SWRL que permita cubrir dicha falta de expresividad:

$$Singleton(?x),\ Singleton(?y) \rightarrow SameAs\ (?x,\ ?y)$$

En consecuencia, un concepto unitario se transforma en una clase OWL 2 junto a la sentencia SWRL antes definida, la cual establece que si más de un individuo es definido para dicha clase, luego los individuos son en realidad los mismos.
En el contexto del dominio modelado en el ejemplo, la compañía modelada es el único individuo del concepto *'Company'*. Luego, la existencia de individuos pertenecientes a la clase OWL 2 *'Company'* es restringida por la sentencia SWRL:

$$Company(?x),\ Company(?y) \rightarrow SameAs\ (?x,\ ?y).$$

La especificación formal de este patrón puede encontrarse en el Apéndice A.2.

2.5.1.3. Rol de Concepto Verbal (Verb Concept Role)

SBVR define *rol* como *"concepto sustantivo determinado de acuerdo a la función que asume o la manera en que es utilizado en una situación dada"*. Luego, SBVR define *rol de concepto verbal* como *"un rol que específicamente caracteriza sus instancias por su participación en una instancia de un concepto verbal determinado"*. Luego, incorpora aquellas características requeridas por el concepto verbal para las instancias del rol.
En consecuencia, un rol de concepto verbal se transforma en una clase OWL 2 definida en términos de (1) la relación que le brinda significado al rol y (2) el concepto general pasible de jugar dicho rol. De esta manera, el rol de concepto verbal es *reificado* como un clase OWL 2 definida en función de dos propiedades de objeto. La primera de ellas representa el concepto verbal en el cual el rol participa. La segunda establece el concepto general que juega el rol definido. Por ejemplo, el concepto verbal *'employee is in charge of boss'* del ejemplo establece la relación entre los roles del empleado y su superior. Ambos roles son jugados

por recursos humanos de la organización.
La especificación formal de este patrón puede encontrarse en el Apéndice A.3.

2.5.1.4. Concepto verbal binario (Binary Verb Concept)

SBVR define *concepto verbal binario* como *"concepto verbal que tiene exactamente dos roles"*. Las expresiones *'employee is in charge of boss'* y *'human resource has skill'* son ejemplos de conceptos verbales binarios. La primera establece la relación entre los roles de empleado y su superior, mientras la segunda representa la relación entre los recursos humanos y sus habilidades. En consecuencia, la transformación de un concepto verbal binario en una sentencia OWL 2 es realizada de acuerdo al tipo de los roles involucrados.
Si el concepto verbal relaciona conceptos, luego la expresión se transforma en una propiedad de objeto OWL 2 con sus correspondientes dominio y rango. Luego, el concepto verbal binario *'employee is in charge of boss'* se transforma en la propiedad de objeto OWL 2 *'is_in_charge_of'*, con las clases OWL 2 *'Employee'* y *'Boss'* como sus dominio y rango respectivos.
Si el concepto verbal relaciona un concepto con un literal, luego la expresión se transforma en una propiedad de datos OWL 2. Luego, el concepto verbal binario *'human resource has skill'* se transforma en la propiedad de datos OWL 2 *'has_skill'*, con la clase OWL 2 *'Human Resource'* como dominio y el tipo de datos *'string'* como rango.
La especificación formal de este patrón puede encontrarse en el Apéndice A.4.

2.5.1.5. Característica/Concepto verbal unario (Characteristic/Unary Verb Concept)

SBVR define *característica* como *"concepto verbal que tiene exactamente un rol, una abstracción de una propiedad de un objeto o de un conjunto de objetos"*. Por ejemplo, la expresión *'human resource has senior degree'* representa el nivel de experiencia de un recurso humano determinado.
Una característica se transforma a una propiedad de datos OWL 2 con el tipo de datos booleano como rango. Luego, la expresión del ejemplo antes mencionado se transforma en la propiedad de datos OWL 2 *'has_senior_degree'*, con la clase OWL 2 *'Human Resource'* como dominio y el tipo de datos booleano como rango.

La especificación formal de este patrón puede encontrarse en el Apéndice A.5.

2.5.1.6. Concepto individual (Individual Concept)

SBVR define *concepto individual* como *"concepto que corresponde a un único objeto"*. La expresión *'John Smith'* es un ejemplo de concepto individual representando un recurso humano con dicho nombre.
Un concepto individual se transforma en un individuo nombrado OWL 2. Por ejemplo, el concepto individual *'John Smith'* se modela como el individuo nombrado OWL 2 *'John Smith'*. Resulta importante destacar que los individuos generados por la aplicación de este patrón no pertenecen a ninguna clase OWL 2 en particular, sino que corresponden a instancias de una clase OWL 2 predefinida denominada *'Thing'*. Esta clase engloba todos los individuos del dominio descripto. La asignación de un individuo a una clase determinada requiere de una sentencia de clasificación SBVR, como se describe en la Sección 2.5.1.7.
La especificación formal de este patrón puede encontrarse en el Apéndice A.6.

2.5.1.7. Clasificación (Classification)

SBVR define *clasificación* como *"preposición que establece que un concepto individual determinado es una instancia de un concepto general dado"*.
Una clasificación SBVR se transforma en un axioma de aserción de clase OWL 2 *'ClassAssertion(C I)'*, el cual establece que el individuo *'I'* pertenece a la clase *'C'*. En consecuencia, la aplicación de este patrón incorpora una mayor semántica al modelado de un individuo determinado.
La especificación formal de este patrón puede encontrarse en el Apéndice A.7.

2.5.1.8. Formulación Atómica (Atomic Formulation)

SBVR define una *formulación atómica* como *"formulación atómica que se encuentra basada en un concepto verbal donde cada uno de sus roles se encuentra instanciado"*.
La expresión *'John Smith is in charge of Ben Arten'* es un ejemplo de una formulación atómica basada en el concepto verbal binario *'employee is in charge of boss'* antes mencionado. La formulación atómica ejemplificada vincula el individuo *'John Smith'* con el rol de *'employee'* mientras *'Ben Arten'* adopta el

rol de *'boss'*. Otro ejemplo lo constituye la expresión *'John Smith has mechanical engineering skills'*, la cual vincula un recurso humano determinado con una habilidad particular. Luego, la transformación de una formulación atómica en una sentencia OWL 2 debe considerar el tipo de los roles involucrados.

Si la formulación atómica se basa en un concepto verbal que vincula conceptos, la expresión SBVR se transforma en una aserción positiva de propiedad de objeto OWL 2 *'ObjectPropertyAssertion(OPE I_1 I_2)'*. Dicha expresión manifiesta que el individuo *'I_1'* se encuentra vinculado por la propiedad de objeto *'OPE'* al individuo *'I_2'*. La expresión *'John Smith is in charge of Ben Arten'* es un ejemplo de este tipo.

Si la formulación atómica se basa en un concepto verbal que vincula un concepto con un literal, la expresión SBVR se transforma en una aserción positiva de propiedad de datos OWL 2 *'DataPropertyAssertion(DPE I DR)'*. Dicha expresión manifiesta que el individuo *'I'* se encuentra vinculado por la propiedad de datos *'DPE'* a un literal del rango de datos *'DR'*. La expresión *'John Smith has mechanical engineering skill'* es un ejemplo de este tipo.

La especificación formal de este patrón puede encontrarse en el Apéndice A.8.

2.5.1.9. Esquema de Referencia (Reference Scheme)

SBVR define *esquema de referencia* como *medio de identificación de los individuos de un concepto dado*. Un esquema de referencia es la manera de referirse a individuos particulares por medio de sus atributos, ya sean literales u objetos. La identificación única de los recursos humanos de la compañía ilustrada constituye un ejemplo de este tipo. Luego, SBVR permite utilizar como esquema de referencia uno o más roles de concepto verbales binarios y/o una o más características.

Un esquema de referencia se transforma en la sentencia OWL 2 *'HasKey(CE (OPE_1 ... OPE_m) (DPE_1 ... DPE_n))'*, la cual establece que cada individuo de la clase *'CE'* es identificada unívocamente por las propiedades de objeto *'OPE_i'* y/o las propiedades de datos *'DPE_j'*. Esto es, no existen dos instancias distintas de *'CE'* que coincidan en los valores de todas las propiedades de objetos *'OPE_i'* y en los valores de todas las propiedades de datos *'DPE_j'*. La clase *'CE'* representa el concepto cuyos individuos se desea identificar, y las propiedades de objeto *'OPE_i'* junto a las propiedades de datos *'DPE_j'* representan los roles de conceptos

verbales binarios y las características definidas para la identificación.
La especificación formal de este patrón puede encontrarse en el Apéndice A.9.

2.5.2. Operaciones Lógicas (Logical Operations)

SBVR define *operación lógica* como *"formulación lógica que expresa un significado basado exclusivamente en la veracidad o falsedad de los significados de uno o más operandos lógicos, donde cada operando constituye una formulación lógica por sí misma"*. La transformación de las operaciones lógicas SBVR - *negación*, *conjunción*, *disyunción*, *equivalencia*, *disyunción exclusiva*, *disyunción negada*, y *conjunción negada* - son descriptas en las siguientes secciones.

2.5.2.1. Negación Lógica (Logical Negation)

SBVR define *negación lógica* como *"operación lógica que tiene exactamente un operando lógico cuyo significado es falso"*. Una negación lógica SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.
Si el operando lógico es un concepto verbal vinculando conceptos, luego la expresión se transforma en una sentencia OWL 2 *'ObjectComplementOf(CE)'*, la cual comprende la totalidad de los individuos que no resultan instancias de la expresión de clase *'CE'*.
Siguiendo el ejemplo presentado, la expresión *'intern does not have a senior degree'* presenta una negación lógica que vincula un concepto general y una característica. La expresión se transforma en la sentencia OWL 2 *'ObjectComplementOf(DataHasValue(:has_senior_degree true))'*. En cambio, la expresión *'intern has not in charge employee'* presenta una negación lógica vinculando conceptos generales mediante un concepto verbal binario. Dicha expresión se transforma en una expresión OWL 2 *'ObjectComplementOf(ObjectSomeValuesFrom(:has_in_charge :Employee))'*.
Si el operando lógico es una formulación atómica vinculando individuos, luego la expresión se transforma en una sentencia OWL 2 *'NegativeObjectPropertyAssertion(OPE I_1 I_2)'*, la cual manifiesta que el individuo *'I_1'* no se encuentra vinculado por la propiedad de

objeto *'OPE'* al individuo *'I_2'*. La expresión *'Peter Tomsom is not in charge of Ben Arten'*, la cual manifiesta que el primer individuo no se encuentra a cargo del segundo - se transforma en la expresión OWL 2 *'NegativeObjectPropertyAssertion(:is_in_charge_of :Peter_Tomsom :Ben_Arten)'*.
Si el operando lógico es una formulación atómica vinculando un individuo con un literal, luego la expresión se transforma en una sentencia OWL 2 *'NegativeDataPropertyAssertion(DPE I DR)'*, la cual establece que el individuo *'I'* no se encuentra vinculado por la propiedad de datos *'DPE'* a un literal del rango de datos *'DR'*. Siguiendo el ejemplo, la sentencia *'Peter Tomsom does not have a senior degree'* se transforma en una expresión OWL 2 *'NegativeDataPropertyAssertion(:has_senior_degree :Peter_Tomsom true)'*.
La especificación formal de este patrón puede encontrarse en el Apéndice A.10.

2.5.2.2. Conjunción (Conjunction)

SBVR define *conjunción* como *"operación lógica binaria donde el significado de cada uno de sus operandos es verdadero"*. Una conjunción SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.
Si la conjunción se aplica sobre conceptos verbales, la expresión se transforma en una sentencia OWL 2 *'ObjectIntersectionOf(CE_1 CE_2)'*, la cual contiene la totalidad de los individuos que pertenecen simultáneamente a las clases *'CE_1'* y *'CE_2'*. La misma transformación se aplica si la conjunción es formulada sobre conceptos generales. La expresión *'has senior degree and is in charge of'* es un ejemplo de conjunción sobre los conceptos verbales *'senior degree'* and *'is in charge of'*.
Si la conjunción se aplica sobre tipos de datos - por ejemplo, números reales, números enteros, cadenas de caracteres, etc. -, la expresión se transforma en una sentencia OWL 2 *'DataIntersectionOf(DR_1 DR_2)'*, la cual contiene la totalidad de los literales que pertenecen simultáneamente a los tipos de datos *'DR_1'* y *'DR_2'*. La expresión *'nonNegativeNumbers and NonPositiveNumbers'* es un ejemplo de conjunción sobre los conjuntos de datos referenciados.
La especificación formal de este patrón puede encontrarse en el Apéndice A.11.

2.5.2.3. Disyunción (Disjunction)

SBVR define *disyunción* como *"operación lógica binaria donde el significado de al menos uno de sus operandos es verdadero"*. Una disyunción SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.

Si la disyunción SBVR se aplica sobre conceptos verbales, la expresión se transforma en una sentencia OWL 2 *'ObjectUnionOf(CE_1 CE_2)'*, la cual contiene la totalidad de los individuos que pertenecen al menos a una de las clases *'CE_i'*.

La misma transformación se aplica si la disyunción es formulada sobre conceptos generales. Las expresiones SBVR *'has senior degree or has skill'* y *'employee or boss'* son ejemplos de disyunciones sobre conceptos verbales y conceptos generales, respectivamente.

Si la disyunción se aplica sobre tipos de datos, la expresión se transforma en una sentencia OWL 2 *'DataUnionOf(DR_1 DR_2)'*, la cual contiene la totalidad de los literales que pertenecen al menos a uno de los tipos de datos *'DR_i'*. La expresión SBVR *'nonNegativeNumbers or NonPositiveNumbers'* es un ejemplo de disyunción sobre los conjuntos de datos referenciados.

La especificación formal de este patrón puede encontrarse en el Apéndice A.12.

2.5.2.4. Equivalencia (Equivalence)

SBVR define *equivalencia* como *"operación lógica binaria donde el significado de sus operandos son todos verdaderos o todos falsos"*. Una equivalencia SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.

Si la equivalencia se aplica sobre conceptos generales, la expresión se transforma en una sentencia OWL 2 *'EquivalentClasses(CE_1 CE_2)'*, la cual establece que las expresiones de clase *'CE_1'* y *'CE_2'* son semánticamente equivalentes. La expresión SBVR *'human resource is equivalent to h.r.'* es un ejemplo de equivalencia entre conceptos generales.

Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos, la expresión se transforma en una sentencia OWL 2 *'EquivalentObjectProperties(OP_1 OP_2)'*, la cual establece que las expresiones de propiedad de objetos *'OP_1'* y *'OP_2'* son semánticamente equivalentes. La expresión SBVR *'respond to is*

equivalent to is in charge of' es un ejemplo de equivalencia entre conceptos verbales que relacionan conceptos.
Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'EquivalentDataProperties(DP_1 DP_2)'*, la cual establece que las expresiones de propiedad de datos *'DP_1'* y *'DP_2'* son semánticamente equivalentes. La expresión SBVR *'has senior degree is equivalent to is senior"* es un ejemplo de equivalencia entre conceptos verbales que relacionan conceptos con literales.
Si la equivalencia se aplica sobre individuos, luego la expresión se transforma en una sentencia OWL 2 *'SameIndividual(I_1 I_2)'*, la cual establece que los individuos *'I_1'* e *'I_2'* son en realidad el mismo. La expresión SBVR *'John Smith is equivalent to J.S.'* es un ejemplo de equivalencia entre individuos.
La especificación formal de este patrón puede encontrarse en el Apéndice A.13.

2.5.2.5. Disyunción Exclusiva (Exclusive Disjunction)

SBVR define *'disyunción exclusiva'* como *"operación lógica binaria donde el significado de uno de sus operandos es verdadero y el otro es falso"*. Una disyunción exclusiva SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.
Si la disyunción exclusiva SBVR se aplica sobre conceptos generales la expresión se transforma en una sentencia OWL 2 *'DisjointClasses(CE_1 CE_2 ... CE_n)'*, la cual establece que las expresiones de clase *'CE_i'* son mutuamente exclusivas, es decir, ningún individuo puede ser al mismo tiempo una instancia de *'CE_i'* y *'CE_j'* para $i \neq n$. La expresión SBVR *'Female Person xor Male Person'* es un ejemplo de disyunción exclusiva entre conceptos generales.
Si la disyunción exclusiva SBVR se aplica sobre conceptos verbales que vinculan conceptos la expresión se transforma en una sentencia OWL 2 *'DisjointObjectProperties(OPE_1 OPE_2 ... OPE_n)'*, la cual establece que las expresiones de propiedad de objeto OPE_i son mutuamente exclusivas, es decir ningún individuo *'I_1'* se encuentra conectado al individuo *'I_2'* por *'OPE_i'* y *'OPE_j'* para $i \neq n$. La expresión SBVR *'is in charge of xor has in charge'* es un ejemplo de disyunción exclusiva entre conceptos verbales vinculando conceptos.
Si la disyunción exclusiva SBVR se aplica sobre conceptos verbales vinculando conceptos con literales la expresión se transforma en una sentencia OWL 2

DisjointDataProperties(DPE_1 DPE_2 ... DPE_n), la cual establece que las expresiones de propiedad de datos *'DPE_i'* son mutuamente exclusivas, es decir ningún individuo *'I_1'* se encuentra conectado al literal *'lt'* por *'DPE_i'* y *'DPE_j'* para $i \neq n$. La expresión SBVR *'has senior degree xor has junior degree'* es un ejemplo de disyunción exclusiva entre conceptos verbales vinculando conceptos con literales.
La especificación formal de este patrón puede encontrarse en el Apéndice A.14.

2.5.2.6. Conjunción Negada (Nand)

SBVR define *'conjunción negada'* como *"operación lógica binaria donde el significado de al menos uno de sus operandos es falso"*. Una conjunción negada SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.
Si la conjunción negada se aplica sobre conceptos verbales la expresión se transforma en una sentencia OWL 2 *'ObjectComplementOf(ObjectIntersectionOf(CE_1 CE_2 ... CE_n)'*, la cual comprende la totalidad de los individuos que no resultan instancias simultáneas de las clases *'CE_i'* para todo $i \neq n$. La expresión SBVR *'has senior degree nand has in charge of Employee'* es un ejemplo de conjunción negada entre conceptos verbales. Se aplica la misma transformación si la conjunción se formula sobre conceptos generales. La expresión SBVR *'Senior Employee nand Boss'* es un ejemplo de conjunción negada entre conceptos generales.
Si la conjunción negada se aplica sobre tipos de datos la expresión se transforma en una sentencia OWL 2 *'DataComplementOf(DataIntersectionOf(DR_1 DR_2 ... DR_n)'*, la cual comprende la totalidad de los literales que no forman parte simultáneamente de los rangos de datos *'DR_i'* para todo $i \neq n$. La expresión SBVR *'nonNegativeInteger nand nonPositiveInteger'* es un ejemplo de conjunción negada entre tipos de datos.
La especificación formal de este patrón puede encontrarse en el Apéndice A.15.

2.5.2.7. Disyunción Negada (Nor)

SBVR define *'disyunción negada'* como *"operación lógica binaria donde el significado de todos sus operandos es falso"*. Una disyunción negada SBVR

se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados.
Si la disyunción negada SBVR se aplica sobre conceptos verbales la expresión se transforma en una sentencia OWL 2 *'ObjectComplementOf(ObjectUnionOf(CE_1 CE_2 ... CE_n ')*, la cual comprende la totalidad de los individuos que no resultan instancias de alguna de las clases *'CE_i'* para todo $i \neq n$. La expresión SBVR *'has senior degree nor has mechanical engineering skill'* es un ejemplo de disyunción negada entre conceptos verbales. Se aplica la misma transformación si la disyunción negada se formula sobre conceptos generales. La expresión SBVR *'Intern nor Employee'* es un ejemplo de disyunción negada entre conceptos generales.
Si la disyunción negada se aplica sobre tipos de datos la expresión se transforma en una sentencia OWL 2 *'DataComplementOf(DataUnionOf(CE_1 CE_2 ... CE_n)'*, la cual comprende la totalidad de los literales que no forman parte de alguno de los rangos de datos *'DR_i'* para todo $i \neq n$. La expresión SBVR *'nonNegativeInteger nor nonPositiveInteger'* es un ejemplo de disyunción negada entre tipos de datos.
La especificación formal de este patrón puede encontrarse en el Apéndice A.16.

2.5.3. Cuantificaciones (Quantifications)

SBVR define *cuantificaciones* como *"formulaciones lógicas que introducen una variable con alguno de los siguientes significados: todos los referentes de la variable satisfacen una formulación de alcance o un número delimitado de referentes de la variable existen y satisfacen una formulación de alcance - si es que existe alguna -"*.
La transformación de las cuantificaciones SBVR - *'universal'*, *'existencial'*, *'como máximo n'*, *'al menos n'*, *'exactamente n'*, *'como máximo uno'*, *'exactamente uno'*, y *'rango numérico'* - son descriptas en las siguientes secciones.

2.5.3.1. Cuantificación Universal (Universal Quantification)

SBVR define *cuantificación universal* como *"cuantificación cuyo ámbito es una formulación lógica, con el siguiente significado: para cada referente de la variable introducida por la cuantificación, el significado de la formulación lógica para el referente es verdadero"*. Una cuantificación universal SBVR se transforma

en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos, la expresión se transforma en una sentencia OWL 2 *'ObjectAllValuesFrom(OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran vinculados por la propiedad de objeto *'OPE' sólo* a individuos de la clase *'CE'*. La expresión SBVR *'works in a given department'* es un ejemplo de cuantificación universal del tipo antes mencionado.
Si la cuantificación se aplica sobre un concepto verbal vinculando conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'DataAllValuesFrom(DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE' sólo* a literales en el rango de datos *'DR'*. La expresión SBVR *'has a given skill'* es un ejemplo de cuantificación universal del tipo descripto.
La especificación formal de este patrón puede encontrarse en el Apéndice A.17.

2.5.3.2. Cuantificación Existencial (Existential Quantification)

Una *cuantificación existencial* SBVR es una *cuantificación cuyo ámbito es una formulación lógica, con el siguiente significado: para al menos un referente de la variable introducida por la cuantificación, el significado de la formulación lógica para el referente es verdadero*. Una cuantificación existencial SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos, la expresión se transforma en una sentencia OWL 2 *'ObjectSomeValuesFrom(OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de objeto *'OPE'* a un individuo de la clase *'CE'*. La expresión SBVR *'has in charge some employee'* es un ejemplo de cuantificación existencial del tipo antes mencionado.
Si la cuantificación se aplica sobre un concepto verbal vinculando conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'DataSomeValuesFrom(DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran vinculados por la propiedad de datos *'DPE'* a literales en el rango de datos *'DR'*. La expresión *'has some skill'* es un ejemplo de cuantificación existencial del tipo antes descripto.
La especificación formal de este patrón puede encontrarse en el Apéndice A.18.

2.5.3.3. Cuantificación Como Máximo N (At Most N Quantification)

SBVR define la cuantificación *'como máximo n'* como *"cuantificación que tiene una cardinalidad máxima de valor n - donde n es un entero positivo - y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es mayor que la cardinalidad máxima n"*. Una cuantificación como máximo n SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos, la expresión se transforma en una sentencia OWL 2 *'ObjectMaxCardinality(n OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de objeto *'OPE'* a *'n'* individuos diferentes como máximo, los cuales pertenecen a la clase *'CE'*. La expresión SBVR *'has in charge at most 25 employees'* es un ejemplo de cuantificación del tipo antes mencionado.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'DataMaxCardinality(n DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE'* a *'n'* literales diferentes como máximo, en el rango de datos *'DR'*. La expresión SBVR *'has at most 1 HR-ID'* es un ejemplo de cuantificación de este tipo.

La especificación formal de este patrón puede encontrarse en el Apéndice A.19.

2.5.3.4. Cuantificación Al Menos N (At Least N Quantification)

SBVR define la cuantificación *'al menos n'* como *"cuantificación que tiene una cardinalidad mínima n - donde n es un entero positivo - y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es menor que la cardinalidad mínima n"*. Una cuantificación al menos n SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos, la expresión se transforma en una sentencia OWL 2 *'ObjectMinCardinality(n OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran conectados

por la propiedad de objeto *'OPE'* a *'n'* individuos diferentes al menos, los cuales pertenecen a la clase *'CE'*. La expresión SBVR *'has in charge at least 1 employee'* es un ejemplo de cuantificación del tipo antes mencionado.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'DataMinCardinality(n DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE'* a *'n'* literales diferentes al menos, en el rango de datos *'DR'*. La expresión SBVR *'has at least 1 skill'* es un ejemplo de cuantificación de este tipo.
La especificación formal de este patrón puede encontrarse en el Apéndice A.20.

2.5.3.5. Cuantificación Exactamente N (Exactly N Quantification)

SBVR define la cuantificación *'exactamente n'* como *"cuantificación que tiene una cardinalidad n - donde n es un entero positivo - y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), es igual al de la cardinalidad n"*. Una cuantificación exactamente n SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos, la expresión se transforma en una sentencia OWL 2 *'ObjectExactCardinality(n OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de objeto *'OPE'* a exactamente *'n'* individuos diferentes, los cuales pertenecen a la clase *'CE'*. La expresión SBVR *'is in charge of exactly 1 boss'* es un ejemplo de cuantificación del tipo antes mencionado.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'DataExactCardinality(n DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE'* a exactamente *'n'* literales diferentes, en el rango de datos *'DR'*. La expresión SBVR *'has exactly 1 HR-ID'* es un ejemplo de cuantificación de este tipo.
La especificación formal de este patrón puede encontrarse en el Apéndice A.21.

2.5.3.6. Cuantificación Como Máximo Uno (At Most One Quantification)

SBVR define la cuantificación *'como máximo uno'* como *"cuantificación que tiene una cardinalidad máxima de valor $n = 1$ y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es mayor a 1"*. Una cuantificación como máximo uno SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia OWL 2 *'ObjectMaxCardinality(* 1 *OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de objeto *'OPE'* a como máximo *'*1*'* individuo perteneciendo a la clase *'CE'*. La expresión SBVR *'works in at most 1 Department'* es un ejemplo de cuantificación del tipo antes mencionado.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia OWL 2 *'DataMaxCardinality(* 1 *DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE'* a como máximo *'*1*'* literal en el rango de datos *'DR'*. La expresión SBVR *'has at most 1 HR-ID'* es un ejemplo de cuantificación de este tipo.

La especificación formal de este patrón puede encontrarse en el Apéndice A.22.

2.5.3.7. Cuantificación Exactamente Uno (Exactly One Quantification)

SBVR define la cuantificación *'exactamente uno'* como *"cuantificación que tiene una cardinalidad de valor $n = 1$ y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), es igual a 1"*. Una cuantificación exactamente uno SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia OWL 2 *'ObjectExactCardinality(* 1 *OPE CE)'*, la cual contiene todos aquellos individuos que se encuentran conectados

por la propiedad de objeto *'OPE'* a exactamente *'1'* individuo perteneciendo a la clase *'CE'*. La expresión SBVR *'is in charge of exactly 1 Boss'* es un ejemplo de cuantificación del tipo antes mencionado.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia OWL 2 *'DataExactCardinality(1 DPE DR)'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE'* a exactamente *'1'* literal en el rango de datos *'DR'*. La expresión SBVR *'has exactly 1 HR-ID'* es un ejemplo de cuantificación de este tipo.
La especificación formal de este patrón puede encontrarse en el Apéndice A.23.

2.5.3.8. Cuantificación de Rango Numérico (Numeric Range Quantification)

SBVR define la cuantificación *'de rango numérico'* como *"cuantificación que tiene una cardinalidad mínima y una cardinalidad máxima (donde la cardinalidad máxima es mayor a la cardinalidad mínima) y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es menor a la cardinalidad mínima ni mayor a la cardinalidad máxima"*. Una cuantificación de rango numérico SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se trasforma en una sentencia OWL 2 *'ObjectIntersectionOf(ObjectMinCardinality(n_1 OPE CE) ObjectMaxCardinality(n_2 OPE CE))'*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de objeto *'OPE'* al menos a *'n_1'* y como máximo a *'n_2'* individuos diferentes pertenecientes a la clase *'CE'*. La expresión SBVR *'has in charge at least 1 Employee and has in charge at most 25 Employee'* es un ejemplo de cuantificación del tipo antes mencionado.
Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia OWL 2 *DataIntersectionOf(DataMinCardinality(n_1 DPE DR) DataMaxCardinality(n_2 DPE DR))*, la cual contiene todos aquellos individuos que se encuentran conectados por la propiedad de datos *'DPE'* al menos a *'n_1'* y como máximo a *'n_2'* literales diferentes

pertenecientes al rango de datos *'DR'*. La expresión SBVR *'has at least 1 HR-ID and has at most 1 HR-ID'* es un ejemplo de cuantificación de este tipo.
La especificación formal de este patrón puede encontrarse en el Apéndice A.24.

2.5.4. Categorizaciones

La transformación de las categorizaciones SBVR - *'especialización/generalización'*, *'esquema de categorización'*, y *'segmentación'* - son descriptas en las siguientes secciones.

2.5.4.1. Especialización/Generalización (Specialization/Generalization)

La *especialización* es utilizada para representar una relación de jerarquía entre conceptos. Sean conceptos C_1 y C_2, donde C_1 especializa a C_2. Luego C_1 incorpora cada característica que forma parte de C_2, más al menos una característica adicional diferenciadora. Una especialización SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de conceptos involucrados.
Si la especialización se aplica sobre conceptos generales, la expresión se transforma en una sentencia OWL 2 *'SubClassOf(CE_1 CE_2)'*, la cual expresa la relación de jerarquía entre las expresiones de clase *'CE_1'* y *'CE_2'*.
Si la especialización se aplica sobre un concepto verbal que vincula conceptos con literales, la expresión se transforma en una sentencia OWL 2 *'SubDataPropertyOf(DPE_1 DPE_2)'*, la cual expresa la relación de jerarquía entre las propiedades de datos OWL 2 *'DPE_1'* y *'DPE_2'*.
Si la especialización se aplica sobre un concepto verbal que vincula conceptos, la expresión se transforma en una sentencia OWL 2 *'SubObjectPropertyOf(OPE_1, OPE_2)'*, la cual expresa la relación de jerarquía entre las propiedades de objetos OWL 2 *'OPE_1'* y *'OPE_2'*.
Si la especialización se aplica entre un concepto individual y un concepto general, la expresión es interpretada como una sentencia de clasificación y transformada como se describe en la Sección 2.5.1.7.
La especificación formal de este patrón puede encontrarse en el Apéndice A.25.

2.5.4.2. Esquema de categorización (Categorization Scheme)

SBVR define *esquema de categorización* como un medio de particionar conceptos individuales pertenecientes a un concepto general determinado, de acuerdo a un conjunto de categorías predefinidas para dicho esquema. La definición de diferentes esquemas de categorización - y sus correspondientes categorías - para un mismo concepto general permite clasificar sus conceptos individuales en función de diversos criterios: cada esquema representa un criterio dado el cual resulta materializado a través de sus respectivas categorías. Finalmente, un esquema de categorización puede ser definido en forma (1) *parcial*, donde pueden existir individuos del concepto clasificado que no pertenezcan a ninguna de las categorías definidas, o (2) *completa*, donde cada individuo del concepto clasificado pertenece necesariamente a alguna de las categorías consideradas.
Un esquema de categorización SBVR se transforma en sentencias OWL 2 de la siguiente manera:

1. El esquema de categorización se transforma en una clase OWL 2, representado el criterio de clasificación considerado.

2. Cada una de las categorías del esquema de categorización son transformadas en clases OWL 2.

3. La clase representado el esquema de categorización es especializado por el conjunto de clases representado las categorías.

4. Se crea una propiedad de objeto OWL 2 entre la clase sobre la cual se aplica el criterio de clasificación y la clase representando dicho criterio.

Si el esquema de categorización es completo, la clase *'C'* que lo representa es definida como la unión de las categorías que la especializan (*'*CE_1 *...* CE_n*'*mediante la sentencia OWL 2 *'EquivalentClasses(C ObjectUnionOf(* CE_1 *...* CE_n *))'*.
La especificación formal de este patrón puede encontrarse en el Apéndice A.26.

2.5.4.3. Segmentación

SBVR define *segmentación* como un esquema de categorización (1) *completo*, donde cada individuo del concepto clasificado pertenece necesariamente

a alguna de las categorías consideradas y (2) *disjunto*, donde los individuos del concepto clasificado pertenecen a una y sólo una de las categorías definidas.
Una segmentación SBVR se transforma como se describe en la Sección 2.5.4.2, donde la clase *'C'* que representa el esquema de categorización es definida como la unión disjunta de las categorías que la especializan (*'CE_1 ... CE_n'*) mediante la sentencia OWL 2 *'DisjointUnion(C CE_1 ... CE_n)'*.
La especificación formal de este patrón puede encontrarse en el Apéndice A.27.

2.6. Discusión

En los apartados siguientes se presentan ciertas consideraciones referentes a los patrones propuestos.

2.6.1. Sobre expresiones modales

La primera cuestión a tratar se relaciona con las expresiones modales. Como se ha mencionado en la Sección 2.2, todas las reglas SBVR poseen una modalidad asociada: si ninguna expresión modal es utilizada en la especificación de una regla, la modalidad alética de necesidad es asumida por defecto. Por ejemplo, la regla *'each human resource has at least 1 skill'*debe ser interpretada como la regla alética *'it is necessary that each human resource has at least 1 skill'*.
Luego, resulta necesario explorar las alternativas para el modelado ontológico de reglas deónticas a fin de soportar los tipos de modalidades soportados por el metamodelo SBVR.

2.6.2. Sobre condiciones necesarias

Otra cuestión importante lo constituyen las condiciones necesarias de los conceptos descriptos mediante SBVR: las reglas estructurales generalmente describen condiciones necesarias de los conceptos estableciendo que cierta condición resulta siempre verdadera para todas las instancias de dicho concepto. Sin embargo, es importante no confundir el uso del término 'each' con la especificación de una restricción universal. Dicha expresión implica la definición

de una condición necesaria que es modelada en OWL 2 mediante el axioma de sub-clasificación. Luego, la sentencia OWL 2 *SubClassOf(C CE)* relaciona la clase 'C' con su condición necesaria 'CE', la cual ha sido modelada por medio de una expresión de clase.

2.6.3. Sobre las implicaciones

Es importante remarcar también que no se ha definido un patrón para el mapeo de la operación lógica de implicación, dado que el lenguaje OWL 2 no dispone de primitivas que permitan el modelado de tales sentencias. Sin embargo, es posible modelar tales expresiones mediante las Cláusulas de Horn del lenguaje SWRL descripto en la Sección 2.4. El modelado de la restricción de los conceptos unitarios (Sección 2.5.1.2) es un ejemplo de modelado de implicaciones lógicas mediante dichas cláusulas.

2.6.4. Sobre las relaciones parte-de

SBVR define *concepto verbal partitivo* como un concepto verbal binario donde cada instancia representa la relación existente entre una parte y un todo, la cual ubica, lista o categoriza una parte o componente en un total o grupo mayor, una clase o una agregación. Entre los ejemplos que menciona el estándar pueden destacarse:

- un país incluido en una región geográfica
- un modelo de autos incluido en un grupo de autos
- un tambor metálico incluido en un portaminas

Sin embargo, tanto la definición como los ejemplos evidencian la falta de una semántica formal para el concepto verbal partitivo. El primer ejemplo describe una relación basada en la locación de las entidades participantes. El segundo muestra una relación de membresía en una colección de individuos. El último presenta la relación estructural existente entre un componente y el todo que compone. Resulta claro que tales expresiones poseen una semántica

completamente distinta.

La presente tesis considera la propuesta de Keet y Artale (2008) como la mejor alternativa para efectuar tal distinción, dado que presenta una taxonomía formal de relaciones entre una parte y un todo basándose en (1) las propiedades inherentes a la relación en sí misma, y (2) los tipos de las entidades participantes en la relación.

Con respecto a las propiedades de la relación, la principal distinción se basa en las nociones de *mereología* y *meronimia*. La mereología es una sub-disciplina filosófica que trata consideraciones ontológicas formales de las relaciones entre las partes y un todo. La meronimia estudia tales relaciones desde una perspectiva lingüístico-cognitiva. En consecuencia, ciertos usos de la relación *parte-de* en el lenguaje natural y el modelado conceptual no poseen las mismas propiedades que la relacion *parte-de* en el contexto de un estudio mereológico. El común denominador de ambas teorías es denominado *Mereología Base*, lo cual permite establecer que la relación *parte-de* representa un orden parcial (1) *reflexivo*, (2) *anti-simétrico*, y (3) *transitivo*. En cambio, las relaciones meronímicas pueden ser tanto *transitivas* como *no-transitivas*. En cuanto a los tipos de las entidades participantes en las relaciones *parte-de*, la propuesta de los autores se basa en las categorías definidas en la ontología fundacional denominada *Descriptive Ontology for Linguistic and Cognitive Engineering* (DOLCE) (Masolo y otros, 2003).

La taxonomía propuesta por los autores considera aspectos ontológicos de las relaciones *parte-de* que resultan de utilidad durante el modelado conceptual de un dominio dado, evitando al mismo tiempo un exceso de formalismo que dificultaría su utilización práctica. Respecto a la elección de la ontología fundacional utilizada para diferenciar los tipos de entidades participantes en una relación, la elección de DOLCE es justificada dada su alto nivel de formalización, la existencia de un mapeo directo al lenguaje OWL, y su amplia utilización en diferentes dominios para el desarrollo de sistemas de información dirigidos por ontologías.

En conclusión, un concepto verbal partitivo SBVR corresponde a un concepto verbal binario, cuya transformación ha sido descrita en la Sección 2.5.1.4. Sin embargo, esta tesis considera que las sentencias ontológicas resultantes de la transformación deberían ser enriquecidas de acuerdo a la propuesta de Keet y Artale (2008), a fin de lograr una representación del dominio más precisa explicitando las diferencias semánticas identificadas.

2.7. Conclusiones

Este capítulo presenta un conjunto de patrones de diseño de ontologías explícitamente formalizados, los cuales permiten la obtención de una ontología OWL/SWRL mediante la aplicación de un conjunto de reglas de transformación de meta-modelo sobre la especificación SBVR de un dominio de negocio. El objetivo es proveer un conjunto de transformaciones fácilmente utilizables que permita a los expertos del dominio obtener rápidamente una ontología desde la especificación en lenguaje natural de su conocimiento del negocio. Esta capacidad es un paso intermedio fundamental y necesario en la satisfacción del objetivo de investigación global del autor de esta tesis.

Los patrones propuestos han sido validados de forma heurística mediante ciclos de "generación y prueba". Esta estrategia proporciona un diseño que sirve a los propósitos de la investigación cuando la solución óptima - en este caso, una validación formal - se encuentra fuera de alcance. Luego, resulta necesario medir la "bondad" de las soluciones. Una alternativa para realizar dicha medición consiste en comparar la solución con artefactos pre-existentes que ataquen el mismo problema (Hevner y otros, 2004). En Reynares y otros (2013) y Reynares y otros (2014c) se describe un conjunto de transformaciones similares, cuya factibilidad técnica ha sido evaluada mediante la ejecución de los experimentos publicados en Reynares y otros (2014a) y Reynares y otros (2014b). La descripción completa de ambos experimentos puede encontrarse en el Apéndice B y el Apéndice C, respectivamente. La aplicación de esta estrategia heurística e iterativa ha permitido obtener resultados de gran relevancia, lo cual ha motivado su adopción en la validación de los patrones presentados en este capítulo. Dicha validación comprende dos enfoques. El primero de ellos consiste en un marco de trabajo - denominado OQuaRE - basado en el estándar SQuaRE de evaluación de la calidad de software (Duque-Ramos y otros, 2011)(Duque-Ramos y otros, 2013)(ISO, 2005). OQuaRE evalúa una ontología independientemente del proceso ejecutado para su desarrollo, aplicando un modelo de calidad conformado por un conjunto de dimensiones. La calificación obtenida en cada una de las dimensiones permite la identificación de las fortalezas y debilidades de la ontología evaluada. El segundo enfoque considerado define un procedimiento de evaluación automática de la estructura taxonómica de una ontología por medio de un conjunto de

métricas formalizadas (Beydoun y otros, 2011). Este trabajo también proporciona una metodología algorítmica para la construcción de taxonomías de contenido especificado formalmente, el cual resulta especialmente valioso - según afirman los autores - en contextos donde las taxonomías son desarrolladas por usuarios sin formación en áreas relacionadas a la ingeniería de software o la ingeniería de conocimiento. Se encuentra en proceso de formalización un estudio detallado de la aplicación de estos patrones por parte de 42 estudiantes del último nivel de la carrera de Ingeniería en Sistemas. Aunque los resultados preliminares de las evaluaciones de calidad de las ontologías obtenidas son satisfactorios, es interesante destacar las apreciaciones subjetivas de los participantes del estudio antes mencionado. En general, los estudiantes manifestaron su conformidad con la formalización de las transformaciones mediante una herramienta conceptual - los patrones de diseño - ampliamente reconocida y utilizada en su campo de formación. Por otro lado, los estudiantes como sujetos experimentales resultan representativos de profesionales de escasa experiencia, lo cual es actualmente la condición general de los profesionales de ingeniería de software en relación al uso de tecnologías semánticas.

Capítulo **3**

PATRON: marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías

Este capítulo presenta un marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías. En este contexto, las ontologías encapsulan el conocimiento del dominio y son utilizadas en tiempo de ejecución por el sistema de software. El marco de trabajo se denomina ***PATRON*** y se encuentra basado en la utilización de los patrones de diseño de ontologías definidos en el Capítulo 2.

3.1. Aspectos generales

3.1.1. Objetivos

PATRON ha sido concebido para satisfacer tres objetivos fundamentales:

1. Desarrollar desde cero una ontología con la finalidad de (a) encapsular el conocimiento de un dominio dado, (b) soportar un subconjunto de los requerimientos funcionales de un sistema de software, y (c) ser utilizada en tiempo de ejecución por dicho sistema.

2. Facilitar la integración de la ontología y el software, mediante la ejecución paralela de las actividades de desarrollo de ambos artefactos.

3. Posibilitar la rápida incorporación de las nuevas reglas de negocio en la ontología resultante, mediante la definición de un modelo iterativo de

desarrollo.

Es importante destacar que la utilización de una ontología en la forma antes mencionada constituye una decisión arquitectónica que excede el alcance de la propuesta de esta tesis. La ejecución de las actividades de elicitación de requerimientos del sistema de software y de toma de decisiones arquitectónicas preceden a la aplicación del marco de trabajo presentado (Pohl, 2010).

3.1.2. Roles

PATRON requiere del trabajo colaborativo de expertos del dominio, ingenieros ontológicos, e ingenieros de software. Los expertos del dominio poseen el conocimiento a ser modelado, producto de su interacción cotidiana con el dominio del problema. Los ingenieros ontológicos son responsables del modelado del dominio de la forma que resulte más adecuada a los requerimientos funcionales del sistema. Los ingenieros de software aportan su conocimiento en el desarrollo de las aplicaciones de software. Facilitar la comunicación entre estos perfiles tan disímiles requiere que las etapas tempranas del proceso general sean ejecutadas en un alto nivel de abstracción. Por otro lado, la naturaleza iterativa del proceso permite a los participantes tomar ventaja de lo que fue aprendido durante la generación de versiones previas de la ontología y el sistema. Dicho aprendizaje proviene tanto del proceso de construcción como del uso de la ontología desarrollada.

3.1.3. Proceso

PATRON adopta un enfoque iterativo e incremental en consecuencia a los objetivos antes enunciados. Su aplicación posibilita la evolución gradual de la ontología desarrollada mediante la incorporación cíclica de los nuevos requerimientos a ser soportados. La propuesta se basa en la ejecución de iteraciones de desarrollo restringidas en su duración temporal, lo cual permite reflejar rápidamente la dinámica conceptual del dominio y las consecuentes modificaciones en los requerimientos del sistema de software. Cada iteración concluye con el desarrollo de una nueva versión de la ontología y su posterior

integración con el código de programa.
La naturaleza cíclica de ***PATRON*** impone una restricción importante en cuanto a la elección del proceso de desarrollo del software, el cual debe adoptar el mismo enfoque a los fines de posibilitar la integración de las sucesivas versiones de ambos artefactos. La aplicación de ***PATRON*** junto a un proceso de desarrollo de sistemas de software puede apreciarse en la Figura 3.1.
Los 4 procesos que componen ***PATRON*** se presentan a continuación.

1. *Proceso 1: Análisis y Especificación.* Establecer las razones por las cuales la ontología será desarrollada, quiénes y de qué manera la utilizarán, y cuáles son los requerimientos que debe satisfacer en la iteración actual.

2. *Proceso 2: Desarrollo de la ontología.* Desarrollar la ontología que soporta los requerimientos funcionales seleccionados previamente.

3. *Proceso 3: Integración de las ontologías.* Integrar las ontologías desarrolladas en la iteración actual y la iteración previa. Cada iteración produce una ontología que soporta un conjunto disjunto de requerimientos funcionales. Luego, la ejecución de este proceso permite integrar en una única ontología la satisfacción de la totalidad de los requerimientos seleccionados hasta la iteración actual.

4. *Proceso 4: Integración de la ontología y el código de programa.* Integrar la ontología y los requerimientos funcionales implementados en código de programa a fin de conformar la aplicación de software.

5. Repetir desde el proceso 1.

Los procesos mencionados involucran la ejecución de diversas *actividades de desarrollo*, las cuales permiten evolucionar desde un modelo abstracto hasta una ontología implementada en un lenguaje procesable por una máquina. Sin embargo, el éxito de tales procesos requiere de la ejecución paralela de ciertas *actividades de soporte*. Las actividades de soporte comprenden la *elicitación de conocimiento* y la *evaluación* de los artefactos generados. La importancia de cada una de ellas depende de la naturaleza de la tarea desarrollada. Por ejemplo, la elicitación de conocimiento resulta fundamental durante las actividades de especificación y conceptualización de la ontología, mientras que la evaluación gana importancia y

complejidad a medida que se avanza con la ejecución del proceso de desarrollo. Los procesos y actividades mencionados pueden apreciarse en la Figura 3.2. En las secciones posteriores se describe cada una de las actividades detallando (1) qué tarea se realiza, (2) cuál es el objetivo de su ejecución, (3) cuáles son los artefactos requeridos por la actividad, (4) cuáles son los artefactos resultantes de la actividad, y (5) quién o quiénes se ven involucrados en su ejecución. En aquellos casos en los cuales se requieren detalles adicionales se adjuntan notas aclaratorias. Los procesos y actividades se ilustran en el Capítulo 4 mediante la presentación de los artefactos resultantes de su ejecución en la resolución de un caso de estudio.

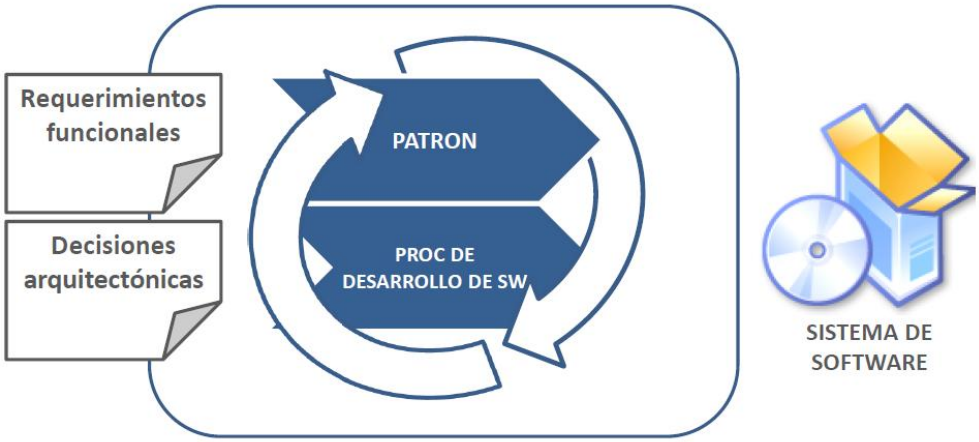

Figura 3.1: ***PATRON*** y el proceso de desarrollo de software

3.2. Proceso 1: Análisis y Especificación

Las actividades de desarrollo involucradas en la ejecución de este proceso comprenden (1) la identificación de requerimientos, (2) la priorización de tales requerimientos, (3) la identificación y priorización de entidades de dominio, y (4) la especificación de la ontología. En forma paralela, se ejecutan las actividades de soporte destinadas a la elicitación de conocimiento y la evaluación de los artefactos generados. El proceso completo puede apreciarse en la Figura 3.3. Cada una de las actividades se describen en las siguientes secciones.

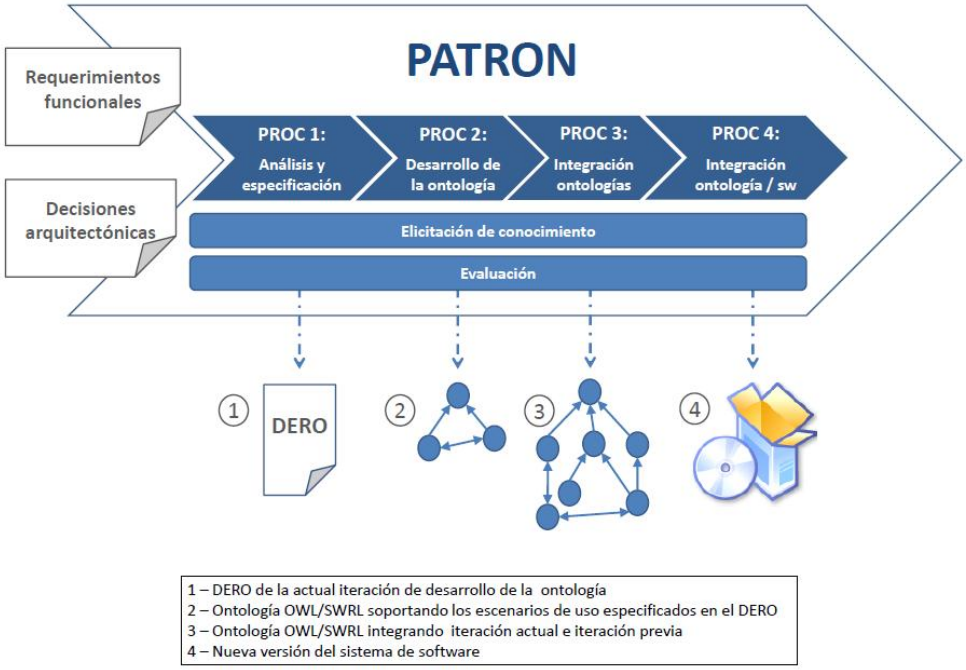

Figura 3.2: Procesos de ***PATRON***

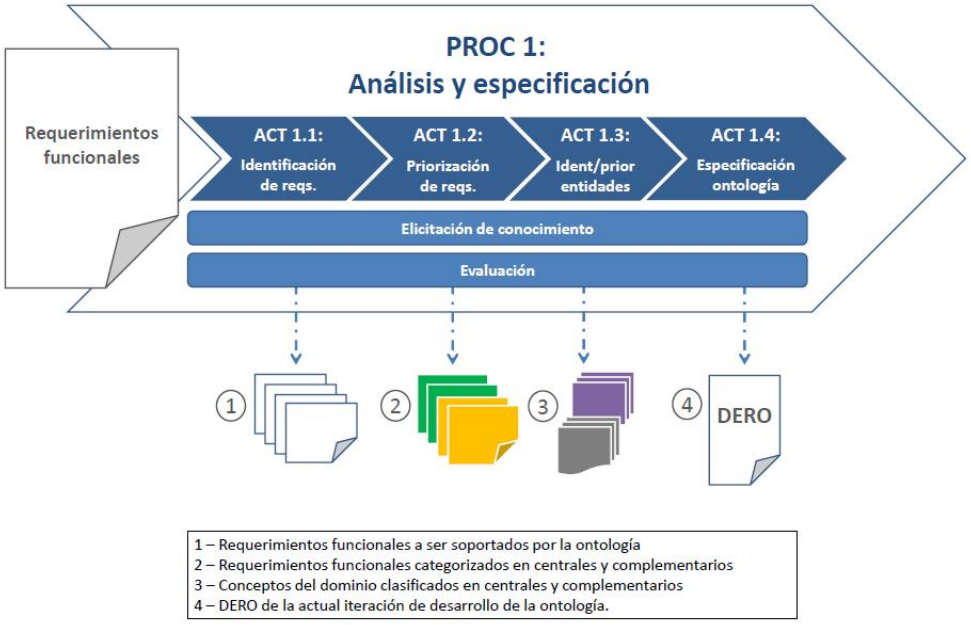

Figura 3.3: Proceso 1: Análisis y Especificación

3.2.1. Actividad 1.1: Identificación de requerimientos

En qué consiste:

Identificar los requerimientos funcionales del sistema de software a ser soportados por la ontología. Tales requerimientos involucran la ejecución de reglas de negocio.

Objetivos:

Establecer las razones por las cuales la ontología será desarrollada.

Entrada:

Requerimientos funcionales del sistema de software.

Salida:

Requerimientos funcionales a ser soportados por la ontología.

Quién:

La actividad es ejecutada colaborativamente por ingenieros ontológicos e ingenieros de software.

Cuándo:

La ejecución de las actividades de elicitación de requerimientos del sistema de software y de toma de decisiones arquitectónicas preceden a la ejecución de esta actividad.

Notas: -.

3.2.2. Actividad 1.2: Priorización de requerimientos

En qué consiste:

Priorizar los requerimientos funcionales a ser soportados por la ontología.

Objetivos:

Determinar la relevancia de los requerimientos funcionales a ser soportados por la ontología.

Entrada:

Requerimientos funcionales a ser soportados por la ontología.

Salida:

Requerimientos funcionales a ser soportados por la ontología categorizados en dos grupos: requerimientos centrales y requerimientos complementarios.

Quién:

La actividad es ejecutada colaborativamente por expertos del dominio, ingenieros ontológicos e ingenieros de software. El conocimiento y experiencia de los expertos del dominio resulta indispensable a fin de determinar adecuadamente las prioridades.

Cuándo:

La identificación de los requerimientos funcionales del sistema a ser soportados por la ontología precede a la ejecución de esta actividad.

Notas:

Esta categorización facilita la posterior determinación de las entidades centrales del dominio, las cuales conformarán el núcleo de la ontología a desarrollar. Aún en entornos de alto dinamismo, tales entidades difícilmente son alteradas y generalmente se encuentran involucradas en la satisfacción de los requerimientos funcionales más relevantes del sistema de software en proceso de desarrollo.

3.2.3. Actividad 1.3: Identificación y priorización de entidades de dominio

En qué consiste:

Identificar y priorizar las entidades del dominio involucradas en la satisfacción de los requerimientos funcionales a ser soportados por la ontología.

Objetivos:

Identificar las componentes conceptuales fundamentales de la ontología a desarrollar.

Entrada:

Requerimientos funcionales a ser soportados por la ontología categorizados en dos grupos: requerimientos centrales y requerimientos complementarios.

Salida:

Conceptos del dominio involucrados en la satisfacción de los requerimientos funcionales a ser soportados por la ontología. Los conceptos identificados se clasifican en dos grupos: conceptos centrales y conceptos complementarios. Para cada concepto, se especifican los requerimientos funcionales en los cuales se encuentra involucrado.

Quién:

La actividad es ejecutada colaborativamente por ingenieros ontológicos e ingenieros de software.

Cuándo:

La priorización de los requerimientos funcionales a ser soportados por la ontología precede a la ejecución de esta actividad.

Notas:

Los conceptos identificados son priorizados de acuerdo a la relevancia del requerimiento de mayor nivel en el cual se vean involucrados.

3.2.4. Actividad 1.4: Especificación de la ontología

En qué consiste:

Definir el *Documento de Especificación de Requerimientos de la Ontología (DERO)* de la iteración actual de desarrollo de la ontología.

Objetivos:

Establecer las razones por las cuales la ontología será desarrollada, de qué manera será utilizada, quiénes la utilizarán, y cuáles son los requerimientos que debe satisfacer.

Entrada:

- Requerimientos funcionales a ser soportados por la ontología categorizados en dos grupos: requerimientos centrales y requerimientos complementarios.
- Conceptos del dominio involucrados en la satisfacción de los requerimientos funcionales y clasificados en dos grupos: conceptos centrales y conceptos complementarios.

Salida:

DERO de la actual iteración de desarrollo de la ontología.

Quién:

La actividad es ejecutada colaborativamente por ingenieros ontológicos e ingenieros de software.

Cuándo:

La priorización de las entidades de dominio involucradas en la satisfacción de los requerimientos funcionales precede a la ejecución de esta actividad.

Notas:

La ejecución de esta actividad resulta en la definición del Documento de Especificación de Requerimientos de la Ontología (DERO), cuya estructura general se basa en el trabajo de Suárez-Figueroa y otros (2009). A continuación se describen las secciones que componen este documento.

1. *Propósito*, estableciendo las razones por las cuales la ontología se desarrollará.

2. *Alcance*, describiendo la porción del conocimiento del dominio que la ontología modela y la granularidad con la cual lo hace.

3. *Escenarios de uso*, especificando el conjunto de requerimientos funcionales de software que la ontología debe soportar en la iteración actual. Estos escenarios surgen de la selección de los requerimientos previamente priorizados. Dado que los ciclos de desarrollo se encuentran restringidos en su duración temporal, es probable que no sea posible implementar la totalidad de los requerimientos centrales en una única iteración. En consecuencia, los escenarios de uso a soportar en una determinada iteración de desarrollo generalmente corresponden a un subconjunto de los requerimientos funcionales antes mencionados.

4. *Requerimientos*, expresados mediante la formulación de las preguntas de competencia y sus correspondientes respuestas. Las *preguntas de competencia (CQ, del inglés Competency Questions)* (Grüninger y Fox, 1995) constituyen las consultas que la ontología desarrollada debe ser capaz de responder, formuladas a nivel conceptual y expresadas informalmente en lenguaje natural. Las CQs surgen de los escenarios de uso antes definidos y su definición proporciona ciertas ventajas sobre la mera utilización de los escenarios antes mencionados. En primer lugar, la reformulación de los escenarios de uso en consultas en lenguaje natural (con sus correspondientes repuestas) posibilita la generación temprana de casos de prueba atómicos para la evaluación de calidad de la ontología resultante. En segundo lugar, este incremento de la atomicidad en la expresión de los requerimientos que la ontología debe satisfacer permite identificar las entidades complementarias del dominio. Estas entidades no resultan centrales al dominio del problema y por lo tanto no resultan fácilmente identificables en la expresión de los escenarios de uso, aunque resultan imprescindibles a los fines de satisfacer los requerimientos funcionales del sistema de software. Al concluir este proceso, resultará difícil identificar relaciones 1-a-1 entre los escenarios de uso y las CQs. Es decir, en general cada escenario de uso requiere de la satisfacción de múltiples CQs, y gran

parte de las CQs se encuentran involucradas en el soporte de múltiples escenarios de uso.

5. *Fuentes de información*, las cuales identifican el soporte material del conocimiento a ser modelado por la ontología en desarrollo.

6. *Glosario*, constituido por los conceptos e individuos identificados en la definición de las CQs y sus correspondientes respuestas. Los conceptos generalmente son representados por sustantivos comunes, adjetivos o verbos. Los individuos comúnmente surgen de la identificación de sustantivos propios. Además, cada término identificado es complementado con un valor de frecuencia - representando el número de veces que dicho término fue utilizado en la expresión de las CQs y sus respuestas -, lo cual proporciona algún indicio de la importancia relativa de cada uno de ellos.

3.2.5. Actividad de Soporte: Elicitación de conocimiento

En qué consiste:
Identificar las fuentes de información y conocimiento del dominio a ser modelado por la ontología en desarrollo.
Objetivos:
Determinar el origen del conocimiento a modelar.
Entrada: -.
Salida:
Fuentes de información y conocimiento a ser utilizadas en el proceso de desarrollo de la ontología.
Quién:
La actividad es ejecutada colaborativamente por expertos del dominio e ingenieros ontológicos.
Cuándo:
La ejecución de las actividades de elicitación de requerimientos del sistema de software y de toma de decisiones arquitectónicas preceden a la ejecución de esta actividad.
Notas:
Las fuentes de información y conocimiento pueden ser agrupadas en tres grandes

categorías: (1) *Estructuradas*, entre las que se encuentran las bases de datos, los tesauros, y las ontologías pre-existentes; (2) *Semi-estructuradas*, como reportes técnicos, formularios y documentos XML; y (3) *no-estructuradas*, entre las que pueden contarse manuales, estándares y los expertos del dominio. Esta clasificación permite la identificación de técnicas y herramientas a utilizar para realizar la minería de conocimiento (Gómez-Pérez y otros, 2004)(Gangemi y Presutti, 2009). La disponibilidad y necesidad de fuentes específicas de conocimiento se encuentra fuertemente relacionada al dominio de aplicación y al proyecto de software global del cual forma parte la ontología a desarrollar.

3.2.6. Actividad de Soporte: Evaluación

En qué consiste:
Inspeccionar el DERO generado por la actividad de especificación de la ontología.
Objetivos:
Identificar la falta de CQs necesarias para la satisfacción de los escenarios de uso y/o la existencia de inconsistencias o contradicciones en el conjunto de requerimientos definidos mediante dichas CQs.
Entrada:

- Requerimientos funcionales a ser soportados por la ontología categorizados en dos grupos: requerimientos centrales y requerimientos complementarios.
- Conceptos del dominio involucrados en la satisfacción de los requerimientos funcionales y clasificados en dos grupos: conceptos centrales y conceptos complementarios.

Salida:
DERO de la actual iteración de desarrollo de la ontología.
Quién:
La actividad es ejecutada colaborativamente por expertos del dominio e ingenieros ontológicos.
Cuándo:
La ontología es especificada en forma paralela a la ejecución de esta actividad de evaluación.
Notas:

Las características a satisfacer por el DERO resultante de la actividad de especificación de la ontología se basan en el trabajo de Suárez-Figueroa y otros (2009), siendo reformuladas a fin de adaptarse a los objetivos de ***PATRON***. A continuación se describen dichas características:

- *Correcto.* Un conjunto de requerimientos es correcto si cada uno de ellos es necesario para la satisfacción de los escenarios de uso y debe ser soportado por la ontología.
- *Completo*, cuando la totalidad de los escenarios de uso considerados se encuentra soportada por medio de la satisfacción de las CQs definidas.
- *Internamente consistente*, cuando no existen conflictos en el conjunto de requerimientos definidos. Tales conflictos surgen de la utilización de terminología diferente para referirse a un mismo concepto o entidad.
- *Verificable*, lo cual posibilita la definición de uno o varios casos de prueba que permitan evaluar su satisfacción en la ontología resultante.
- *Comprensible.* La totalidad de los requerimientos debe ser comprensible para cada uno de los involucrados en la ejecución de la actividad de especificación de la ontología.
- *No ambigüo.* Esta característica es satisfecha cuando la totalidad de los requerimientos definidos es interpretada de una única manera por cada uno de los involucrados en la ejecución de la actividad de especificación de la ontología.
- *Conciso.* Cada requerimiento es único - es decir, no debe existir otro requerimiento equivalente -, y debe encontrarse asociado a por lo menos un escenario de uso.

3.3. Proceso 2: Desarrollo de la ontología

Las actividades de desarrollo involucradas en la ejecución de este proceso comprenden (1) la conceptualización de la ontología, (2) su implementación, y (3) refinamiento. En forma paralela, se ejecutan las actividades de soporte destinadas a la elicitación de conocimiento y la evaluación de los artefactos generados. El

proceso completo puede apreciarse en la Figura 3.4. Cada una de las actividades se describen en las siguientes secciones.

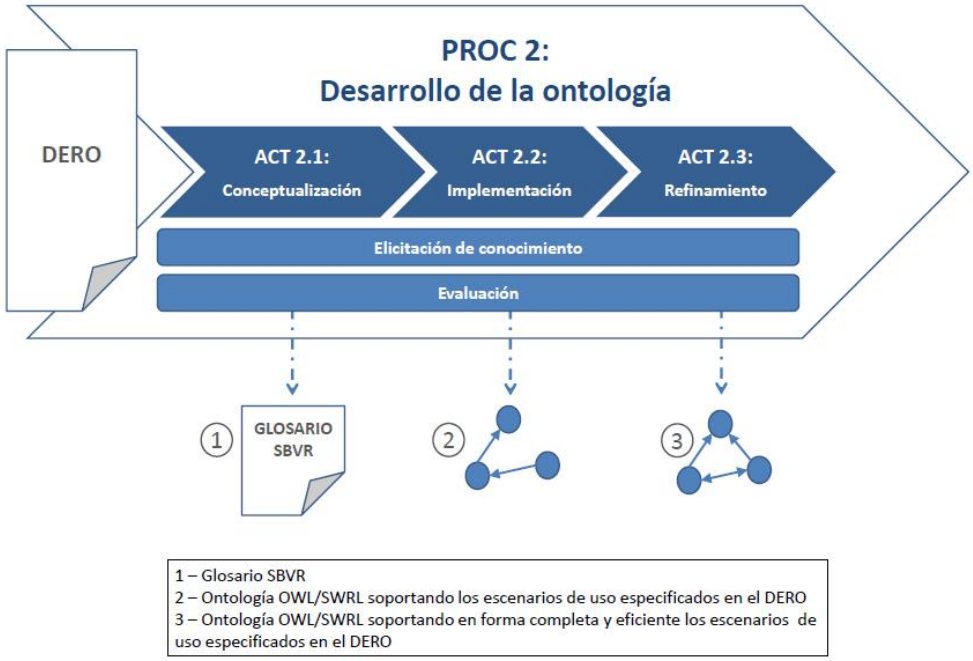

Figura 3.4: Proceso 2: Desarrollo de la ontología

3.3.1. Actividad 2.1: Conceptualización de la ontología

En qué consiste:

Expresar el conocimiento del dominio mediante la especificación de sentencias SBVR.

Objetivos:

Posibilitar la expresión de un entendimiento común sobre el dominio del negocio mediante la utilización de una abstracción de alto nivel.

Entrada:

DERO resultante de la actividad descrita en la Sección 3.2.4.

Salida:

Glosario SBVR.

Quién:

La actividad es ejecutada colaborativamente por expertos del dominio e ingenieros ontológicos.

Cuándo:

El proceso de análisis y especificación descrito en la Sección 3.2 precede a la ejecución de esta actividad.

Notas:

El glosario SBVR producto de la ejecución de esta actividad modela lingüísticamente la semántica de cada uno de los conceptos del dominio. Cada entrada del glosario modela un concepto dado y adopta la estructura que a continuación se describe.

- *Término.* Representa el término lingüístico, etiqueta o nombre con que los expertos del dominio comúnmente se refieren a un concepto determinado.

- *Definición.* Expresión formal que puede ser sustituida lógicamente por el término. Representa las condiciones suficientes y necesarias para que un objeto determinado pueda ser clasificado como un miembro del término descripto.

- *Concepto padre.* Permite indicar la existencia de un concepto que generaliza aquél que está siendo descripto por la entrada actual del glosario. El ítem de definición previamente presentado puede incluir esta relación de jerarquía, haciendo innecesario el establecimiento explícito de un concepto padre.

- *Tipo de concepto.* Establece el tipo de entidad descripta de acuerdo al meta-modelo SBVR.

- *Necesidad.* Representa las condiciones necesarias de aquellos objetos que resulten miembros del término descripto.

- *Esquema de Referencia.* Determina la forma en que identificará unívocamente cada instancia de un concepto dado.

- *Ejemplo.* Permite clarificar la definición de un término dado mediante un ejemplo concreto del dominio siendo modelado.

- *Sinónimo.* Lista de términos lingüísticos lógicamente equivalentes al concepto objeto de la descripción actual.

3.3.2. Actividad 2.2: Implementación de la ontología

En qué consiste:

Aplicar los patrones de diseño de ontologías propuestos en el Capítulo 2 para la generación de una ontología OWL/SWRL desde la especificación SBVR del conocimiento del dominio.

Objetivos:

Generar una ontología OWL/SWRL que brinde soporte a los escenarios de uso especificados en el DERO.

Entrada:

- DERO resultante de la actividad de especificación de la ontología (Sección 3.2.4).
- Glosario SBVR.
- Patrones de diseño de ontologías.

Salida:

Ontología OWL/SWRL soportando los escenarios de uso especificados en el DERO.

Quién:

La actividad es ejecutada por ingenieros ontológicos.

Cuándo:

La conceptualización de la ontología precede a la ejecución de esta actividad.

Notas: -.

3.3.3. Actividad 2.3: Refinamiento de la ontología

En qué consiste:

Modelar las expresiones SBVR que no sea posible transformar mediante la aplicación de los patrones de diseño de ontologías propuestos en el Capítulo 2.

Objetivos:

Introducir las modificaciones necesarias para soportar en forma completa y eficiente los escenarios de uso especificados en el DERO.

Entrada:

- DERO resultante de la actividad descrita en la Sección 3.2.4.
- Glosario SBVR.
- Ontología OWL/SWRL

Salida:
Ontología OWL/SWRL soportando en forma completa y eficiente los escenarios de uso especificados en el DERO.
Quién:
La actividad es ejecutada colaborativamente por ingenieros ontológicos e ingenieros de software.
Cuándo:
La implementación de la ontología precede a la ejecución de esta actividad.
Notas:
La ejecución de esta actividad resulta necesaria por dos razones. Primero, si bien los aspectos más complejos y relevantes de un dominio pueden ser modelados ontológicamente mediante la utilización de los patrones de diseño propuestos, aún no soportan la totalidad de los constructores del meta-modelo SBVR. El modelado de implicaciones lógicas es un ejemplo en tal sentido. Segundo, los ingenieros de software poseen el conocimiento necesario para la implementación de los requerimientos funcionales del sistema en las forma más performante posible. Tal conocimiento generalmente deriva en ciertas refactorizaciones menores en la forma en que la ontología modela determinados aspectos del dominio.

3.3.4. Actividad de Soporte: Elicitación de conocimiento

En qué consiste:
Aplicar las técnicas y herramientas de elicitación de conocimiento más adecuadas a las fuentes de información identificadas durante la especificación de la ontología.
Objetivos:
Elicitar el conocimiento a modelar.
Entrada:
DERO resultante de la actividad descrita en la Sección 3.2.4.
Salida:
Glosario SBVR.

Quién:

La actividad es ejecutada colaborativamente por expertos del dominio e ingenieros ontológicos.

Cuándo:

La ontología es conceptualizada en forma paralela a la ejecución de esta actividad de elicitación.

Notas:

En Gómez-Pérez y otros (2004) y Gangemi y Presutti (2009) puede encontrarse el análisis de las diversas técnicas disponibles para la elicitación de conocimiento.

3.3.5. Actividad de Soporte: Evaluación

En qué consiste:

Aplicar procedimientos de evaluación multi-dimensional sobre la ontología desarrollada.

Objetivos:

Verificar y validar la ontología desarrollada.

Entrada:

- DERO resultante de la actividad descrita en la Sección 3.2.4.
- Glosario SBVR.
- Ontología OWL/SWRL

Salida:

- Ontología OWL/SWRL verificada y validada.
- Conjunto de pruebas unitarias verificadas por la ontología desarrollada.

Quién:

La actividad es ejecutada colaborativamente por expertos del dominio e ingenieros ontológicos.

Cuándo:

La ontología es refinada en forma paralela a la ejecución de esta actividad de evaluación.

Notas:

La evaluación de una ontología comprende actividades de verificación y validación. Las primeras permiten determinar si las definiciones implementadas soportan los requerimientos funcionales establecidos previamente. Las segundas refieren a la capacidad de la ontología de representar adecuadamente el dominio que modela (Vrandečić, 2010).
Las actividades de verificación son ejecutadas mediante la definición de un conjunto de pruebas unitarias, las cuales surgen del análisis de los requerimientos funcionales del sistema y de las preguntas de competencia de la ontología. A diferencia del enfoque seguido en el campo de la ingeniería de software, en este caso las pruebas unitarias consisten en la definición de un conjunto de consultas ontológicas que implementan las preguntas de competencia y las cuales poseen un resultado preestablecido.
Existen diversas propuestas para la ejecución de las actividades de evaluación. En Corcho y otros (2004), Völker y otros (2005), Wang (2006) y Guarino y Welty (2009) se presentan distintos enfoques para la evaluación formal de la correctitud del modelado ontológico. En Gangemi y otros (2006), Rogers (2006), Ma y otros (2009), Vrandečić (2010), Duque-Ramos y otros (2011), y Duque-Ramos y otros (2013) se describen marcos de trabajo, métodos y métricas para la determinación de la calidad global de una ontología dada.

3.4. Proceso 3: Integración de las ontologías

El alineamiento de las sucesivas versiones de la ontología constituye la única actividad de desarrollo involucrada en la ejecución de este proceso. La actividad de soporte destinada a la evaluación de los artefactos generados se ejecuta en forma paralela. El proceso completo puede apreciarse en la Figura 3.5. Cada una de las actividades se describen en las siguientes secciones.

3.4.1. Actividad 3.1: Integración ontología/ontología

En qué consiste:
Integrar las ontologías desarrolladas en la iteración actual y la iteración previa.
Objetivos:
Generar una ontología que soporte los requerimientos especificados en la iteración

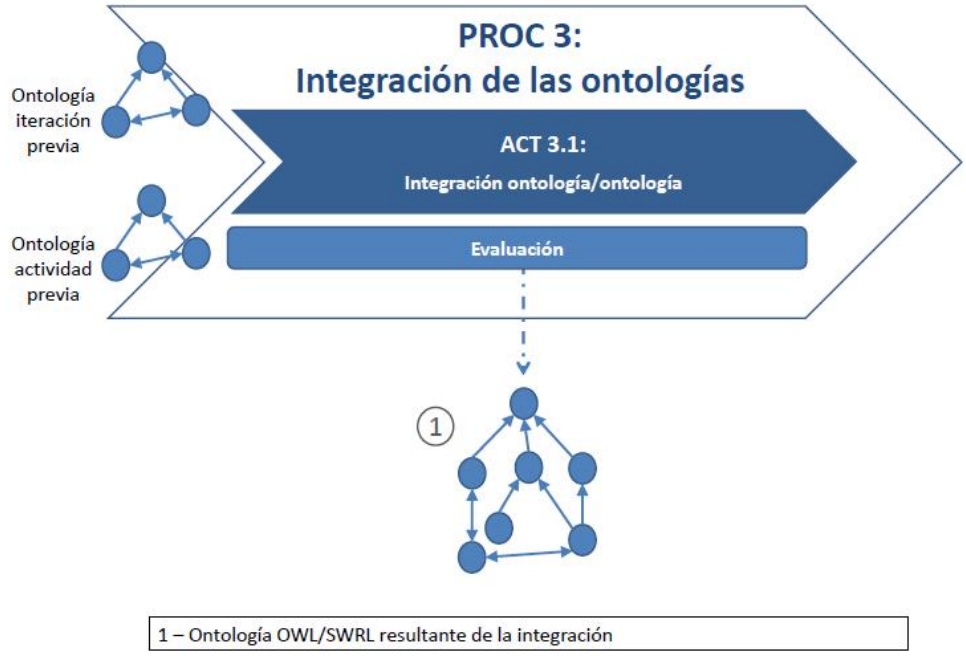

Figura 3.5: Proceso 3: Integración de las ontologías

actual y la iteración previa.

Entrada:

- Ontología OWL/SWRL resultante de la iteración previa de desarrollo.
- Ontología OWL/SWRL en desarrollo en la iteración actual.

Salida:

Ontología OWL/SWRL resultante de la integración.

Quién:

La actividad es ejecutada colaborativamente por expertos del dominio e ingenieros ontológicos.

Cuándo:

El proceso de desarrollo descrito en la Sección 3.3 precede a la ejecución de esta actividad.

Notas:

La actividad de integración consiste en la determinación de un conjunto de correspondencias entre las entidades pertenecientes a las distintas ontologías, resultando en la generación de una nueva ontología compuesta por sub-ontologías (Euzenat y otros, 2007)(Shvaiko y Euzenat, 2013). Cada correspondencia C_0 es

una tupla $\{id, e_1, e_2, r\}$ la cual establece que entre las entidades e_1 y e_2 existe la relación r, donde:

- id es el identificador de la correspondencia definida.
- e_1 y e_2 son las entidades de la ontología previa y la ontología actual, involucradas en la definición de correspondencia.
- r es la relación que establece la correspondencia entre las entidades e_1 y e_2. Tal relación puede ser de equivalencia, de generalización, de especialización, de disjunción, etc.

3.4.2. Actividad de Soporte: Evaluación

En qué consiste:
Aplicar procedimientos de evaluación funcional sobre la ontología desarrollada.
Objetivos:
Verificar la ontología desarrollada.
Entrada:

- Ontología OWL/SWRL resultante de la iteración previa de desarrollo.
- Conjunto de pruebas unitarias verificadas por la ontología desarrollada en la iteración previa.
- Ontología OWL/SWRL en desarrollo en la iteración actual.
- Conjunto de pruebas unitarias verificadas por la ontología desarrollada en la iteración actual.

Salida:
Ontología OWL/SWRL resultante de la integración (verificada).
Quién:
La actividad es ejecutada por ingenieros ontológicos.
Cuándo:
Las ontologías desarrolladas en iteraciones sucesivas son integradas en forma paralela a la ejecución de esta actividad de evaluación.
Notas:
Una iteración del proceso de desarrollo se da por finalizada sólo cuando la

totalidad de las pruebas unitarias definidas en la iteración actual y la iteración previa han sido verificadas positivamente, es decir, cuando las consultas implementadas arrojan los resultados esperados.

3.5. Proceso 4: Integración de la ontología y el código de programa

La integración de la ontología y el código de programa constituye la única actividad de desarrollo involucrada en la ejecución de este proceso. La actividad de soporte destinada a la evaluación de los artefactos generados se ejecuta en forma paralela. El proceso completo puede apreciarse en la Figura 3.6. Cada una de las actividades se describen en las siguientes secciones.

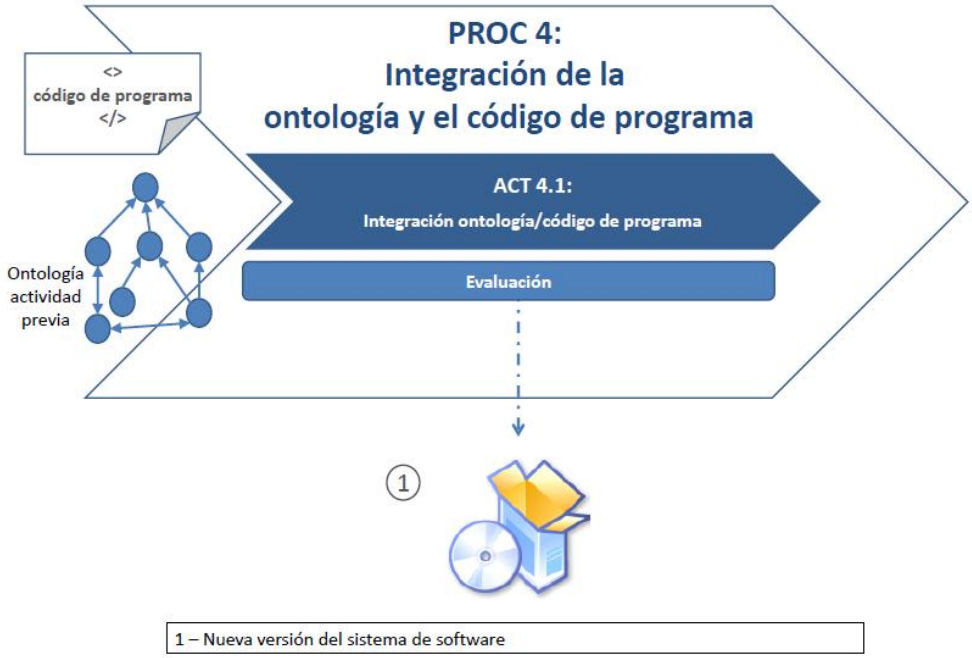

Figura 3.6: Proceso 4: Integración de la ontología y el código de programa

3.5.1. Actividad 4.1: Integración ontología/código de programa

En qué consiste:
Integrar la ontología y los requerimientos funcionales implementados en código de programa.

Objetivos:
Generar una nueva versión del sistema de software.

Entrada:

- Ontología OWL/SWRL soportando en forma completa y eficiente los escenarios de uso especificados en el DERO.
- Código de programa desarrollado en forma paralela a la construcción de la ontología.

Salida:
Nueva versión del sistema de software.

Quién:
La actividad es ejecutada colaborativamente por ingenieros ontológicos e ingenieros de software.

Cuándo:
El proceso de integración descrito en la Sección 3.4 precede a la ejecución de esta actividad.

Notas: -.

3.5.2. Actividad de Soporte: Evaluación

En qué consiste:
Aplicar procedimientos de evaluación sobre la nueva versión del sistema de software.

Objetivos:
Verificar y validar la nueva versión del sistema de software.

Entrada:
Nueva versión del sistema de software.

Salida:

Nueva versión verificada y validada del sistema de software.

Quién:

La actividad es ejecutada colaborativamente por ingenieros ontológicos e ingenieros de software.

Cuándo:

La ontología y el código de programa son integrados en forma paralela a la ejecución de esta actividad de evaluación.

Notas:

Las actividades de evaluación están destinadas a comprobar que la integración de la ontología y el código de programa conformen un sistema de software correcto y completo en cuanto a la satisfacción de los requerimientos establecidos. Los detalles relativos a la utilización de técnicas y herramientas para la ejecución de esta tarea proviene de la metología de desarrollo de software aplicada en cada proyecto en particular.

3.6. Conclusiones

Este capítulo presenta ***PATRON***, un marco de trabajo para el desarrollo integrado de sistemas de software basados en ontologías. ***PATRON*** satisface tres objetivos: (1) el desarrollo de una ontología concebida para encapsular el conocimiento de un dominio dado y destinada a soportar un subconjunto de los requerimientos funcionales de un sistema de software, (2) la integración de la ontología y el código de programa del sistema antes mencionado, y (3) la rápida incorporación de las nuevas reglas de negocio en la ontología desarrollada.

El núcleo de ***PATRON*** se encuentra conformado por los patrones de diseño presentados en el Capítulo 2, los cuales permiten obtener una ontología OWL/SWRL desde la especificación SBVR de un dominio de negocio. La conceptualización ontológica realizada por medio de un lenguaje natural controlado facilita la comunicación entre los expertos del dominio, los ingenieros ontológicos y los ingenieros de software.

El proceso adopta un enfoque iterativo e incremental, desarrollando la ontología mediante la incorporación gradual de los nuevos requerimientos. Cada iteración concluye con el desarrollo de una nueva versión de la ontología y su posterior integración con el código de programa. A los fines de posibilitar la integración de

las sucesivas versiones de ambos artefactos el proceso de desarrollo del software debe adoptar el mismo enfoque.

PATRON es resultado del análisis y posterior descripción del proceso *ad-hoc* que el autor de esta tesis ha utilizado para el desarrollo de las ontologías involucradas en sus tareas como doctorando. El dictado del curso de grado denominado *"Desarrollo de Sistemas de Información basados en Ontologías"* - el cual forma parte del último nivel de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional en la ciudad de Santa Fe, Argentina - ha permitido identificar la necesidad de proveer un conjunto de guías generales cuyo seguimiento presente una curva de aprendizaje extremadamente reducida para profesionales de la ingeniería de software. Dicho curso constituyó la oportunidad de definir ***PATRON***, proveyó el contexto necesario para evaluar la propuesta con 32 equipos distintos de desarrollo, y permitió obtener el conocimiento empírico y la realimentación necesaria para el refinamiento del proceso general. Un reporte de experiencia aplicando una primera aproximación al proceso descripto en este capítulo puede encontrarse en Gaspoz y otros (2013). Un análisis de las alternativas tecnológicas actualmente disponibles para la implementación, utilización, almacenamiento y ejecución de procesos de razonamiento sobre ontologías se presenta en Martínez y otros (2014).

Capítulo 4

PATRON: Caso de Estudio

El objetivo de este capítulo es ilustrar la ejecución de las actividades de desarrollo del marco de trabajo presentado en el Capítulo 3, con la finalidad de mostrar las transformaciones sufridas por los artefactos generados en cada una de dichas actividades y ejemplificar su evolución desde la especificación abstracta del conocimiento del dominio hasta la implementación de una ontología que soporta los requerimientos funcionales de un sistema de software.
En la Sección 4.1 se presenta una breve reseña de los requerimientos funcionales del sistema de información en base al cual se ilustrará la aplicación de ***PATRON***. La Sección 4.2 y la Sección 4.3 describen la ejecución de la primera y la segunda iteración de desarrollo, respectivamente.

4.1. Descripción general del sistema

El objetivo fundamental del sistema de software es asistir a los evaluadores de las solicitudes de Becas (1) de Ayuda Económica, (2) de Servicio, y (2) de Investigación y Desarrollo de una universidad ficticia. Dicha asistencia consiste en la elaboración automática de un ranking de postulantes de acuerdo a la normativa definida por las autoridades de dicha universidad. A continuación se presentan los requerimientos funcionales.

RF: Ingreso de usuario alumno al sistema

El sistema de software debe permitir el ingreso de usuarios que posean el rol de alumnos. El ingreso se realiza proveyendo un nombre de usuario y contraseña.

En el primer ingreso al sistema, el nombre de usuario es el número de libreta universitaria del alumno y la contraseña es su DNI. Una vez realizado el primer acceso al sistema, se exigirá al usuario proceder a la modificación de su contraseña.

RF: Ingreso de usuario profesor al sistema

El sistema de software debe permitir el ingreso de usuarios que posean el rol de profesores. El ingreso se realiza proveyendo un nombre de usuario y contraseña. En el primer ingreso al sistema, el nombre de usuario es el número de legajo del profesor y la contraseña es su DNI. Una vez realizado el primer acceso al sistema, se exigirá al usuario proceder a la modificación de su contraseña.

RF: Ingreso de usuario administrativo al sistema

El sistema de software debe permitir el ingreso de usuarios que posean el personal administrativo. El ingreso se realiza proveyendo un nombre de usuario y contraseña. En el primer ingreso al sistema, el nombre de usuario es el número de legajo del administrativo y la contraseña es su DNI. Una vez realizado el primer acceso al sistema, se exigirá al usuario proceder a la modificación de su contraseña.

RF: Cambio de contraseña de usuario

El sistema de software debe permitir la modificación de una contraseña preexistente. La contraseña debe consistir en una cadena alfanumérica de caracteres de al menos 8 (ocho) caracteres. El sistema validará la adecuada conformación de la nueva contraseña, requiriendo su modificación por parte del usuario en caso de no satisfacer el criterio antes mencionado.

RF: Alta de Convocatoria de Beca de Ayuda Ecónomica

El sistema de software debe permitir a un usuario con el rol de personal administrativo dar de alta una nueva convocatoria a Becas de Ayuda Económica. El usuario debe definir los siguientes aspectos de la convocatoria:

- Apertura de la convocatoria, lo cual determina la fecha a partir de la cual el sistema puede aceptar postulaciones.

- Cierre de la convocatoria, lo cual determina la fecha a partir de la cual el sistema debe rechazar nuevas postulaciones.
- Importe mensual de la beca a ser percibido por los futuros beneficiarios.
- Número de becas disponibles en la convocatoria, lo cual define el número máximo de beneficiarios posibles.

RF: Alta de Convocatoria de Beca de Servicio

El sistema de software debe permitir a un usuario con el rol de personal administrativo dar de alta una nueva convocatoria a Becas de Servicio. El usuario debe definir los siguientes aspectos de la convocatoria:

- Apertura de la convocatoria: determina la fecha a partir de la cual el sistema puede aceptar postulantes.
- Cierre de la convocatoria: determina la fecha a partir de la cual el sistema debe rechazar nuevos postulantes.
- Inicio de la beca: determina la fecha a partir de la cual los beneficiarios comienzan a percibir el beneficio.
- Dependencia en la cual se desarrollarán tareas técnicas no administrativas en condición de contra-prestación por el monto percibido.
- Importe mensual de la beca a ser percibido por los futuros beneficiarios.
- Número de becas disponibles en la convocatoria, lo cual define el número máximo de beneficiarios posibles.

RF: Alta de Convocatoria de Beca de Investigación y Desarrollo

El sistema de software debe permitir a un usuario con el rol de profesor dar de alta una nueva convocatoria a Becas de Investigación y Desarrollo. El usuario debe definir los siguientes aspectos de la convocatoria:

- Apertura de la convocatoria: determina la fecha a partir de la cual el sistema puede aceptar postulantes.

- Cierre de la convocatoria: determina la fecha a partir de la cual el sistema debe rechazar nuevos postulantes.
- Inicio de la beca: determina la fecha a partir de la cual los beneficiarios comienzan a percibir el beneficio.
- Dependencia en la cual se desarrollarán tareas de investigación y desarrollo en condición de contra-prestación por el monto percibido.
- Descripción (no mayor 2000 caracteres) del proyecto de investigación y desarrollo en el cual se enmarcan las tareas a ejecutar.
- Importe mensual de la beca a ser percibido por los futuros beneficiarios.
- Número de becas disponibles en la convocatoria, lo cual define el número máximo de beneficiarios posibles.

RF: Postulación a Beca de Ayuda Económica

El sistema de software debe permitir a un usuario con el rol de alumno postularse a una Beca de Ayuda Económica. El proceso de postulación sólo puede realizarse en caso que exista una convocatoria abierta y el alumno satisfaga los siguientes criterios:

- No poseer título universitario otorgado por universidad pública o privada.
- No poseer cargo rentado en una facultad.
- Ser alumno regular de alguna carrera de grado de la facultad.
- No haber cursado ni estar cursando ninguna materia de 4º (cuarto) año o superior.
- Haber aprobado al menos 5 (cinco) materias en el ciclo lectivo anterior.
- Tener un promedio (con aplazos) no menor a 5.5 (cinco punto cinco) puntos.

En caso que el postulante no verifique alguna de estas condiciones, el sistema debe rechazar la postulación informando al usuario el criterio insatisfecho.

RF: Postulación a Beca de Servicio

El sistema de software debe permitir a un usuario con el rol de alumno postularse a una Beca de Servicio. El proceso de postulación sólo puede realizarse en caso que exista una convocatoria abierta y el alumno satisfaga los siguientes criterios:

- No poseer título universitario otorgado por universidad pública o privada.
- No poseer cargo rentado en una facultad.
- Ser alumno regular de alguna carrera de grado de la facultad.
- Haber aprobado al menos 7 (siete) materias en los 2 (dos) últimos ciclos lectivos.
- Tener un promedio (con aplazos) no menor a 5.5 (cinco punto cinco) puntos.

En caso que el postulante no verifique alguna de estas condiciones, el sistema debe rechazar la postulación informando al usuario el criterio insatisfecho.

RF: Postulación a Beca de Investigación y Desarrollo

El sistema de software debe permitir a un usuario con el rol de alumno postularse a una Beca de Investigación y Desarrollo. El proceso de postulación sólo puede realizarse en caso que exista una convocatoria abierta y el alumno satisfaga los siguientes criterios:

- No poseer título universitario otorgado por universidad pública o privada.
- No poseer cargo rentado en una facultad.
- Ser alumno regular de alguna carrera de grado de la facultad.
- Haber aprobado al menos 7 (siete) materias en los 2 (dos) últimos ciclos lectivos.
- Tener un promedio (con aplazos) no menor a 6 (seis) puntos.

En caso que el postulante no verifique alguna de estas condiciones, el sistema debe rechazar la postulación informando al usuario el criterio insatisfecho.

RF: Elaboración de ranking de postulantes a Beca de Ayuda Económica

El sistema de software debe elaborar un ranking de los alumnos calificados para obtener una Beca de Ayuda Económica. Este proceso se ejecutará automáticamente al cerrar la convocatoria de la beca. Los aspectos evaluados y sus valores máximos pueden verse en la tabla:

Descripción	*Ponderación*
Nivel de ingresos	30
Condición de actividad laboral	5
Vivienda	5
Salud	5
Tasa de dependencia	5
Promedio general (con aplazos)	35
Rendimiento académico	15
TOTAL	100

La asignación de puntajes en cada uno de estos aspectos se realiza de acuerdo a los criterios presentados a continuación.

- *Ingresos*
  Se consideran los ingresos mensuales (x_i) del grupo familiar. Para su ponderación se considera un piso mínimo y una función continua (entre k_1 y k_2 pesos de ingreso).
  Si $x_1 = k_1$ = valor canasta familiar entonces = 30 puntos
  Si $x_1 = k_1 * 2$ entonces = 0 puntos
  Para valores intermedios: 30 ? $0.15 * (x_1$? $k_1)$ = Puntaje

- *Condición de Actividad Laboral*
  Puntaje = Suma de puntos totales de miembros económicamente activos / Cantidad de miembros económicamente activos

Categoría	*Valor*
Ocupado	0
Sub-ocupado	3
Desocupado	5

- *Vivienda*

  Se toma como indicador la situación jurídica del inmueble que habita el postulante.

Categoría	*Valor*
Propietario sin deuda	0
Propietario con deuda	3
Inquilino u Ocupante	5

- *Salud*

  Se considera la condición de cobertura de salud del postulante. El servicio de medicina pre-paga se considera cobertura integral.

Categoría	*Valor*
Cobertura Total	0
Cobertura Parcial	3
Sin cobertura	5

- *Tasa de dependencia*

  Se mide de acuerdo a la cantidad de hijos menores que residan en el hogar.

Categoría	*Valor*
Sin hijos	0
1-3 hijos	3
Más de 3 hijos	5

- *Promedio general*

  Se multiplica el promedio (con aplazos) por 3.5.

  En caso de ingresantes, el puntaje por defecto es 14.

- *Rendimiento académico*
  Puntaje = (Materias aprobadas / Materias regularizadas) $*$ 10
  En caso ingresantes, el puntaje por defecto es 4.

RF: Elaboración de ranking de postulantes a Beca de Servicio

El sistema de software debe elaborar un ranking de los alumnos calificados para obtener una Beca de Servicio. Este proceso se ejecutará automáticamente al cerrar la convocatoria de la beca de acuerdo a la siguiente fórmula de cálculo.
Puntaje = Promedio General + (Nº de materias aprobadas $*$ 0.1) + ((Nº de materias cursadas $*$ 3) / Total de materias de la carrera) + (2 / (1 + Nº aplazos))
Donde: promedio general = promedio (con aplazos) $*$ 3.5. En caso de ingresantes, el puntaje por defecto es 14.

RF: Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo

El sistema de software debe elaborar automáticamente un ranking de los alumnos calificados para obtener una Beca de Investigación y Desarrollo. Este proceso se ejecutará automáticamente al cerrar la convocatoria de la beca de acuerdo a la siguiente fórmula de cálculo.
Puntaje = Promedio General + (Nº de materias aprobadas $*$ 0.1) + ((Nº de materias cursadas $*$ 3) / Total de materias de la carrera) + (2 / (1 + Nº aplazos))
Donde: promedio general = promedio (con aplazos) $*$ 3.5. En caso de ingresantes, el puntaje por defecto es 14.

4.1.0.1. RF: Aviso de beneficiario de Beca

El sistema de software deberá enviar un mail a cada uno de los beneficiarios de una beca determinada, informando la calificación y posición obtenida en el proceso de elaboración del ranking de postulantes.

4.2. Primera Iteración

4.2.1. Proceso 1: Análisis y Especificación

4.2.1.1. Actividad 1.1: Identificación de Requerimientos

A continuación se presentan los requerimientos funcionales del sistema que involucran la ejecución de reglas de negocio y, en consecuencia, serán soportados por la ontología:

- *RF: Postulación a Beca de Ayuda Económica.*
- *RF: Postulación a Beca de Servicio.*
- *RF: Postulación a Beca de Investigación y Desarrollo.*
- *RF: Elaboración de ranking de postulantes a Beca de Ayuda Económica.*
- *RF: Elaboración de ranking de postulantes a Beca de Servicio.*
- *RF: Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.*

4.2.1.2. Actividad 1.2: Priorización de Requerimientos

La totalidad de los requerimientos funcionales a ser soportados por la ontología se consideran *centrales*, dada la íntima relación existente entre los mismos y su carácter fundamental en el contexto del proceso de desarrollo de software considerado.

4.2.1.3. Actividad 1.3: Identificación y Priorización de Entidades de Dominio

A continuación se describe una fracción de las entidades del dominio involucradas en la satisfacción de los requerimientos funcionales identificados en la Actividad 1.1.

1. **Alumno**
   Categoría: Concepto central
   RFs involucrados:

Postulación a Beca de Ayuda Económica.

Postulación a Beca de Servicio.

Postulación a Beca de Investigación y Desarrollo.

Elaboración de ranking de postulantes a Beca de Ayuda Económica.

Elaboración de ranking de postulantes a Beca de Servicio.

Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.

2. **Beca de Ayuda Económica**

   Categoría: Concepto central

   RFs involucrados:

   Postulación a Beca de Ayuda Económica.

   Elaboración de ranking de postulantes a Beca de Ayuda Económica.

3. **Beca de Servicio**

   Categoría: Concepto central

   RFs involucrados:

   Postulación a Beca de Servicio.

   Elaboración de ranking de postulantes a Beca de Servicio.

4. **Beca de Investigación y Desarrollo**

   Categoría: Concepto central

   RFs involucrados:

   Postulación a Beca de Investigación y Desarrollo.

   Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.

5. **Postulante Beca de Ayuda Económica**

   Categoría: Concepto central

   RFs involucrados:

   Postulación a Beca de Ayuda Económica.

   Elaboración de ranking de postulantes a Beca de Ayuda Económica.

6. **Postulante Beca de Servicio**

   Categoría: Concepto central

   RFs involucrados:

   Postulación a Beca de Servicio.

   Elaboración de ranking de postulantes a Beca de Servicio.

7. **Postulante Beca de Investigación y Desarrollo**

   Categoría: Concepto central

   RFs involucrados:

   Postulación a Beca de Investigación y Desarrollo.

   Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.

8. **Título universitario**

   Categoría: Concepto complementario

   RFs involucrados:

   Postulación a Beca de Ayuda Económica.

   Postulación a Beca de Servicio.

   Postulación a Beca de Investigación y Desarrollo.

9. **Cargo rentado**

   Categoría: Concepto complementario

   RFs involucrados:

   Postulación a Beca de Ayuda Económica.

   Postulación a Beca de Servicio.

   Postulación a Beca de Investigación y Desarrollo.

10. **Alumno regular**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

    Postulación a Beca de Servicio.

    Postulación a Beca de Investigación y Desarrollo.

11. **Carrera**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

    Postulación a Beca de Servicio.

    Postulación a Beca de Investigación y Desarrollo.

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

    Elaboración de ranking de postulantes a Beca de Servicio.

    Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.

12. **Materia**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

    Postulación a Beca de Servicio.

    Postulación a Beca de Investigación y Desarrollo.

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

    Elaboración de ranking de postulantes a Beca de Servicio.

    Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.

13. **Nivel de materia**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

14. **Ciclo lectivo**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

    Postulación a Beca de Servicio.

    Postulación a Beca de Investigación y Desarrollo.

15. **Promedio general**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

    Postulación a Beca de Servicio.

    Postulación a Beca de Investigación y Desarrollo.

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

    Elaboración de ranking de postulantes a Beca de Servicio.

    Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.

16. **Ingresos**

    Categoría: Concepto complementario

    RFs involucrados:

    Postulación a Beca de Ayuda Económica.

17. **Actividad Laboral**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

18. **Ocupado**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

19. **Sub-ocupado**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

20. **Desocupado**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

21. **Vivienda**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

22. **Propietario sin deuda**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

23. **Propietario con deuda**

    Categoría: Concepto complementario

    RFs involucrados:

    Elaboración de ranking de postulantes a Beca de Ayuda Económica.

24. **Inquilino/Ocupante**

    Categoría: Concepto complementario

RFs involucrados:

Elaboración de ranking de postulantes a Beca de Ayuda Económica.

4.2.1.4. Actividad 1.4: Especificación de la Ontología

A continuación se presenta un fragmento del DERO de la iteración actual de desarrollo de la ontología, estableciendo las razones por las cuales será desarrollada, quiénes y de qué manera la utilizarán, y cuáles son los requerimientos que deberá satisfacer.

Propósito:

Soportar la toma de decisiones en la evaluación de solicitudes de Becas de Ayuda Económica, de Servicio, y de Investigación y Desarrollo.

Alcance:

La ontología modela el conocimiento relacionado a los criterios de admisibilidad de las becas ofrecidas y aspectos personales y de rendimiento de los alumnos, a fin de ser capaz de generar un ranking de postulantes calificados para la obtención de becas de los tipos antes mencionados. El nivel de granularidad se encuentra determinado por las preguntas de competencia y los términos identificados en relación a las reglas de negocio del dominio.

Escenarios de uso:

Los requerimientos funcionales que la ontología debe soportar en la iteración actual de desarrollo son los siguientes:

- Postulación a Beca de Ayuda Económica.
- Elaboración de ranking de postulantes a Beca de Ayuda Económica.

La evaluación de los criterios de admisibilidad y la determinación de los puntajes finales de los postulantes a Becas de Ayuda Económica constituyen los procesos de mayor complejidad a ser soportados por la ontología, constituyendo requisitos críticos a ser implementados lo más tempranamente posible.

Requerimientos (fragmento):

Preguntas de competencia (CQ) y sus respuestas posibles (A).

- CQ: ¿Existen convocatorias abiertas a Beca de Ayuda Económica? A: Respuesta booleana (Si/No)

- CQ: ¿Cuál es la fecha de apertura de una convocatoria a Beca de Ayuda Económica? A: Fecha
- CQ: ¿Cuál es la fecha de cierre de una convocatoria a Beca de Ayuda Económica? A: Fecha
- CQ: ¿Cuál es el importe mensual de una Beca de Ayuda Económica? A: Importe (valor numérico)
- CQ: ¿Cuál es el número máximo de posibles beneficiarios de una Beca de Ayuda Económica? A: Valor entero
- CQ: ¿El alumno satisface los criterios de postulación de una Beca de Ayuda Económica? A: Respuesta booleana (Si/No)
- CQ: ¿El alumno posee un título universitario? A: Respuesta booleana (Si/No)
- CQ: ¿El alumno posee un cargo universitario rentado? A: Respuesta booleana (Si/No)
- CQ: ¿Qué carrera de grado cursa el alumno? A: Carreras dictadas en la facultad
- CQ: ¿El alumno se encuentra en condición de regular? A: Respuesta booleana (Si/No)
- CQ: ¿Qué materias cursa el alumno? A: Conjunto de materia en condición cursando
- CQ: ¿A qué nivel pertenecen cada una de las materias cursadas por el alumno? A: Valor entero (entre 1 y 6)
- CQ: ¿Cuántas materias aprobó el alumno en el ciclo lectivo anterior? A: Valor entero
- CQ: ¿Cuál es el promedio con aplazos del alumno? A: Valor real (entre 0.0 y 10.0)

- CQ: ¿Qué puntaje general obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su postulación a Beca de Ayuda Económica? A: Valor real (entre 0.0 y 100.0)

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su nivel de ingresos? A: Valor real (entre 0.0 y 30.0)

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su condición de actividad laboral? A: Valor real (entre 0.0 y 5.0)

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su condición de vivienda? A: Valores posibles [0, 3, 5])

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su cobertura de salud? A: Valores posibles [0, 3, 5])

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su tasa de dependencia? A: Valores posibles [0, 3, 5])

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su promedio general? A: Valor real (entre 0.0 y 14)

- CQ: ¿Qué puntaje obtuvo un postulante a Beca de Ayuda Económica en la evaluación de su rendimiento académico? A: Valor real (entre 0.0 y 15)

Fuentes de información:

Las políticas de asignación de becas estudiantiles de la universidad se encuentran especificadas originalmente en un documento oficial de la institución y descriptas en lenguaje natural.

Glosario (fragmento):

Término	*Frecuencia*
Beca de Ayuda Económica	15
Alumno	9
Puntaje General	9
Postulante a Beca de Ayuda Económica	8
Convocatoria	5
Materia	4
Carrera de Grado	2
Título Universitario	1
Ciclo Lectivo	1
Cargo Rentado	1
Alumno Regular	1
Nivel de Materia	1
Promedio General	1
Promedio con Aplazos	1
Ingresos	1
Actividad Laboral	1
Vivienda	1

4.2.2. Proceso 2: Desarrollo de la ontología

4.2.2.1. Actividad 2.1: Conceptualización de la Ontología

A continuación se presenta un fragmento del glosario SBVR, el cual modela lingüísticamente los conceptos del dominio.

Beca de Ayuda Económica
Definición: -
Concepto padre: Beca
Tipo de concepto: Concepto General
Necesidad:

- Cada Beca de Ayuda Económica posee exactamente 1 fecha de apertura de convocatoria

- Cada Beca de Ayuda Económica posee exactamente 1 fecha de cierre de convocatoria
- Cada Beca de Ayuda Económica posee exactamente 1 importe mensual
- Cada Beca de Ayuda Económica posee exactamente 1 número máximo de posibles beneficiarios

Esquema de Referencia: fecha de apertura de convocatoria, fecha de cierre de convocatoria
Ejemplo: -
Sinónimo: BAE

Alumno
Definición: Persona inscripta en al menos 1 Carrera de Grado
Concepto padre: -
Tipo de concepto: Rol de Concepto Verbal
Necesidad:

- Cada Alumno posee al menos 1 Nombre
- Cada Alumno posee al menos 1 Apellido
- Cada Alumno posee exactamente 1 DNI
- Cada Alumno posee exactamente 1 Nº Libreta Universitaria

Esquema de Referencia: Nº Libreta Universitaria
Ejemplo: -
Sinónimo: Estudiante

Postulante a Beca de Ayuda Económica
Definición: Alumno postulado en al menos 1 Beca de Ayuda Economica
Concepto padre: Postulante
Tipo de concepto: Rol de Concepto Verbal
Necesidad:

- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Puntaje General

- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Nivel de Ingresos
- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Condición de Actividad Laboral
- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Condición de Vivienda
- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Condición de Cobertura de Salud
- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Tasa de Dependencia
- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Promedio General
- Cada Postulante a Beca de Ayuda Económica posee exactamente 1 Rendimiento Académico

Esquema de Referencia: Nº Libreta Universitaria, postulado en Beca de Ayuda Económica
Ejemplo: -
Sinónimo: -

postulado en
Definición: -
Concepto padre: -
Tipo de concepto: Concepto Verbal Binario
Necesidad:

- El primer rol del concepto verbal es Alumno
- El segundo rol del concepto verbal es Beca

Esquema de Referencia: -
Ejemplo: -
Sinónimo: -

cursa
Definición: -
Concepto padre: -
Tipo de concepto: Concepto Verbal Binario
Necesidad:

- El primer rol del concepto verbal es Alumno
- El segundo rol del concepto verbal es Materia

Esquema de Referencia: -
Ejemplo: -
Sinónimo: -

4.2.2.2. Actividad 2.2: Implementación de la Ontología

A continuación se presenta un fragmento de la ontología obtenida mediante la aplicación de los patrones de diseño propuestos en el Capítulo 2.

Declaration(Class(:Alumno))
EquivalentClasses(:Alumno ObjectIntersectionOf(ObjectMinCardinality(1 :inscripto_en :Carrera_de_Grado) :Persona))
Declaration(Class(:BAE))
EquivalentClasses(:BAE :Beca_de_Ayuda_Economica)
Declaration(Class(:Beca))
Declaration(Class(:Beca_de_Ayuda_Economica))
EquivalentClasses(:Beca_de_Ayuda_Economica :BAE)
SubClassOf(:Beca_de_Ayuda_Economica :Beca)
SubClassOf(:Beca_de_Ayuda_Economica DataExactCardinality(1 :fecha_de_apertura xsd:dateTime))
SubClassOf(:Beca_de_Ayuda_Economica DataExactCardinality(1 :fecha_de_cierre xsd:dateTime))
SubClassOf(:Beca_de_Ayuda_Economica DataExactCardinality(1 :importe_mensual xsd:dateTime))

SubClassOf(:Beca_de_Ayuda_Economica DataExactCardinality(1 :nº_max_posibles_beneficiarios xsd:dateTime))
Declaration(Class(:Carrera_de_Grado))
Declaration(Class(:Persona))
Declaration(Class(:Postulante))
Declaration(Class(:Postulante_a_Beca_de_Ayuda_Economica))
EquivalentClasses(:Postulante_a_Beca_de_Ayuda_Economica
ObjectIntersectionOf(ObjectMinCardinality(1 :postulado_en :Beca_de_Ayuda_Economica) :Alumno))
SubClassOf(:Postulante_a_Beca_de_Ayuda_Economica :Postulante)
Declaration(ObjectProperty(:inscripto_en))
ObjectPropertyDomain(:inscripto_en :Persona)
ObjectPropertyRange(:inscripto_en :Carrera_de_Grado)
Declaration(ObjectProperty(:postulado_en))
ObjectPropertyDomain(:postulado_en :Alumno)
ObjectPropertyRange(:postulado_en :Beca_de_Ayuda_Economica)
Declaration(DataProperty(:fecha_de_apertura))
DataPropertyDomain(:fecha_de_apertura :Beca_de_Ayuda_Economica)
DataPropertyRange(:fecha_de_apertura xsd:dateTime)
Declaration(DataProperty(:fecha_de_cierre))
DataPropertyDomain(:fecha_de_cierre :Beca_de_Ayuda_Economica)
DataPropertyRange(:fecha_de_cierre xsd:dateTime)
Declaration(DataProperty(:importe_mensual))
DataPropertyDomain(:importe_mensual :Beca_de_Ayuda_Economica)
DataPropertyRange(:importe_mensual xsd:float)
Declaration(DataProperty(:nº_Libreta_Universitaria¿))
DataPropertyDomain(:nº_Libreta_Universitaria :Alumno)
DataPropertyRange(:nº_Libreta_Universitaria¿xsd:string)
Declaration(DataProperty(:nº_max_posibles_beneficiarios¿))
DataPropertyDomain(:nº_max_posibles_beneficiarios :Beca_de_Ayuda_Economica)
DataPropertyRange(:nº_max_posibles_beneficiarios¿xsd:integer)
HasKey(:Alumno () (:nº_Libreta_Universitaria))
HasKey(:Beca_de_Ayuda_Economica () (:fecha_de_apertura :fecha_de_cierre))
HasKey(:Postulante_a_Beca_de_Ayuda_Economica (:postulado_en)

(:nº_Libreta_Universitaria)))

4.2.2.3. Actividad 2.3: Refinamiento de la Ontología

Aquellas expresiones SBVR que resulten transformables en sentencias OWL mediante los patrones de diseño propuestos, deben ser implementadas de acuerdo al criterio y experiencia de los ingenieros ontológicos e ingenieros de software. Por ejemplo, en el caso de estudio presentado no existen patrones destinados a la transformación de los cálculos matemáticos para a la determinación de los puntajes de los criterios de otorgamiento de una beca. El criterio de rendimiento académico - para citar sólo un caso - se expresa de la siguiente manera:

Puntaje = (Materias aprobadas / Materias regularizadas) $*$ 10. En caso de ingresantes, el puntaje por defecto es 4.

Luego, se requiere de una expresión SWRL a fin de ser capaces de modelar en la ontología dicha formulación:

Postulante_a_Beca_de_Ayuda_Economica(?x),
materias_aprobadas(?x, ?y),
materias_regularizadas(?x, ?z),
divide(?r, ?y, ?z),
multiply(?s, ?r, 10) $\rightarrow$ *puntaje_rendimiento_academico(?x, ?s)*

Postulante_a_Beca_de_Ayuda_Economica(?x),
Ingresante(?x) $\rightarrow$ *puntaje_rendimiento_academico(?x, 4)*

De la misma forma debe procederse al análisis de las brechas semánticas existentes entre los meta-modelos SBVR y OWL 2, a fin de obtener una ontología que soporte en forma completa los escenarios de uso para los cuales fue concebida.

4.2.3. Proceso 3: Integración de las ontologías

La ejecución de la actividad de integración involucrada en este proceso resulta innecesaria, dado que existe una única ontología producto de una primera iteración de desarrollo.

4.2.4. Proceso 4: Integración de la ontología y el código de programa

La integración de la ontología y el código de programa constituye la única actividad de este proceso, lo cual resulta en la generación de una nueva versión del sistema de software. En esta primera iteración de desarrollo del caso de estudio, los requerimientos funcionales implementados en código de programa deberían permitir interactuar con la ontología a fin de posibilitar: (1) a los alumnos postularse a una Beca de Ayuda Económica, y (2) al personal administrativo obtener automáticamente un ranking de postulantes de acuerdo a la normativa definida por las autoridades de la universidad.

4.3. Segunda Iteración

A fin de evitar reiteraciones en la descripción de la ejecución de la totalidad de las actividades de desarrollo de ***PATRON***, esta segunda iteración ilustra de forma coloquial sólo aquellas aspectos que resulten relevantes para la comprensión global del proceso.
Como puede apreciarse en la Sección 4.1), la satisfacción de los requerimientos funcionales a ser implementados en esta iteración resultan más simples en comparación a aquellos soportados en la primera:

- *RF: Postulación a Beca de Servicio.*
- *RF: Postulación a Beca de Investigación y Desarrollo.*
- *RF: Elaboración de ranking de postulantes a Beca de Servicio.*
- *RF: Elaboración de ranking de postulantes a Beca de Investigación y Desarrollo.*

Esto se debe a dos razones. Primero, la mayor parte del conocimiento del dominio - tanto lo referente a entidades, sus relaciones y las reglas de negocio que restringen sus interacciones - ha sido especificado, modelado e implementado en la primera iteración. Segundo, los criterios a analizar para la confección del ranking de postulantes a Becas de Servicio y Becas de Investigación y Desarrollo se reducen a una simple fórmula matemática. Si bien un entorno real de desarrollo requiere de la ejecución estricta de cada uno de los procesos y actividades del marco de trabajo propuesto, esta sección se limita a reseñar brevemente la actividad de integración correspondiente al tercer proceso de ***PATRON***. La actividad consiste en la determinación de las correspondencias entre las entidades pertenecientes a las distintas versiones de la ontología, mediante el establecimiento de una relación r entre dichas entidades.
Luego, la implementación de los requerimientos funcionales de esta iteración no requiere de la modificación de las sentencias especificadas en la versión previa de la ontología, sino la incorporación de nuevas entidades y su correspondiente relación con las entidades previas. A continuación se presentan algunas de las correspondencias identificadas de acuerdo al formato presentado en el Capítulo 3.4.

$C_0 = \{0, \text{Beca}, \text{Beca de Servicio}, \supseteq\}$
$C_1 = \{1, \text{Beca}, \text{Beca de Investigación y Desarrollo}, \supseteq\}$
$C_2 = \{2, \text{Beca de Ayuda Económica}, \text{Beca de Servicio}, \perp\}$
$C_3 = \{3, \text{Beca de Ayuda Económica}, \text{Beca de Investigación y Desarrollo}, \perp\}$
$C_4 = \{4, \text{Postulante}, \text{Postulante a Beca de Servicio}, \supseteq\}$
$C_5 = \{5, \text{Postulante}, \text{Postulante a Beca de Investigación y Desarrollo}, \supseteq\}$

Finalmente, se procede a la integración de la ontología y el código de programa a fin de generar de una nueva versión del sistema de software. Al finalizar esta segunda iteración de desarrollo, los requerimientos funcionales implementados en código de programa deberían permitir interactuar con la ontología a fin de posibilitar (1) a los alumnos postularse a una Beca de Ayuda Económica, de Servicio o de Investigación y Desarrollo, y (2) al personal administrativo obtener automáticamente un ranking de postulantes de cada beca de acuerdo a la normativa definida por las autoridades de la universidad.

Capítulo 5

Conclusiones y Trabajos Futuros

Este capítulo presenta las conclusiones de la tesis. Se describen las contribuciones realizadas (Sección 5.1) y se presentan las futuras líneas de investigación (Sección 5.2).

5.1. Conclusiones

El objetivo de la investigación global del autor consiste en proveer un entorno completo para el desarrollo integrado de sistemas de software basados en ontologías, donde las ontologías encapsulan la especificación declarativa de la lógica del negocio, facilitan la representación del conocimiento sin ambigüedades, brindan servicios de razonamiento al sistema de software, y mejoran la gestión de entornos de alto dinamismo conceptual.
El marco de trabajo denominado ***PATRON*** satisface dicho objetivo mediante (1) la definición de un proceso de desarrollo de una ontología concebida para encapsular el conocimiento de un dominio dado y destinada a soportar un subconjunto de los requerimientos funcionales de un sistema de software, (2) la integración de la ontología y el código de programa del sistema antes mencionado, y (3) la rápida incorporación de las nuevas reglas de negocio en la ontología desarrollada. El proceso adopta un enfoque iterativo e incremental, desarrollando la ontología mediante la incorporación gradual de los nuevos requerimientos. Cada iteración concluye con el desarrollo de una nueva versión de la ontología y su posterior integración con el código de programa. Además de contextualizar el desarrollo de la ontología en el ámbito de un proceso de desarrollo de software,

la naturaleza cíclica de ***PATRON*** constituye un modo eficiente de abordar la dinámica conceptual. Por otro lado, la definición de un modelo de fases claramente definido facilita la evolución desde un modelo abstracto hasta una ontología implementada en un lenguaje computable.

El núcleo de ***PATRON*** se encuentra conformado por los patrones de diseño presentados en el Capítulo 2, los cuales permiten la obtención de una ontología OWL/SWRL mediante la aplicación de un conjunto de reglas de transformación de meta-modelo sobre la especificación SBVR de un dominio de negocio. La naturaleza reusable de los patrones propuestos, dada por su capacidad de ser utilizados en la solución de diversos problemas de modelado, facilita el proceso de desarrollo y mejora la calidad de la ontología resultante. Esto permite a los expertos del dominio obtener rápidamente una ontología desde la especificación en lenguaje natural de su conocimiento del negocio, y facilita la comunicación entre los diversos roles - expertos del dominio, ingenieros ontológicos e ingenieros de software - involucrados en la ejecución del proceso. Por otra parte, la definición ontológica de conceptos en un lenguaje formal es muy similar a la descripción de dichos conceptos por medio del lenguaje natural. Aplicar este principio en la etapa de conceptualización de una ontología constituye un mecanismo intuitivo de identificación de los conceptos ontológicos, el modo que se relacionan y las restricciones impuestas sobre dichas relaciones.

La disponibilidad de tecnologías que garanticen una ingeniería eficiente de ontologías de alta calidad constituye un requerimiento clave para la adopción de tecnologías semánticas en entornos reales. Esto requiere la realización de estudios empíricos de las metodologías, las técnicas y los enfoques propuestos en los ámbitos académicos. Tanto ***PATRON*** como los patrones de diseño presentados en esta tesis han sido evaluados experimentalmente en el contexto de la cátedra denominada *"Desarrollo de Sistemas de Información basados en Ontologías"*, el cual forma parte del último nivel de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional en la ciudad de Santa Fe, Argentina.

5.2. Trabajos futuros

Un trabajo futuro de corto plazo y naturaleza práctica consiste en la implementación de las transformaciones propuestas en un prototipo de software que posibilite la transformación automática de las especificaciones SBVR en una ontología OWL 2.

Un punto de partida interesante para trabajos futuros en el área es la exploración de diversas alternativas para la evaluación de calidad de los patrones de diseño. Una estrategia formal de validación resultaría un enfoque posible: por ejemplo, podría asegurarse equivalencia semántica entre la ontología producida y el conjunto original de sentencias SBVR mediante la definición de un co-morfismo entre las teorías lógicas subyacentes a ambos meta-modelos.

Dado que en este contexto las ontologías resultan un artefacto destinado a formar parte de la aplicación resultante, sus atributos de calidad impactan de manera directa en los indicadores de calidad del sistema desarrollado. En consecuencia, la evaluación aislada de la calidad de una ontología resulta parcial e incompleta en el contexto de un proceso de desarrollo de software. Dichas actividades de evaluación deben responder a un enfoque integral que contemple los indicadores de calidad de los dos principales artefactos de un ODIS: el software y la ontología. Luego, la determinación de la mejor combinación de características de calidad de una ontología a fin de optimizar los indicadores de calidad de los ODIS constituye una futura línea de investigación a seguir.

Otro aspecto interesante de abordar en trabajos futuros reside en las actividades de verificación ejecutadas durante el proceso de desarrollo del marco de trabajo propuesto. La utilización de un tipo particular de patrones de diseño proporcionaría un enfoque alternativo para la realización de estas actividades de verificación. Los patrones de diseño de ontologías denominados *"de Contenido"* brindan una solución probada a un conjunto específico de preguntas de competencia que representan el problema abordado. Dado que estos patrones resultan instanciaciones o composiciones de patrones lógicos, su utilización permitiría modelar el dominio mediante los patrones de re-ingeniería propuestos. Luego, la composición de tales patrones para la instanciación de los patrones de contenido conformarían la interfaz entre el modelado del dominio y la satisfacción de los usos previstos.

Una última línea de investigación se encuentra ligada a la integración de la ontología y el código de programa. La ejecución de dicha integración generalmente requiere de una interfaz de programación, la cual involucra la gestión de mapeos complejos entre los constructores ontológicos y las entidades involucradas en un paradigma de programación determinado (Carroll y otros, 2004)(Kalyanpur y otros, 2004)(Oren y otros, 2007)(Paar y Vrandecic, 2011)(Scheglmann y otros, 2013)(Leinberger y otros, 2014). Sin embargo, resultaría interesante la definición de un nuevo paradigma de programación donde los constructores fundamentales reflejen la semántica de las entidades del dominio modeladas ontológicamente.

Apéndice **A**

Patrones de reingeniería de esquemas: SBVR a OWL2

Las siguientes secciones presentan la especificación formal de los *patrones de reingeniería de esquemas* descriptos en el Capítulo 2. La especificación comprende la definición del objetivo del patrón, el constructor SBVR original y la expresión OWL 2/SWRL resultante, el proceso a ejecutar para su aplicación, y la ilustración del patrón aplicado mediante un ejemplo. El ejemplo utilizado se corresponde al presentado previamente en el Capítulo 2, el cual describe la organización estructural de una compañía ficticia (Figura A.1).

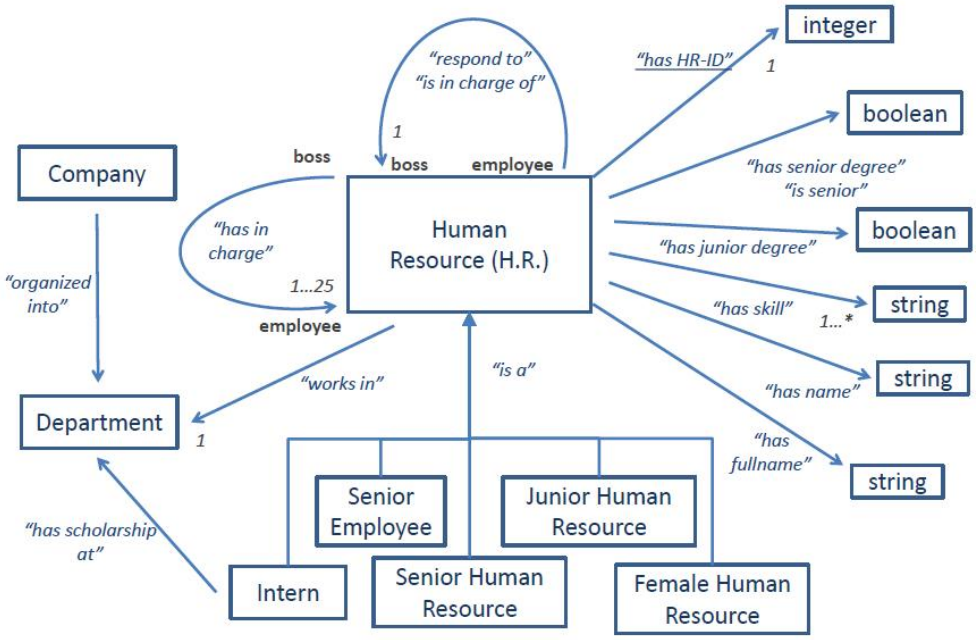

Figura A.1: Estructura general de la compañía

A.1. SBVR Concepto General

Nombre

SBVR Concepto General (General Concept) a OWL 2.

Objetivo

Expresar un concepto general SBVR mediante axiomas OWL 2.

Constructor SBVR

Un concepto general SBVR es un concepto sustantivo que clasifica las cosas en función de sus propiedades comunes.

Expresión OWL 2

Un concepto general SBVR es transformado en una clase OWL 2.

Proceso

- Paso 1. Crear una clase OWL 2 representando el concepto general SBVR.

  Declaration(Class(C))

Ejemplo

- Entrada del glosario SBVR

Human Resource
Tipo de concepto: Concepto General

- Expresión OWL 2
  Declaration(Class(:Human_Resource))

A.2. SBVR Concepto Unitario

Nombre

SBVR Concepto Unitario (Unitary Concept) a OWL 2.

Objetivo

Expresar un concepto unitario SBVR mediante axiomas OWL 2.

Constructor SBVR

Un concepto unitario SBVR es un concepto sustantivo que posee una instancia como máximo.

Expresión OWL 2

OWL 2 no posee un constructor para modelar directamente conceptos unitarios SBVR. Un concepto unitario SBVR es transformado a una clase OWL 2 y una sentencia SWRL estableciendo que si más de un individuo es definido para dicha clase, luego los individuos son en realidad los mismos. A continuación se presenta dicha sentencia:

$$Singleton(?x),\ Singleton(?y) \rightarrow SameAs\ (?x,\ ?y)$$

Proceso

- Paso 1. Crear una clase OWL 2 representando el concepto unitario SBVR.

  $$Declaration(Class(C))$$

- Paso 2. Crear una sentencia SWRL como la antes descripta.

  $$C(?x),\ C(?y) \rightarrow SameAs\ (?x,\ ?y)$$

Ejemplo

- Entrada del glosario SBVR

Company
Tipo de concepto: Concepto Unitario

- Expresión OWL 2/SWRL.
  OWL 2: Declaration(Class(:Company))
  SWRL: Company(?x), Company(?y) → SameAs (?x, ?y))

A.3. SBVR Rol de Concepto Verbal (Verb Concept Role)

Nombre

SBVR Rol de Concepto Verbal (Verb Concept Role) a OWL 2.

Objetivo

Expresar un rol de concepto verbal SBVR mediante axiomas OWL 2.

Constructor SBVR

Un rol SBVR es un concepto sustantivo determinado de acuerdo a la función que asume o la manera en que es utilizado en una situación dada. Un rol de concepto verbal SBVR es un rol que específicamente caracteriza sus instancias por su participación en una instancia de un concepto verbal determinado, incorporando aquellas características requeridas por el concepto verbal para las instancias del rol.

Expresión OWL 2

Un rol de concepto verbal SBVR es transformado a una clase OWL 2 definida en términos de la relación que le brinda significado al rol y el concepto general pasible de jugar dicho rol. El rol de concepto verbal es *reificado* como un clase OWL 2 definida en función de propiedades de objeto. La primera de ellas representa el concepto verbal en el cual el rol participa. La segunda establece el concepto general que juega el rol definido.

Proceso

- Paso 1. Crear una clase OWL 2 representando el rol de concepto verbal SBVR.

  $Declaration(Class(C_{rcb}))$

- Paso 2. Crear una propiedad de objeto relacionando la clase OWL 2 que representa el rol de concepto verbal con la clase OWL que representa el concepto general que juega dicho rol.

  $Declaration(Class(C_{cg}))$

  $Declaration(ObjectProperty(OPE_1))$

  $ObjectPropertyDomain(OPE_1\ C_{rcb})$

  $ObjectPropertyRange(OPE_1\ C_{cg})$

- Paso 3. Crear una propiedad de objeto relacionando la clase OWL 2 que representa el rol de concepto verbal con la clase OWL 2 que le da sentido al rol representado.

  $Declaration(Class(C_{rol}))$

Declaration(ObjectProperty(OPE_2))

ObjectPropertyDomain(OPE_2 C_{rcb})

ObjectPropertyRange(OPE_2 C_{rol})

Ejemplo

- Entrada del glosario SBVR

Employee is in charge of Boss	
Tipo de concepto:	Concepto Verbal Binario

Boss has in charge Employee	
Tipo de concepto:	Concepto Verbal Binario

Employee	
Definición:	Human Resource who is in charge of Boss
Tipo de concepto:	Rol de Concepto Verbal

Boss	
Definición:	Human Resource who has in charge Employee
Tipo de concepto:	Rol de Concepto Verbal

- Expresión OWL 2

Declaration(Class(:Boss))
Declaration(Class(:Employee))
Declaration(Class(:Human_Resource))
Declaration(ObjectProperty(:Boss_role_of))
ObjectPropertyDomain(:Boss_role_of :Boss)
ObjectPropertyRange(:Boss_role_of :Human_Resource)
Declaration(ObjectProperty(:Employee_role_of))
ObjectPropertyDomain(:Employee_role_of :Employee)
ObjectPropertyRange(:Employee_role_of :Human_Resource)

Declaration(ObjectProperty(:has_in_charge))
ObjectPropertyDomain(:has_in_charge :Boss)
ObjectPropertyRange(:has_in_charge :Employee)
Declaration(ObjectProperty(:is_in_charge_of))
ObjectPropertyDomain(:is_in_charge_of :Employee)
ObjectPropertyRange(:is_in_charge_of :Boss)

A.4. SBVR Concepto Verbal Binario (Binary Verb Concept)

Nombre

SBVR Rol de Concepto Verbal Binario (Binary Verb Concept) a OWL 2.

Objetivo

Expresar un concepto verbal binario SBVR mediante axiomas OWL 2.

Constructor SBVR

Un concepto verbal binario SBVR es un concepto verbal que tiene exactamente roles.

Expresión OWL 2

La transformación de un concepto verbal binario en una sentencia OWL 2 es realizada de acuerdo al tipo de los roles involucrados. Si el concepto verbal relaciona conceptos, la expresión SBVR se transforma en una propiedad de objeto OWL 2. Si el concepto verbal relaciona un concepto con un literal, la expresión SBVR se transforma en una propiedad de datos OWL 2.

Proceso

Si el concepto verbal relaciona conceptos:

- Paso 1. Crear una propiedad de objetos OWL relacionando ambos conceptos.

  Declaration(Class(CE_1))

  Declaration(Class(CE_2))

  Declaration(ObjectProperty(OPE))

ObjectPropertyDomain(OPE CE_1)

ObjectPropertyRange(OPE CE_2)

Si el concepto verbal relaciona un concepto con un literal:

- Paso 1. Crear una propiedad de datos OWL relacionando el concepto y el literal.

  Declaration(Class(CE_1))

  Declaration(DataProperty(DPE))

  DataPropertyDomain(DPE CE_1)

  DataPropertyRange(DPE DR)

Ejemplo

Si el concepto verbal relaciona conceptos:

- Entrada del glosario SBVR

Employee is in charge of Boss	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Boss))
  Declaration(Class(:Employee))
  Declaration(ObjectProperty(:is_in_charge_of))
  ObjectPropertyDomain(:is_in_charge_of :Employee)
  ObjectPropertyRange(:is_in_charge_of :Boss)

Si el concepto verbal relaciona un concepto y un literal:

- Entrada del glosario SBVR

Human Resource has skill	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_skill))
  DataPropertyDomain(:has_skill :Human_Resource)
  DataPropertyRange(:has_skill xsd:string)

A.5. SBVR Característica/Concepto Verbal Unario (Characteristic/Unary Verb Concept)

Nombre

SBVR Característica/Concepto Verbal Unario (Characteristic/Unary Verb Concept)) a OWL 2.

Objetivo

Expresar una característica/concepto verbal unario SBVR mediante axiomas OWL 2.

Constructor SBVR

Una característica/concepto verbal unario SBVR es un concepto verbal que tiene exactamente un rol, una abstracción de una propiedad de un objeto o de un conjunto de objetos.

Expresión OWL 2

Una característica/concepto verbal unario SBVR se transforma en una propiedad de datos OWL 2 con el tipo de datos booleano como rango.

Proceso

- Paso 1. Crear una propiedad de datos OWL relacionando el concepto y el literal de tipo booleano.

  Declaration(Class(CE_1))

  Declaration(DataProperty(DPE))

  DataPropertyDomain(DPE CE_1)

  DataPropertyRange(DPE xsd:boolean)

Ejemplo

- Entrada del glosario SBVR

Human_Resource has senior degree	
Tipo de concepto:	Característica/Concepto Verbal Unario

- Expresión OWL 2

  Declaration(Class(:Human_Resource))

  Declaration(DataProperty(:has_senior_degree))

  DataPropertyDomain(:has_senior_degree :Human_Resource)

  DataPropertyRange(:has_senior_degree xsd:boolean)

A.6. SBVR Concepto Individual (Individual Concept)

Nombre

SBVR Concepto Individual (Individual Concept) a OWL 2.

Objetivo

Expresar un concepto individual SBVR mediante axiomas OWL 2.

Constructor SBVR

Un concepto individual SBVR es unconcepto que corresponde a un único objeto.

Expresión OWL 2

Un concepto individual SBVR es transformado en un individuo nombrado OWL 2.

Proceso

- Paso 1. Crear un individuo nombrado representando el concepto individual.

  Declaration(NamedIndividual(I))

Ejemplo

- Entrada del glosario SBVR

John Smith	
Tipo de concepto:	Concepto Individual

- Expresión OWL 2
  Declaration(NamedIndividual(:John_Smith))

Notas adicionales

Los individuos generados por la aplicación de este patrón no pertenecen a ninguna clase OWL 2 en particular. Corresponden a instancias de una clase OWL 2 predefinida denominada *'Thing'*, la cual engloba todos los individuos del dominio descripto. La asignación de un individuo a un clase determinada requiere de una sentencia de clasificación SBVR.

A.7. SBVR Clasificación (Classification)

Nombre

SBVR Clasificación (Classification) a OWL 2.

Objetivo

Expresar una clasificación SBVR mediante axiomas OWL 2.

Constructor SBVR

Una clasificación SBVR establece que un concepto individual determinado es una instancia de un concepto general dado.

Expresión OWL 2

Una clasificación SBVR se transforma en un axioma de aserción de clase OWL 2 relacionando el individuo con la clase del cual es instancia.

Proceso

- Paso 1. Crear una axioma de aserción de clase OWL 2 relacionando el individuo con la clase del cual es instancia.

  Declaration(NamedIndividual(I))

  ClassAssertion(CE I)

Ejemplo

- Entrada del glosario SBVR

John Smith	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual

- Expresión OWL 2
  Declaration(NamedIndividual(:John_Smith))
  ClassAssertion(:Human_Resource :John_Smith)

A.8. SBVR Formulación Atómica (Atomic Formulation)

Nombre

SBVR Formulación Atómica (Atomic Formulation) a OWL 2.

Objetivo

Expresar una formulación atómica SBVR mediante axiomas OWL 2.

Constructor SBVR

Una formulación atómica SBVR es una formulación atómica que se encuentra basada en un concepto verbal donde cada uno de sus roles se encuentra instanciado.

Expresión OWL 2

La transformación de una formulación atómica en una sentencia OWL 2 considera el tipo de los roles involucrados. Si la formulación atómica se basa en un concepto verbal que vincula conceptos la expresión SBVR se transforma en una aserción positiva de propiedad de objeto OWL 2. Si la formulación atómica se basa en un concepto verbal que vincula un concepto con un literal, la expresión SBVR se transforma en una aserción positiva de propiedad de datos OWL 2.

Proceso

Si la formulación atómica se basa en un concepto verbal que vincula conceptos:

- Paso 1. Crear aserción positiva de propiedad de objeto OWL relacionando ambos conceptos.

  Declaration(NamedIndividual(I_1))

ClassAssertion(CE_1 I_1)

Declaration(NamedIndividual(I_2))

ClassAssertion(CE_2 I_2)

Declaration(ObjectProperty(OPE))

ObjectPropertyDomain(OPE CE_1)

ObjectPropertyRange(OPE CE_2)

ObjectPropertyAssertion(OPE I_1 I_2)

Si la formulación atómica se basa en un concepto verbal que vincula un concepto con un literal:

- Paso 1. Crear una aserción positiva de propiedad de datos OWL relacionando el concepto y el literal.

  Declaration(NamedIndividual(I_1))

  ClassAssertion(CE_1 I_1)

  Declaration(DataProperty(DPE))

  DataPropertyDomain(DPE CE_1)

  DataPropertyRange(OPE DR)

  DataPropertyAssertion(OPE I_1 "thing"^ ^xsd:DR)

Ejemplo

Si la formulación atómica se basa en un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

John Smith	
Concepto padre:	Employee
Tipo de concepto:	Concepto Individual

Ben Arten	
Concepto padre:	Boss
Tipo de concepto:	Concepto Individual

Employee is in charge of Boss	
Tipo de concepto:	Concepto Verbal Binario

John Smith is in charge of Ben Arten	
Tipo de concepto:	Formulación Atómica

- Expresión OWL 2

  Declaration(NamedIndividual(:Ben_Arten))
  ClassAssertion(:Boss :Ben_Arten)
  Declaration(NamedIndividual(:John_Smith))
  ClassAssertion(:Employee :John_Smith)
  Declaration(ObjectProperty(:is_in_charge_of))
  ObjectPropertyDomain(:is_in_charge_of :Employee)
  ObjectPropertyRange(:is_in_charge_of :Boss)
  ObjectPropertyAssertion(:is_in_charge_of :John_Smith :Ben_Arten)

Si la formulación atómica se basa en un concepto verbal que vincula un concepto con un literal:

- Entrada del glosario SBVR

John Smith	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual

Human Resource has skill	
Tipo de concepto:	Concepto Verbal Binario

John Smith has mechanical engineering skill	
Tipo de concepto:	Formulación Atómica

- Expresión OWL 2
  Declaration(NamedIndividual(:John_Smith))
  ClassAssertion(:Human_Resource :John_Smith)
  Declaration(DataProperty(:has_skill))
  DataPropertyDomain(:has_skill :Human_Resource)
  DataPropertyRange(:has_skill xsd:string)
  *DataPropertyAssertion(:has_skill :John_Smith "mechanical_engineering"^*
  ^xsd:string)

A.9. SBVR Esquema de Referencia (Reference Scheme)

Nombre

SBVR Esquema de Referencia (Reference Scheme)a OWL 2.

Objetivo

Expresar un esquema de referencia SBVR mediante axiomas OWL 2.

Constructor SBVR

Un esquema de referencia SBVR es un medio de identificar conceptos individuales por medio de sus atributos, ya sean literales u objetos. SBVR permite utilizar como esquema de referencia uno o más roles de concepto verbales binarios y/o una o más características.

Expresión OWL 2

Un esquema de referencia SBVR es transformado en un axioma de identificación OWL 2, la cual establece que cada individuo de una clase determinada es identificado unívocamente por un conjunto de propiedades de objeto y/o propiedades de datos. Es decir, no existen dos instancias de una clase dada que posean los mismos valores de todas las propiedades de objetos y de todas las propiedades de datos.

Proceso

- Paso 1. Crear un axioma de identificación OWL 2 representando el esquema de referencia SBVR.

  Declaration(Class(CE))

  HasKey(CE (OPE$_1$ OPE$_2$ ... OPE$_n$) (DPE$_1$ DPE$_2$ ... DPE$_m$))

Ejemplo

- Entrada del glosario SBVR

Human Resource has HR-ID	
Tipo de concepto:	Concepto Verbal Binario

Human Resource works in Department	
Tipo de concepto:	Concepto Verbal Binario

Human Resource	
Tipo de concepto:	Concepto General
Esquema de Ref.:	has HR-ID; works in Department

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(Class(:Department))
  Declaration(ObjectProperty(:works_in))
  ObjectPropertyDomain(:works_in :Human_Resource)
  ObjectPropertyRange(:works_in :Department)
  Declaration(DataProperty(:has_HR-ID))
  DataPropertyDomain(:has_HR-ID :Human_Resource)
  DataPropertyRange(:has_HR-ID xsd:integer)
  HasKey(:Human_Resource (:works_in) (:has_HR-ID))

A.10. SBVR Negación Lógica (Logical Negation)

Nombre

SBVR Negación Lógica (Logical Negation) a OWL 2.

Objetivo

Expresar una negación lógica SBVR mediante axiomas OWL 2.

Constructor SBVR

Una negación lógica SBVR es una operación lógica que tiene exactamente un operando lógico cuyo significado es falso.

Expresión OWL 2

Una negación lógica SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si el operando lógico es un concepto verbal la expresión se transforma en una expresión de complemento de clase OWL 2. Si el operando lógico es una formulación atómica vinculando individuos la expresión se transforma en una aserción negativa de propiedad de objeto OWL 2. Si el operando lógico es una formulación atómica vinculando un individuo con un literal la expresión se transforma en una aserción negativa de propiedad de datos OWL 2.

Proceso

Si el operando lógico es un concepto verbal:

- Paso 1. Crear expresión de complemento de clase OWL 2.

  ObjectComplementOf(CE)

Si el operando lógico es una formulación atómica vinculando individuos:

- Paso 1. Crear aserción negativa de propiedad de objeto OWL 2.

  NegativeObjectPropertyAssertion(OPE I_1 I_2*)*

Si el operando lógico es una formulación atómica vinculando un individuo con un literal:

- Paso 1. Crear aserción negativa de propiedad de datos OWL 2.

  NegativeDataPropertyAssertion(DPE I DR)

Ejemplo

Si el operando lógico es un concepto verbal:

- Entrada del glosario SBVR

Intern	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General
Necesidad:	Intern has not senior degree

Human_Resource has senior degree	
Tipo de concepto:	Característica/Concepto Verbal Unario

- Expresión OWL 2

  Declaration(Class(:Intern))
  SubClassOf(:Intern :Human_Resource)
  Declaration(DataProperty(:has_senior_degree))
  DataPropertyDomain(:has_senior_degree :Human_Resource)
  DataPropertyRange(:has_senior_degree xsd:boolean)
  SubClassOf(:Intern ObjectComplementOf(DataHasValue(:has_senior_degree "true"^ ^xsd:boolean)))

Si el operando lógico es una formulación atómica vinculando individuos:

- Entrada del glosario SBVR

Peter Tomsom	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual
Necesidad:	Peter Tomsom is not in charge of Ben Arten

Ben Arten	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual

Employee is in charge of Boss	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(NamedIndividual(:Peter_Tomsom))
  ClassAssertion(:Human_Resource :Peter_Tomsom)
  NegativeObjectPropertyAssertion(:is_in_charge_of :Peter_Tomsom :Ben_Arten)
  Declaration(NamedIndividual(:Ben_Arten))
  ClassAssertion(:Human_Resource :Ben_Arten)
  Declaration(ObjectProperty(:is_in_charge_of))
  ObjectPropertyDomain(:is_in_charge_of :Employee)
  ObjectPropertyRange(:is_in_charge_of :Boss)

Si el operando lógico es una formulación atómica vinculando un individuo con un literal:

- Entrada del glosario SBVR

Peter Tomsom	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual
Necesidad:	Peter Tomsom has not senior degree

Human_Resource has senior degree	
Tipo de concepto:	Característica/Concepto Verbal Unario

- Expresión OWL 2

  Declaration(NamedIndividual(:Peter_Tomsom))
  ClassAssertion(:Human_Resource :Peter_Tomsom)
  Declaration(DataProperty(:has_senior_degree))
  DataPropertyDomain(:has_senior_degree :Human_Resource)
  DataPropertyRange(:has_senior_degree xsd:boolean)
  NegativeDataPropertyAssertion(:has_senior_degree :Peter_Tomsom "true"^^xsd:boolean)

A.11. SBVR Conjunción (Conjunction)

Nombre

SBVR Conjunción (Conjunction) a OWL 2.

Objetivo

Expresar una conjunción SBVR mediante axiomas OWL 2.

Constructor SBVR

Una conjunción SBVR es una operación lógica binaria donde el significado de cada uno de sus operandos es verdadero.

Expresión OWL 2

Una conjunción SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si la conjunción se aplica sobre conceptos verbales la expresión se transforma en una sentencia OWL 2 de intersección de expresiones de clases. Se aplica la misma transformación si la conjunción se formula sobre conceptos generales. Si la conjunción se aplica sobre tipos de datos la expresión se transforma en una expresión OWL 2 de intersección de rangos de datos.

Proceso

Si la conjunción se aplica sobre conceptos verbales:

- Paso 1. Crear sentencia de intersección de expresiones de clases OWL 2.

  $$ObjectIntersectionOf(CE_1\ CE_2\ \ldots\ CE_n)$$

Si la conjunción se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de intersección de expresiones de clases OWL 2.

  $$ObjectIntersectionOf(CE_1\ CE_2\ \ldots\ CE_n)$$

Si la conjunción se aplica sobre tipos de datos:

- Paso 1. Crear sentencia de intersección de expresiones de rangos de datos OWL 2.

  $$\text{DataIntersectionOf}(\text{DR}_1\ \text{DR}_2\ \ldots\ \text{DR}_n)$$

Ejemplo

Si la conjunción se aplica sobre conceptos verbales:

- Entrada del glosario SBVR

Senior Employee	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General
Necesidad:	has senior degree and is in charge of Boss

- Expresión OWL 2

Declaration(Class(:Senior_Employee))
SubClassOf(:Senior_Employee :Human_Resource)
SubClassOf(:Senior_Employee ObjectIntersectionOf(
DataHasValue(:has_senior_degree "true"^^ xsd:boolean)
ObjectSomeValuesFrom(:is_in_charge_of :Boss)))
Declaration(DataProperty(:has_senior_degree))
DataPropertyDomain(:has_senior_degree :Human_Resource)
DataPropertyRange(:has_senior_degree xsd:boolean)
Declaration(ObjectProperty(:is_in_charge_of))
ObjectPropertyDomain(:is_in_charge_of :Employee)
ObjectPropertyRange(:is_in_charge_of :Boss)

Si la conjunción se aplica sobre conceptos generales:

- Entrada del glosario SBVR

Female Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	Female Person and Human Resource

Female Person	
Tipo de concepto:	Concepto General

Human Resource	
Tipo de concepto:	Concepto General

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(Class(:Female_Person))
  Declaration(Class(:Female_Human_Resource))
  SubClassOf(:Female_Human_Resource ObjectIntersectionOf(:Female_Person :Human_Resource))

Si la conjunción se aplica sobre tipos de datos:

- Entrada del glosario SBVR

Zero	
Necesidad:	nonNegativeInteger and nonPositiveInteger

- Expresión OWL 2
  Declaration(Datatype(:Zero))
  DatatypeDefinition(:Zero DataIntersectionOf(xsd:nonPositiveInteger xsd:nonNegativeInteger))

A.12. SBVR Disyunción (Disjunction)

Nombre

SBVR Disyunción (Disjunction) a OWL 2.

Objetivo

Expresar una disyunción SBVR mediante axiomas OWL 2.

Constructor SBVR

Una disyunción SBVR es una operación lógica binaria donde el significado de al menos uno de sus operandos es verdadero.

Expresión OWL 2

Una disyunción SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si la disyunción SBVR se aplica sobre conceptos verbales la expresión se transforma en una sentencia OWL 2 de unión de expresiones de clases. Se aplica la misma transformación si la disyunción se formula sobre conceptos generales. Si la disyunción se aplica sobre tipos de datos

la expresión se transforma en una expresión OWL 2 de unión de rangos de datos.

Proceso

Si la disyunción se aplica sobre conceptos verbales:

- Paso 1. Crear sentencia de unión de expresiones de clases OWL 2.

  ObjectUnionOf(CE_1 CE_2 ... CE_n)

Si la disyunción se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de unión de expresiones de clases OWL 2.

  ObjectUnionOf(CE_1 CE_2 ... CE_n)

Si la disyunción se aplica sobre tipos de datos:

- Paso 1. Crear sentencia de unión de expresiones de rangos de datos OWL 2.

  DataUnionOf(DR_1 DR_2 ... DR_n)

Ejemplo

Si la disyunción se aplica sobre conceptos verbales:

- Entrada del glosario SBVR

Employee	
Tipo de concepto:	Concepto General
Necesidad:	has senior degree or has mechanical engineering skill

- Expresión OWL 2
  Declaration(Class(:Employee))
  SubClassOf(:Employee ObjectUnionOf(DataHasValue(:has_skill "mechanical_engineering"^ ^xsd:string) DataHasValue(:has_senior_degree "true"^ ^xsd:boolean)))

Si la disyunción se aplica sobre conceptos generales:

- Entrada del glosario SBVR

Person	
Tipo de concepto:	Concepto General
Necesidad:	Female Person or Male Person

- Expresión OWL 2
  Declaration(Class(:Person))
  SubClassOf(:Person ObjectUnionOf(:Male_Person :Female_Person))
  Declaration(Class(:Female_Person))
  Declaration(Class(:Male_Person))

Si la disyunción se aplica sobre tipos de datos:

- Entrada del glosario SBVR

Integer	
Necesidad:	nonNegativeInteger or nonPositiveInteger

- Expresión OWL 2
  Declaration(Datatype(:Integer))
  DatatypeDefinition(:Integer DataUnionOf(xsd:nonPositiveInteger xsd:nonNegativeInteger))

A.13. SBVR Equivalencia (Equivalence)

Nombre

SBVR Equivalencia (Equivalence) a OWL 2.

Objetivo

Expresar una equivalencia lógica SBVR mediante axiomas OWL 2.

Constructor SBVR

Una equivalencia SBVR es una operación lógica binaria donde el significado de sus operandos son todos verdaderos o todos falsos.

Expresión OWL 2

Una equivalencia SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si la equivalencia se aplica sobre conceptos generales la expresión se transforma en una sentencia de equivalencia de clases OWL 2. Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos la expresión se transforma en una sentencia de equivalencia de propiedades de objeto OWL 2. Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos con literales la expresión se transforma en una sentencia de equivalencia de propiedades de datos OWL 2. Si la equivalencia se aplica sobre individuos la expresión se transforma en una sentencia de igualdad de individuos OWL 2.

Proceso

Si la equivalencia se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de equivalencia de clases OWL 2.

  EquivalentClasses(CE_1 CE_2 ... CE_n)

Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos:

- Paso 1. Crear sentencia de equivalencia de propiedades de objeto OWL 2.

  EquivalentObjectProperties(OPE_1 OPE_2 ... OPE_n)

Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos con literales:

- Paso 1. Crear sentencia de equivalencia de propiedades de datos OWL 2.

  EquivalentDataProperties(DPE_1 DPE_2 ... DPE_n)

Si la equivalencia se aplica sobre individuos:

- Paso 1. Crear sentencia de igualdad de individuos OWL 2.

  SameIndividual(I_1 I_2 ... I_n)

Ejemplo

Si la equivalencia se aplica sobre conceptos generales:

- Entrada del glosario SBVR

H.R.	
Necesidad:	is equivalent to Human Resource

- Expresión OWL 2
  Declaration(Class(:H.R.))
  Declaration(Class(:Human_Resource)
  EquivalentClasses(:Human_Resource :H.R.)

Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos:

- Entrada del glosario SBVR

respond to	
Tipo de Concepto:	Concepto Verbal Binario
Necesidad:	is equivalent to is in charge of

- Expresión OWL 2
  Declaration(ObjectProperty(:respond_to))
  EquivalentObjectProperties(:respond_to :is_in_charge_of)

Si la equivalencia se aplica sobre conceptos verbales que vinculan conceptos con literales:

- Entrada del glosario SBVR

is senior	
Tipo de concepto:	Concepto Verbal Unario
Necesidad:	is equivalent to has senior degree

- Expresión OWL 2
  Declaration(DataProperty(:is_senior))
  EquivalentDataProperties(:is_senior :has_senior_degree)

Si la equivalencia se aplica sobre individuos:

- Entrada del glosario SBVR

J.S	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual
Necesidad:	is equivalent John Smith

- Expresión OWL 2
  Declaration(NamedIndividual(:John_Smith))
  ClassAssertion(:Human_Resource :John_Smith)
  SameIndividual(:J.S. :John_Smith)

A.14. SBVR Disyunción Exclusiva (Exclusive Disjunction)

Nombre

SBVR Disyunción Exclusiva (Exclusive Disjunction) a OWL 2.

Objetivo

Expresar una disyunción exclusiva SBVR mediante axiomas OWL 2.

Constructor SBVR

Una disyunción exclusiva es una operación lógica binaria donde el significado de uno de sus operandos es verdadero y el otro es falso.

Expresión OWL 2

Una disyunción exclusiva SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si la disyunción exclusiva SBVR se aplica sobre conceptos generales la expresión se transforma en una sentencia de disyunción exclusiva de clases OWL 2. Si la disyunción exclusiva SBVR se aplica sobre conceptos verbales que vinculan conceptos la expresión se transforma en una sentencia de disyunción exclusiva de propiedades de objeto OWL 2. Si la disyunción exclusiva SBVR se aplica sobre conceptos verbales vinculando conceptos con literales la expresión se transforma en una sentencia de disyunción exclusiva de propiedades de datos OWL 2.

Proceso

Si la disyunción exclusiva SBVR se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de disyunción exclusiva de expresiones de clase OWL 2.

$$DisjointClasses(CE_1\ CE_2\ ...\ CE_n)$$

Si la disyunción exclusiva SBVR se aplica sobre conceptos verbales que vinculan conceptos:

- Paso 1. Crear sentencia de disyunción exclusiva de propiedades de objeto OWL 2.

$$DisjointObjectProperties(OPE_1\ OPE_2\ ...\ OPE_n)$$

Si la disyunción exclusiva se aplica sobre conceptos verbales que vinculan conceptos con literales:

- Paso 1. Crear sentencia de disyunción exclusiva de propiedades de datos OWL 2.

$$DisjointDataProperties(DPE_1\ DPE_2\ ...\ DPE_n)$$

Ejemplo

Si la disyunción exclusiva se aplica sobre conceptos generales:

- Entrada del glosario SBVR

Female Person	
Tipo de concepto:	Concepto General
Necesidad:	Female Person xor Male Person

Male Person	
Tipo de concepto:	Concepto General
Definición:	Female Person xor Male Person

- Expresión OWL 2

Declaration(Class(:Male_Person))

Declaration(Class(:Female_Person))

DisjointClasses(:Male_Person :Female_Person)

Si la disyunción exclusiva SBVR se aplica sobre conceptos verbales que vinculan conceptos:

- Entrada del glosario SBVR

is in charge of	
Definición:	is in charge of xor has in charge
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(ObjectProperty(:is_in_charge_of))
  ObjectPropertyDomain(:is_in_charge_of :Employee)
  ObjectPropertyRange(:is_in_charge_of :Boss)
  Declaration(ObjectProperty(:has_in_charge))
  ObjectPropertyDomain(:has_in_charge :Boss)
  ObjectPropertyRange(:has_in_charge :Employee)
  DisjointObjectProperties(:is_in_charge_of :has_in_charge)

Si la disyunción exclusiva se aplica sobre conceptos verbales que vinculan conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has senior degree xor has junior degree

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_senior_degree))
  DataPropertyDomain(:has_senior_degree :Human_Resource)
  DataPropertyRange(:has_senior_degree xsd:boolean)
  Declaration(DataProperty(:has_junior_degree))
  DataPropertyDomain(:has_junior_degree :Human_Resource)
  DataPropertyRange(:has_junior_degree xsd:boolean)
  DisjointDataProperties(:has_senior_degree :has_junior_degree)

A.15. SBVR Conjunción Negada (Nand)

Nombre

SBVR Conjunción Negada (nand) a OWL 2.

Objetivo

Expresar una conjunción negada SBVR mediante axiomas OWL 2.

Constructor SBVR

Una conjunción negada es una operación lógica binaria donde el significado de al menos uno de sus operandos es falso.

Expresión OWL 2

Una conjunción negada SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si la conjunción negada se aplica sobre conceptos verbales la expresión se transforma en una sentencia de complemento de intersección de expresiones de clases OWL 2. Se aplica la misma transformación si la conjunción se formula sobre conceptos generales. Si la conjunción negada se aplica sobre tipos de datos la expresión se transforma en una sentencia de complemento de intersección de rangos de datos OWL 2.

Proceso

Si la conjunción negada se aplica sobre conceptos verbales:

- Paso 1. Crear sentencia de complemento de intersección de clases OWL 2.

  $$ObjectComplementOf(ObjectIntersectionOf(CE_1\ CE_2\ ...\ CE_n)$$

Si la conjunción negada se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de complemento de intersección de clases OWL 2.

  $$ObjectComplementOf(ObjectIntersectionOf(CE_1\ CE_2\ ...\ CE_n)$$

Si la conjunción negada se aplica sobre tipos de datos:

- Paso 1. Crear sentencia de complemento de intersección de rangos de datos OWL 2.

  $$DataComplementOf(DataIntersectionOf(DR_1\ DR_2\ ...\ DR_n)$$

Ejemplo

Si la conjunción negada se aplica sobre conceptos verbales:

- Entrada del glosario SBVR

Intern	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General
Necesidad:	has senior degree nand has in charge of Employee

- Expresión OWL 2

Declaration(Class(:Intern))
SubClassOf(:Intern :Human_Resource)
SubClassOf(:Intern ObjectComplementOf(ObjectIntersectionOf(
DataHasValue(:has_senior_degree "true"^ ^xsd:boolean)
ObjectSomeValuesFrom(:has_in_charge :Employee))))

Si la conjunción negada se aplica sobre conceptos generales:

- Entrada del glosario SBVR

Intern	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General
Necesidad:	Senior Employee nand Boss

- Expresión OWL 2

Declaration(Class(:Intern))
SubClassOf(:Intern :Human_Resource)
Declaration(Class(:Female_Human_Resource)) *SubClassOf(:Intern*
ObjectComplementOf(ObjectIntersectionOf(:Senior_Employee :Boss)))

Si la conjunción negada se aplica sobre tipos de datos:

- Entrada del glosario SBVR

PositiveOrNegativeInteger	
Necesidad:	nonNegativeInteger nand nonPositiveInteger

- Expresión OWL 2
  Declaration(Datatype(:PositiveOrNegativeInteger))
  DatatypeDefinition(:PositiveOrNegativeInteger
  DataComplementOf(DataIntersectionOf(xsd:nonPositiveInteger
  xsd:nonNegativeInteger)))

A.16. SBVR Disyunción Negada (Nor)

Nombre

SBVR Disyunción Negada (nor) a OWL 2.

Objetivo

Expresar una disyunción negada SBVR mediante axiomas OWL 2.

Constructor SBVR

Una disyunción negada SBVR es una operación lógica binaria donde el significado de todos sus operandos es falso.

Expresión OWL 2

Una disyunción negada SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de los operandos involucrados. Si la disyunción negada SBVR se aplica sobre conceptos verbales la expresión se transforma en una sentencia de complemento de unión de expresiones de clases OWL 2. Se aplica la misma transformación si la disyunción negada se formula sobre conceptos generales. Si la disyunción negada se aplica sobre tipos de datos la expresión se transforma en una sentencia de complemento de unión de rangos de datos OWL 2.

Proceso

Si la disyunción negada se aplica sobre conceptos verbales:

- Paso 1. Crear sentencia de complemento de unión de expresiones de clases OWL 2.

$$ObjectComplementOf(ObjectUnionOf(CE_1\ CE_2\ \ldots\ CE_n)$$

Si la disyunción negada se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de complemento de unión de expresiones de clases OWL 2.

$$ObjectComplementOf(ObjectUnionOf(CE_1\ CE_2\ \ldots\ CE_n)$$

Si la disyunción negada se aplica sobre tipos de datos:

- Paso 1. Crear sentencia de complemento de unión de rangos de datos OWL 2.

$$DataComplementOf(DataUnionOf(CE_1\ CE_2\ \ldots\ CE_n)$$

Ejemplo

Si la disyunción negada se aplica sobre conceptos verbales:

- Entrada del glosario SBVR

Intern	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General
Necesidad:	has senior degree nor has mechanical engineering skill

- Expresión OWL 2
  Declaration(Class(:Intern))
  SubClassOf(:Intern :Human_Resource)
  SubClassOf(:Intern ObjectComplementOf(ObjectUnionOf(
  DataHasValue(:has_skill "mechanical_engineering"^ ^xsd:string)
  DataHasValue(:has_senior_degree "true"^ ^xsd:boolean))))

Si la disyunción negada se aplica sobre conceptos generales:

- Entrada del glosario SBVR

Boss	
Necesidad:	Intern nor Employee

- Expresión OWL 2
  Declaration(Class(:Boss))
  SubClassOf(:Boss ObjectComplementOf(ObjectUnionOf(:Intern :Employee)))

Si la disyunción negada se aplica sobre tipos de datos:

- Entrada del glosario SBVR

notNumber	
Necesidad:	nonNegativeInteger nor nonPositiveInteger

- Expresión OWL 2
  Declaration(Datatype(:notNumber)) *DatatypeDefinition(:notNumber* *DataComplementOf(DataUnionOf(xsd:nonPositiveInteger* *xsd:nonNegativeInteger)))*

A.17. SBVR Cuantificación Universal (Universal Quantification)

Nombre

Cuantificación Universal (Universal Quantification) a OWL 2.

Objetivo

Expresar una cuantificación universal SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación universal SBVR es una cuantificación cuyo ámbito es una formulación lógica con el siguiente significado: para cada referente de la variable introducida por la cuantificación, el significado de la formulación lógica para el referente es verdadero.

Expresión OWL 2

Una cuantificación universal SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación universal de objetos OWL 2. Si la cuantificación se aplica sobre un concepto verbal vinculando conceptos con literales la expresión se transforma en una sentencia de cuantificación universal de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación universal de objetos OWL 2.

  ObjectAllValuesFrom(OPE CE)

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación universal de literales OWL 2.

  DataAllValuesFrom(DPE DR)

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	works in a given Department

Department	
Tipo de concepto:	Concepto General

Human Resource works in Department	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(Class(:Department))
  Declaration(ObjectProperty(:works_in))
  ObjectPropertyDomain(:works_in :Human_Resource)
  ObjectPropertyRange(:works_in :Department)
  SubClassOf(:Human_Resource ObjectAllValuesFrom(:works_in :Department))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has a given skill

Human Resource has skill	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_skill))
  DataPropertyDomain(:has_skill :Human_Resource)
  DataPropertyRange(:has_skill xsd:string)) SubClassOf(:Human_Resource
  DataAllValuesFrom(:has_skill xsd:string))

A.18. SBVR Cuantificación Existencial (Existential Quantification)

Nombre

Cuantificación Existencial (Existential Quantification) a OWL 2.

Objetivo

Expresar una cuantificación existencial SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación existencial SBVR es una cuantificación cuyo ámbito es una formulación lógica con el siguiente significado: para al menos un referente de la variable introducida por la cuantificación, el significado de la formulación lógica para el referente es verdadero.

Expresión OWL 2

Una cuantificación existencial SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación existencial de objetos OWL 2. Si la cuantificación se

aplica sobre un concepto verbal vinculando conceptos con literales la expresión se transforma en una sentencia de cuantificación existencial de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación existencial de objetos OWL 2.

  ObjectSomeValuesFrom(OPE CE)

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación existencial de literales OWL 2.

  DataSomeValuesFrom(DPE DR)

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	works in some Department

Department	
Tipo de concepto:	Concepto General

Human Resource works in Department	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(Class(:Department))
  Declaration(ObjectProperty(:works_in))
  ObjectPropertyDomain(:works_in :Human_Resource)

ObjectPropertyRange(:works_in :Department)
SubClassOf(:Human_Resource ObjectSomeValuesFrom(:works_in :Department))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has some skill

Human Resource has skill	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_skill))
  DataPropertyDomain(:has_skill :Human_Resource)
  DataPropertyRange(:has_skill xsd:string)) SubClassOf(:Human_Resource DataSomeValuesFrom(:has_skill xsd:string))

A.19. SBVR Cuantificación Como Máximo N (At Most N Quantification)

Nombre

Cuantificación Como Máximo N (At Most N Quantification) a OWL 2.

Objetivo

Expresar una cuantificación como máximo n SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación como máximo n SBVR es una cuantificación que tiene una cardinalidad máxima de valor n - donde n es un entero positivo - y tiene el siguiente significado: el número de referentes distintos de la variable introducida

por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es mayor que la cardinalidad máxima n.

Expresión OWL 2

Una cuantificación como máximo n SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación máxima (de valor igual a n) de objetos OWL 2. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia de cuantificación máxima (de valor igual a n) de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación máxima (de valor igual a n) de objetos OWL 2.

  ObjectMaxCardinality(n OPE CE)

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación máxima (de valor igual a n) de literales OWL 2.

  DataMaxCardinality(n DPE DR)

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Boss	
Necesidad:	has in charge at most 25 Employee

Boss has in charge Employee	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Boss))
  Declaration(Class(:Employee))
  Declaration(ObjectProperty(:has_in_charge))
  ObjectPropertyDomain(:has_in_charge :Boss)
  ObjectPropertyRange(:has_in_charge :Employee)
  SubClassOf(:Boss ObjectMaxCardinality(25 :has_in_charge :Employee))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has at most 1 HR-ID

Human Resource has HR-ID	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_HR-ID))
  DataPropertyDomain(:has_HR-ID :Human_Resource)
  DataPropertyRange(:has_HR-ID xsd:integer)
  SubClassOf(:Human_Resource DataMaxCardinality(1 :has_HR-ID xsd:integer)))

A.20. SBVR Cuantificación Al Menos N (At Least N Quantification)

Nombre

Cuantificación Al Menos N (At Least N Quantification) a OWL 2.

Objetivo

Expresar una cuantificación al menos n SBVR mediante axiomas OWL 2

Constructor SBVR

Una cuantificación al menos n SBVR es una cuantificación que tiene una cardinalidad mínima n - donde n es un entero positivo - y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es menor que la cardinalidad mínima n.

Expresión OWL 2

Una cuantificación al menos n SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación mínima (de valor igual a n) de objetos OWL 2. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia de cuantificación mínima (de valor igual a n) de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación mínima (de valor igual a n) de objetos OWL 2.

  ObjectMinCardinality(n OPE CE)

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación mínima (de valor igual a n) de literales OWL 2.

  DataMinCardinality(n DPE DR)

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Boss	
Necesidad:	has in charge at least 1 Employee

Boss has in charge Employee	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Boss))
  Declaration(Class(:Employee))
  Declaration(ObjectProperty(:has_in_charge))
  ObjectPropertyDomain(:has_in_charge :Boss)
  ObjectPropertyRange(:has_in_charge :Employee)
  SubClassOf(:Boss ObjectMinCardinality(1 :has_in_charge :Employee))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has at least 1 skill

Human Resource has skill	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_skill))
  DataPropertyDomain(:has_skill :Human_Resource)
  DataPropertyRange(:has_skill xsd:string)
  SubClassOf(:Human_Resource DataMinCardinality(1 :has_skill xsd:string)))

A.21. SBVR Cuantificación Exactamente N (Exactly N Quantification)

Nombre

Cuantificación Exactamente N (Exactly N Quantification) a OWL 2.

Objetivo

Expresar una cuantificación exactamente n SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación exactamente n SBVR es una cuantificación que tiene una cardinalidad n - donde n es un entero positivo - y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), es igual al de la cardinalidad n.

Expresión OWL 2

Una cuantificación exactamente n SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación exacta (de valor igual a n) de objetos OWL 2. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia de cuantificación exacta (de valor igual a n) de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación exacta (de valor igual a n) de objetos OWL 2.

  ObjectExactCardinality(n OPE CE)

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación exacta (de valor igual a n) de literales OWL 2.

  DataExactCardinality(n DPE DR)

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Employee	
Necesidad:	is in charge of exactly 1 Boss

Employee is in charge of Boss	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Boss))
  Declaration(Class(:Employee))
  Declaration(ObjectProperty(:is_in_charge_of))
  ObjectPropertyDomain(:is_in_charge_of :Employee)
  ObjectPropertyRange(:is_in_charge_of :Boss)
  SubClassOf(:Employee ObjectExactCardinality(1 :is_in_charge_of :Boss))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has exactly 1 HR-ID

Human Resource has HR-ID	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_HR-ID))

DataPropertyDomain(:has_HR-ID :Human_Resource)
DataPropertyRange(:has_HR-ID xsd:integer)
SubClassOf(:Human_Resource DataExactCardinality(1 :has_HR-ID xsd:integer)))

A.22. SBVR Cuantificación Como Máximo Uno (At Most One Quantification)

Nombre

Cuantificación Como Máximo Uno (At Most One Quantification) a OWL 2.

Objetivo

Expresar una cuantificación como máximo uno SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación como máximo uno SBVR es una cuantificación que tiene una cardinalidad máxima de valor $n = 1$ y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es mayor a 1.

Expresión OWL 2

Una cuantificación como máximo uno SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación máxima (de valor igual a 1) de objetos OWL 2. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia de cuantificación máxima (de valor igual a 1) de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación máxima (de valor igual a 1) de objetos OWL 2.

  *ObjectMaxCardinality(*1 *OPE CE)*

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación máxima (de valor igual a 1) de literales OWL 2.

  *DatatMaxCardinality(*1 *DPE DR)*

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	works in at most 1 Department

Department	
Tipo de concepto:	Concepto General

Human Resource works in Department	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(Class(:Department))
  Declaration(ObjectProperty(:works_in))
  ObjectPropertyDomain(:works_in :Human_Resource)
  ObjectPropertyRange(:works_in :Department)
  SubClassOf(:Human_Resource ObjectMaxCardinality(1 :works_in :Department))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has at most 1 HR-ID

Human Resource has HR-ID	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_HR-ID))
  DataPropertyDomain(:has_HR-ID :Human_Resource)
  DataPropertyRange(:has_HR-ID xsd:integer)
  SubClassOf(:Human_Resource DataMaxCardinality(1 :has_HR-ID xsd:integer)))

A.23. SBVR Cuantificación Exactamente Uno (Exactly One Quantification)

Nombre

Cuantificación Exactamente Uno (Exactly One Quantification) a OWL 2.

Objetivo

Expresar una cuantificación exactamente uno SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación exactamente uno SBVR es una cuantificación que tiene una cardinalidad de valor $n = 1$ y tiene el siguiente significado: el número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), es igual a 1.

Expresión OWL 2

Una cuantificación exactamente uno SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de cuantificación exacta (de valor igual a 1) de objetos OWL 2.

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia de cuantificación exacta (de valor igual a 1) de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación exacta (de valor igual a 1) de objetos OWL 2.

  *ObjectExactCardinality(*1 *OPE CE)*

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación exacta (de valor igual a 1) de literales OWL 2.

  *ObjectExactCardinality(*1 *OPE CE)*

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Employee	
Necesidad:	in charge of exactly 1 Boss

Employee is in charge of Boss	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Boss))
  Declaration(Class(:Employee))
  Declaration(ObjectProperty(:is_in_charge_of))
  ObjectPropertyDomain(:is_in_charge_of :Employee)
  ObjectPropertyRange(:is_in_charge_of :Boss)
  SubClassOf(:Employee ObjectExactCardinality(1 :is_in_charge_of :Boss))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has exactly 1 HR-ID

Human Resource has HR-ID	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_HR-ID))
  DataPropertyDomain(:has_HR-ID :Human_Resource)
  DataPropertyRange(:has_HR-ID xsd:integer)
  SubClassOf(:Human_Resource DataExactCardinality(1 :has_HR-ID xsd:integer)))

A.24. SBVR Cuantificación de Rango Numérico (Numeric Range Quantification)

Nombre

Cuantificación de Rango Numérico (Numeric Range Quantification) a OWL 2.

Objetivo

Expresar una cuantificación de rango numérico SBVR mediante axiomas OWL 2.

Constructor SBVR

Una cuantificación de rango numérico SBVR es una cuantificación que tiene una cardinalidad mínima y una cardinalidad máxima (donde la cardinalidad máxima es mayor a la cardinalidad mínima) y tiene el siguiente significado: el

número de referentes distintos de la variable introducida por la cuantificación, que existen y satisfacen el ámbito de la formulación (si hay alguno), no es menor a la cardinalidad mínima ni mayor a la cardinalidad máxima.

Expresión OWL 2

Una cuantificación de rango numérico SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de formulación lógica involucrada. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos la expresión se trasforma en una sentencia de intersección de expresiones de clase OWL 2. Una de las expresiones de clase es una sentencia de cuantificación mínima de objetos OWL 2 y la restante es una sentencia de cuantificación máxima de objetos OWL 2. Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se trasforma en una sentencia de intersección de expresiones de clase OWL 2. Una de las expresiones de clase es una sentencia de cuantificación mínima de literales OWL 2 y la restante es una sentencia de cuantificación máxima de literales OWL 2.

Proceso

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de cuantificación mínima (de valor igual a n_1) de objetos OWL 2.

  ObjectMinCardinality(n_1 OPE CE)

- Paso 2. Crear sentencia de cuantificación máxima (de valor igual a n_2) de objetos OWL 2.

  ObjectMaxCardinality(n_2 OPE CE)

- Paso 3. Crear sentencia de intersección de expresiones de clase vinculando las sentencias creadas en los pasos anteriores.

  ObjectIntersectionOf(ObjectMinCardinality(n_1 OPE CE) ObjectMaxCardinality(n_2 OPE CE))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de cuantificación mínima (de valor igual a n_1) de literales OWL 2.

DataMinCardinality(n_1 DPE DR)

- Paso 2. Crear sentencia de cuantificación máxima (de valor igual a n_2) de literales OWL 2.

  DataMaxCardinality(n_2 DPE DR)

- Paso 3. Crear sentencia de intersección de expresiones de clase vinculando las sentencias creadas en los pasos anteriores.

  DataIntersectionOf(DataMinCardinality(n_1 DPE DR) DataMaxCardinality(n_2 DPE DR))

Ejemplo

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Boss	
Necesidad:	has in charge at least 1 Employee and has in charge at most 25 Employee

Boss has in charge Employee	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Boss))
  Declaration(Class(:Employee))
  Declaration(ObjectProperty(:has_in_charge))
  ObjectPropertyDomain(:has_in_charge :Boss)
  ObjectPropertyRange(:has_in_charge :Employee)
  SubClassOf(:Boss ObjectIntersectionOf(ObjectMaxCardinality(25 :has_in_charge :Employee) ObjectMinCardinality(1 :has_in_charge :Employee)))

Si la cuantificación se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General
Necesidad:	has at least 1 HR-ID and has at most 1 HR-ID

Human Resource has HR-ID	
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(DataProperty(:has_HR-ID))
  DataPropertyDomain(:has_HR-ID :Human_Resource)
  DataPropertyRange(:has_HR-ID xsd:integer)
  SubClassOf(:Human_Resource ObjectIntersectionOf(DataMaxCardinality(1 :has_HR-ID xsd:integer) DataMinCardinality(1 :has_HR-ID xsd:integer)))

A.25. SBVR Especialización/Generalización (Especialization/Generalization)

Nombre

SBVR Especialización/Generalización (Especialization/Generalization) a OWL 2.

Objetivo

Expresar una especialización/generalización SBVR mediante axiomas OWL 2.

Constructor SBVR

Una especialización SBVR es utilizada para representar una relación de jerarquía entre conceptos. Sean conceptos C_1 y C_2, donde C_1 especializa a C_2. Luego C_1 incorpora cada característica que forma parte de C_2, más al menos una característica adicional diferenciadora.

Expresión OWL 2

Una especialización SBVR se transforma en una expresión OWL 2 de acuerdo al tipo de conceptos involucrados.

Si la especialización se aplica sobre conceptos generales la expresión se transforma en una sentencia de especialización de clases OWL 2. Si la especialización se aplica sobre un concepto verbal que vincula conceptos la expresión se transforma en una sentencia de especialización de propiedades de objeto OWL 2. Si la especialización se aplica sobre un concepto verbal que vincula conceptos con literales la expresión se transforma en una sentencia de especialización de propiedades de datos OWL 2. Si la especialización se aplica entre un concepto individual y un concepto general la expresión es interpretada como una sentencia de clasificación (Sección A.7).

Proceso

Si la especialización se aplica sobre conceptos generales:

- Paso 1. Crear sentencia de especialización de clases OWL 2.

  $SubClassOf(CE_1\ CE_2)$

Si la especialización se aplica sobre un concepto verbal que vincula conceptos:

- Paso 1. Crear sentencia de especialización de propiedades de objeto OWL 2.

  $SubObjectPropertyOf(OPE_1\ OPE_2)$

Si la especialización se aplica sobre un concepto verbal que vincula conceptos con literales:

- Paso 1. Crear sentencia de especialización de propiedades de datos OWL 2.

  $SubDataPropertyOf(DPE_1\ DPE_2)$

Si la especialización se aplica entre un concepto individual y un concepto general:

- Paso 1. Interpretar la expresión como una sentencia de clasificación. Aplicar transformación propuesta en Sección A.7.

  Declaration(NamedIndividual(I))

  ClassAssertion(CE I)

Ejemplo

Si la especialización se aplica sobre conceptos generales:

- Entrada del glosario SBVR

Intern	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General

- Expresión OWL 2

  Declaration(Class(:Intern)) Declaration(Class(:Human_Resource))

  SubClassOf(:Intern :Human_Resource)

Si la especialización se aplica sobre un concepto verbal que vincula conceptos:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General

Intern	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto General

Human Resource works in Department	
Tipo de concepto:	Concepto Verbal Binario

Intern has scholarship at Department	
Concepto padre:	works in
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2

  Declaration(Class(:Intern))

  Declaration(Class(:Human_Resource))

  SubClassOf(:Intern :Human_Resource)

  Declaration(ObjectProperty(:works_in))

ObjectPropertyDomain(:works_in :Human_Resource)
ObjectPropertyRange(:works_in :Department)
Declaration(ObjectProperty(:has_scholarship_at))
SubObjectPropertyOf(:has_scholarship_at :works_in)

Si la especialización se aplica sobre un concepto verbal que vincula conceptos con literales:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General

Human Resource has name	
Tipo de concepto:	Concepto Verbal Binario

Human Resource has fullname	
Concepto padre:	has name
Tipo de concepto:	Concepto Verbal Binario

- Expresión OWL 2
Declaration(Class(:Human_Resource))
Declaration(DataProperty(:has_name))
DataPropertyDomain(:has_name :Human_Resource)
DataPropertyRange(:has_name xsd:string)
Declaration(DataProperty(:has_fullname))
SubDataPropertyOf(:has_fullname :has_name)

Si la especialización se aplica entre un concepto individual y un concepto general:

- Entrada del glosario SBVR

John Smith	
Concepto padre:	Human Resource
Tipo de concepto:	Concepto Individual

- Expresión OWL 2
  Declaration(NamedIndividual(:John_Smith))
  ClassAssertion(:Human_Resource :John_Smith)

A.26. SBVR Esquema de Categorización (Categorization Scheme)

Nombre

SBVR Esquema de Categorización (Categorization Scheme) a OWL 2.

Objetivo

Expresar un esquema de categorización SBVR mediante axiomas OWL 2.

Constructor SBVR

Un esquema de categorización SBVR es un medio de particionar conceptos individuales pertenecientes a un concepto general determinado, de acuerdo a un conjunto de categorías predefinidas para dicho esquema. La definición de diferentes esquemas de categorización - y sus correspondientes categorías - para un mismo concepto general permite clasificar sus conceptos individuales en función de diversos criterios: cada esquema representa un criterio dado el cual resulta materializado a través de sus respectivas categorías. Un esquema de categorización puede ser definido en forma parcial, donde pueden existir individuos del concepto clasificado que no pertenezcan a ninguna de las categorías definidas, o completa, donde cada individuo del concepto clasificado pertenece necesariamente a alguna de las categorías consideradas.

Expresión OWL 2

Un esquema de categorización SBVR es transformado en sentencias OWL 2 de la manera que se indica en el siguiente apartado.

Proceso

Si el esquema de categorización es parcial:

- Paso 1. Crear una clase OWL 2 que representa el criterio de clasificación considerado.

  $Definition(Class(C_{clas}))$

- Paso 2. Crear una clase OWL por cada una de las categorías del esquema de categorización.

  Definition(Class(C_{cat1}))

  Definition(Class(C_{cat2}))

  Definition(Class(C_{catn}))

- Paso 3. Crear una sentencia de especialización OWL 2 entre cada una de las clases que representan las categorías de especialización y la clase que representa el criterio de clasificación considerado.

  SubClassOf(C_{cat1} C_{clas})

  SubClassOf(C_{cat2} C_{clas})

  SubClassOf(C_{catn} C_{clas})

- Paso 4. Crear una propiedad de objeto OWL 2 entre la clase sobre la cual se aplica el criterio de clasificación y la clase representando dicho criterio.

  Declaration(ObjectProperty(OPE))

  ObjectPropertyDomain(OPE C)

  ObjectPropertyRange(OPE C_{clas})

Si el esquema de categorización es completo:

- Paso 1. Crear una clase OWL 2 que representa el criterio de clasificación considerado.

  Definition(Class(C_{clas}))

- Paso 2. Crear una clase OWL por cada una de las categorías del esquema de categorización.

  Definition(Class(C_{cat1}))

  Definition(Class(C_{cat2}))

Definition(Class(C_{catn}))

- Paso 3. Crear una sentencia de equivalencia OWL 2 entre la clase que representa el criterio de clasificación considerado y la unión de las clases que representan las categorías de especialización.

  EquivalentClasses(C_{clas} ObjectUnionOf(C_{cat1} C_{cat2} C_{catn}))

- Paso 4. Crear una propiedad de objeto OWL 2 entre la clase sobre la cual se aplica el criterio de clasificación y la clase representando dicho criterio.

  Declaration(ObjectProperty(OPE))

  ObjectPropertyDomain(OPE C)

  ObjectPropertyRange(OPE C_{clas})

Ejemplo

Si el esquema de categorización es parcial:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General

Expertise Classification	
Definición:	Categorización parcial para Human Resource
Tipo de concepto:	Esquema de Categorización
Necesidad:	Posee la categoría Senior Human Resource
	Posee la categoría Junior Human Resource

Senior Human Resource	
Tipo de concepto:	Categoría

Junior Human Resource	
Tipo de concepto:	Categoría

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(Class(:Expertise_Classification))
  Declaration(Class(:Senior_Human_Resource))
  SubClassOf(:Senior_Human_Resource :Expertise_Classification)
  Declaration(Class(:Junior_Human_Resource))
  SubClassOf(:Junior_Human_Resource :Expertise_Classification)
  Declaration(ObjectProperty(:has_Expertise_Classification))
  ObjectPropertyDomain(:has_Expertise_Classification :Human_Resource)
  ObjectPropertyRange(:has_Expertise_Classification
  :Expertise_Classification)

Si el esquema de categorización es completo:

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General

Expertise Classification	
Definición:	Categorización completa para Human Resource
Tipo de concepto:	Esquema de Categorización
Necesidad:	Posee la categoría Senior Human Resource
	Posee la categoría Junior Human Resource

Senior Human Resource	
Tipo de concepto:	Categoría

Junior Human Resource	
Tipo de concepto:	Categoría

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(Class(:Expertise_Classification))

Declaration(Class(:Senior_Human_Resource))
Declaration(Class(:Junior_Human_Resource))
EquivalentClasses(:Expertise_Classification ObjectUnionOf(:Senior_Human_Resource
:Junior_Human_Resource))
Declaration(ObjectProperty(:has_Expertise_Classification))
ObjectPropertyDomain(:has_Expertise_Classification :Human_Resource)
ObjectPropertyRange(:has_Expertise_Classification
:Expertise_Classification)

A.27. SBVR Segmentación (Segmentation)

Nombre

SBVR Segmentación (Segmentation) a OWL 2.

Objetivo

Expresar una segmentación SBVR mediante axiomas OWL 2.

Constructor SBVR

Una segmentación SBVR es un esquema de categorización completo, donde cada individuo del concepto clasificado pertenece necesariamente a alguna de las categorías consideradas, y disjunto, donde los individuos del concepto clasificado pertenecen a una y sólo una de las categorías definidas.

Expresión OWL 2

Una segmentación SBVR es transformada en sentencias OWL 2 de la manera que se indica en el siguiente apartado.

Proceso

- Paso 1. Crear una clase OWL 2 que representa el criterio de clasificación considerado.

  Definition(Class(C_{clas}))

- Paso 2. Crear una clase OWL por cada una de las categorías de la segmentación.

  Definition(Class(C_{cat1}))

  Definition(Class(C_{cat2}))

Definition(Class(C_{catn}))

- Paso 3. Definir la clase OWL 2 que representa el criterio de clasificación considerado como la unión disjunta de las clases que representan las categorías de especialización.

  DisjointUnion(C_{clas} C_{cat1} C_{cat2} C_{catn})

- Paso 4. Crear una propiedad de objeto OWL 2 entre la clase sobre la cual se aplica el criterio de clasificación y la clase representando dicho criterio.

  Declaration(ObjectProperty(OPE))

  ObjectPropertyDomain(OPE C)

  ObjectPropertyRange(OPE C_{clas})

Ejemplo

- Entrada del glosario SBVR

Human Resource	
Tipo de concepto:	Concepto General

Expertise Classification	
Definición:	Segmentación para Human Resource
Tipo de concepto:	Esquema de Categorización
Necesidad:	Posee la categoría Senior Human Resource
	Posee la categoría Junior Human Resource

Senior Human Resource	
Tipo de concepto:	Categoría

Junior Human Resource	
Tipo de concepto:	Categoría

- Expresión OWL 2

  Declaration(Class(:Human_Resource))
  Declaration(Class(:Expertise_Classification))
  Declaration(Class(:Senior_Human_Resource))
  Declaration(Class(:Junior_Human_Resource))
  DisjointUnion(:Expertise_Classification :Senior_Human_Resource :Junior_Human_Resource)
  Declaration(ObjectProperty(:has_Expertise_Classification))
  ObjectPropertyDomain(:has_Expertise_Classification :Human_Resource)
  ObjectPropertyRange(:has_Expertise_Classification :Expertise_Classification)

Apéndice B

Acerca de la factibilidad técnica de SBVR como lenguaje de modelado para el desarrollo de ontologías: un experimento exploratorio

Este apéndice describe un *experimento exploratorio* destinado a evaluar la factibilidad técnica del mapeo de expresiones de negocio en sentencias ontológicas. La *investigación exploratoria* se realiza en aquellos casos en los cuales el problema no ha sido claramente definido, o cuando se desea ganar familiaridad con el objeto de estudio a fin de adquirir datos adicionales. Los estudios exploratorios se ejecutan mediante métodos tan diversos como lo son los *estudios de prueba*, las *entrevistas*, las *discusiones grupales*, los *experimentos*, o cualquier otra técnica útil en la obtención de nueva información. Aunque la naturaleza básica de este tipo de estudios permite obtener conclusiones sólo con extremo cuidado, su ejecución brinda a los investigadores la información necesaria para mejorar el planeamiento de experimentos futuros. Luego, los estudios exploratorios constituyen un importante mecanismo para la generación de hipótesis y proveen una guía para futuras actividades de investigación (Kitchenham y otros, 2002).

El objetivo del experimento presentado consiste en la evaluación de la factibilidad técnica del mapeo de expresiones de negocio en sentencias ontológicas. El estudio se basa en la evaluación de la calidad de un conjunto de ontologías, las cuales fueron desarrolladas aplicando técnicas basadas en la utilización de UML y SBVR. Las ontologías reflejan el punto de vista de profesionales de la ingeniería de software, dado que el experimento fue llevado a cabo por estudiantes del

último nivel de la carrera de ingeniería en sistemas.

B.1. Objetivo e hipótesis

El experimento estudia la factibilidad técnica del mapeo de expresiones de negocio en sentencias ontológicas evaluando la siguiente hipótesis:
Hipótesis: *Las ontologías desarrolladas mediante un técnica basada en SBVR al menos igualan la calidad de las ontologías desarrolladas mediante un método basado en UML.*
La verificación de la hipótesis se encuentra basada en la evaluación de la calidad de las ontologías desarrolladas por diferentes grupos de estudiantes de ingeniería aplicando las técnicas basadas en UML y SBVR.

B.2. Enfoques bajo estudio

El experimento involucra dos alternativas de estudio:

- El desarrollo de ontologías aplicando el método UPON decripto en de Nicola y otros (2009)

- El desarrollo de ontologías aplicando el mapeo de expresiones SBVR propuesto en Reynares y otros (2013)

UPON es una metodología basada en el Proceso Unificado (UP) (Jacobson y otros, 1999) para el desarrollo incremental de ontologías, la cual utiliza UML en la construcción de los artefactos intermedios generados durante la ejecución de las actividades. El proceso genera un lexicón mediante la identificación de los términos más relevantes del dominio, el cual es enriquecido progresivamente con definiciones hasta obtener un glosario. La determinación de las relaciones ontológicas entre los conceptos del glosario genera una red semántica, la cual es enriquecida y finalmente formalizada como la ontología de dominio.

B.3. Contexto y unidades experimentales

El experimento fue realizado en el contexto del curso de grado denominado *"Desarrollo de Sistemas de Información basados en Ontologías"*, el cual forma parte del último nivel de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional en la ciudad de Santa Fe, Argentina. Las unidades experimentales consistieron en 10 grupos de igual tamaño, conformados por 30 estudiantes del curso. Un proceso aleatorio fue utilizado para la generación de muestras de igual tamaño y la asignación de las unidades experimentales a los distintos enfoques. De esta forma, 5 grupos aplicaron el enfoque de mapeo basado en SBVR, mientras los restantes 5 siguieron el enfoque UML. Los participantes obtuvieron créditos académicos por formar parte del experimento. Además, respondieron una encuesta anónima acerca de su conocimiento previo en relación a los temas involucrados en el experimento, evaluando las mismas en un rango de valores de 1 - *conocimientos previos nulos* - a 10 - *conocimientos previos de nivel profesional* -.

Aunque la naturaleza anónima de la encuesta buscaba obtener respuestas sinceras, es necesario considerar el sesgo subjetivo presente en cualquier auto evaluación. En las Figuras B.1-B.3 se muestran las comparaciones de frecuencia sobre conocimientos previos en (P1) el uso de artefactos UML, (P2) el desarrollo de sentencias lógicas y (P3) la ingeniería ontológica, respectivamente. Para evaluar las diferencias en las respuestas de los enfoques comparados fue utilizado el test de Mann-Whitney-Wilcoxon (MWW). Este test (también denominado estadístico U, o simplemente U) es un test no paramétrico de la hipótesis nula - donde dos poblaciones resultan iguales - contra una hipótesis alternativa - donde una población particular tiende a tener valores mayores que la otra -. El test posee una eficiencia mayor al test *"t"* en distribuciones no-normales y una eficiencia cercana a la del test *"t"* en distribuciones normales (Mann y Whitney, 1947)(Fay y Proschan, 2010).

La Tabla B.1 muestra los intervalos críticos de U para dos muestras de igual tamaño A y B (donde el tamaño $n = 15$), para test direccionales y no direccionales y para los niveles de significancia más comúnmente utilizados. La hipótesis nula- ambos poblaciones tienden a poseer los mismo valores - se acepta si los valores observados de U se encuentran en el rango comprendido entre los límites

superiores e inferiores. La Tabla B.2 presenta el valor observado de U para ambos enfoques en cada una de las preguntas de la encuesta. Los valores observados permiten concluir que no existe diferencia estadísticamente significativa entre ambos enfoques en relación a su conocimiento previo sobre P1, P2, y P3. Tales resultados permitieron concluir que ambos grupos poseían el mismo nivel de conocimientos previos en relación a los temas involucrados en el experimento, los cuales se asemejan al conocimiento que un ingeniero de software sin formación previa en ingeniería ontológica posee inicialmente.

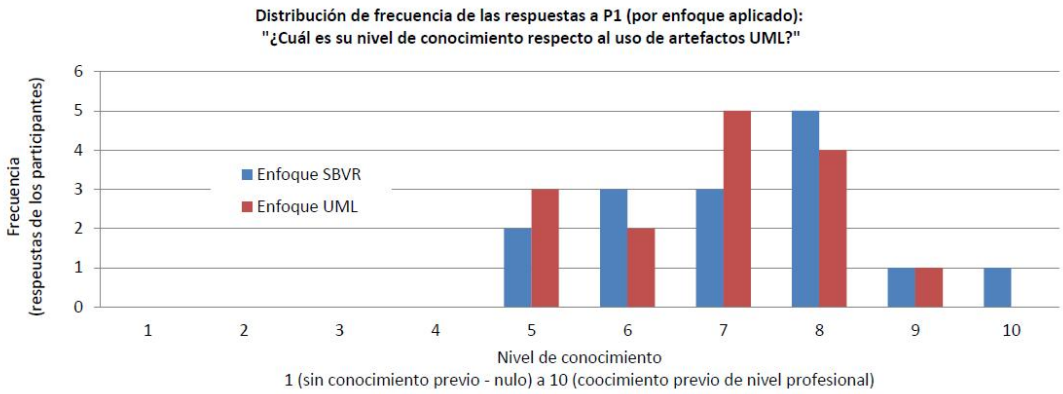

Figura B.1: Distribución de frecuencia de las respuestas a P1 (por enfoque aplicado): "¿Cuál es su nivel de conocimiento respecto al uso de artefactos UML?"

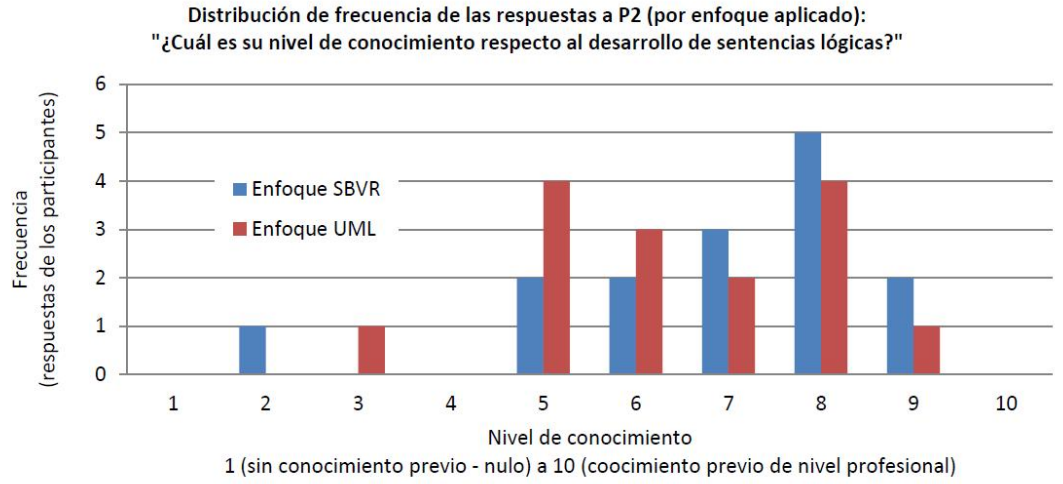

Figura B.2: Distribución de frecuencia de las respuestas a P2 (por enfoque aplicado): "¿Cuál es su nivel de conocimiento respecto al desarrollo de sentencias lógicas?"

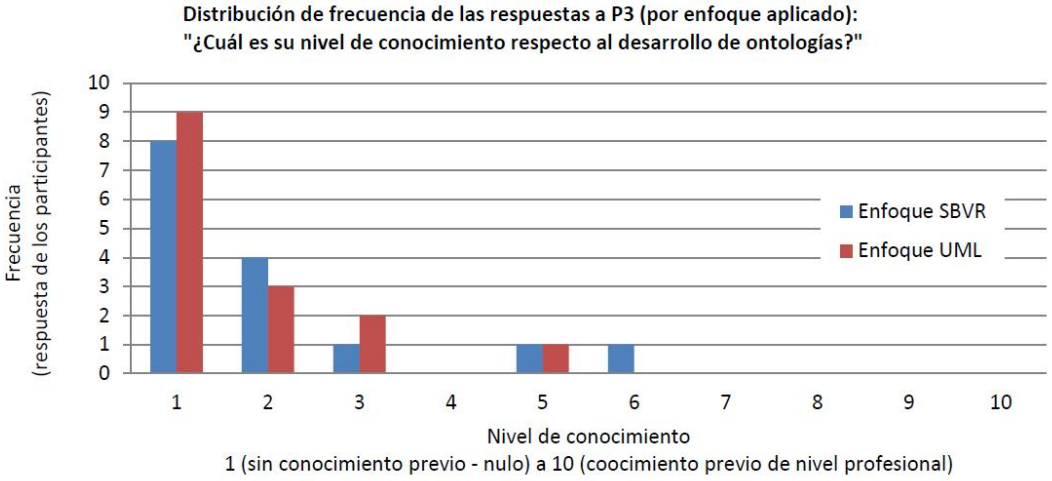

Figura B.3: Distribución de frecuencia de las respuestas a P3 (por enfoque aplicado): "¿Cuál es su nivel de conocimiento respecto al desarrollo de ontologías?"

Intervalos críticos de U			
	Nivel de significancia para:		
	Tests Direccionales		
	0.05	0.025	0.01
	Tests no-Direccionales		
	-	0.05	0.02
Límite inferior	72	64	56
Límite superior	153	161	169

Tabla B.1: Intervalos críticos de U para dos muestras de tamaño $n = 15$)

	Valores de U	
	Enfoque SBVR	Enfoque UML
Conocimientos previos en:		
(P1) el uso de artefactos UML	98	127
(P2) el desarrollo de sentencias lógicas	90	135
(P3) ingeniería ontológica	104.5	120.5

Tabla B.2: Valores de U por enfoque aplicado

B.4. Tarea y materiales

La tarea del experimento consistió en la especificación ontológica de las políticas de asignación de becas estudiantiles de la universidad, especificadas originalmente en un documento oficial de la institución y descriptas en lenguaje natural. Con respecto a las herramientas, ambos enfoques utilizaron editores de

texto para modelar el dominio de negocio e implementaron la ontología en OWL 2 por medio de Protégé, un editor libre y de código abierto.

B.5. Resultados

La performance de los dos enfoques fue comparada mediante la evaluación de la calidad de las ontologías desarrolladas. Tal evaluación fue realizada por los autores de este trabajo mediante OQuaRE (Duque-Ramos y otros, 2011)s(Duque-Ramos y otros, 2013), un marco de trabajo basado en el estándar SQuaRE para la evaluación de la calidad del software (ISO, 2005). OQuaRE evalúa las ontologías independientemente de cualquier proceso de desarrollo particular, lo cual consituye un proceso objetivo de evaluación capaz de ser replicado.

OQuaRE define un modelo de calidad y sus respectivas métricas para la evaluación de las ontologías. El modelo de calidad es dividido en dimensiones - o características - organizadas en sub-dimensiones - o sub-características -, las cuales son evaluadas por medio de un conjunto de métricas. OQuaRE define el criterio para normalizar los valores de las métricas en un rango de 1 a 5: calidad no aceptable es asociada al valor 1, la mínima calidad aceptable responde al valor 3, y el valor 5 excede los requerimientos de calidad. Luego de la normalización, el puntaje asociado a cada una de las características es la media de las sub-características asociadas, cuyo valor es calculado como la media de sus métricas. El resultado final de la evaluación de calidad consiste en el conjunto de valores para cada una de las características evaluadas. Esto permite la identificación de las fortalezas y debilidades de las ontologías consideradas en lugar de señalar la *"mejor"* ontología, como se propone en los trabajos de Lozano-Tello y Gómez-Pérez (2004)Park y otros (2011)(Vrandečić, 2010).

Las dimensiones de calidad evaluadas en el presente experimento se describen a continuación:

- La *dimensión estructural* involucra propiedades formales y semánticas que resultan importantes al evaluar ontologías, tales como consistencia, nivel de formalización, redundancia o complejidad.

- La *dimensión de adecuación funcional* refiere al grado de alineamiento de la ontología para su finalidad prevista, de acuerdo a las categorías identificadas por (Stevens y Lord, 2009).

- La *dimensión de mantenibilidad* está relacionada a la capacidad de las ontologías de ser modificadas de acuerdo a cambios en el entorno, en los requerimientos o en las especificaciones funcionales.

- La *dimensión de compatibilidad* refiere a la habilidad de dos o más ontologías para intercambiar información y/o cumplir su función mientras comparten el mismo entorno de hardware o software. Esta dimensión puede ser evaluada sobre una única ontología - aunque intuitivamente involucra las propiedades de más de una -, dado que es cuantitativamente evaluada por medio de un conjunto de métricas aplicadas a cada ontología en forma separada.

- La *dimensión de transferibilidad* es el grado con el cual la ontología puede ser transferida de un entorno a otro.

- La *dimensión de operatividad* refiere al esfuerzo necesario para utilizar la ontología por un determinado conjunto de usuarios.

- La *dimensión de confiabilidad* es la capacidad de la ontología de mantener su nivel de performance bajo condiciones establecidas, por un período dado de tiempo.

Tres métricas no fueron consideradas en la evaluación de las dimensiones de calidad. Las métricas denominadas riqueza de las anotaciones - número medio de anotaciones por clase - y riqueza de clases - número medio de instancias por clase - no fueron evaluadas dado que la anotación e instanciación de ontologías no formaba parte del experimento. La métrica denominada riqueza de atributos - número medio de atributos por clase - no fue evaluada dado que los atributos no forman parte de la especificación estructural de lenguaje de implementación de las ontologías (W3C, 2009b).
OQuaRE define también las dimensiones denominadas eficiencia de performance y calidad de uso. La eficiencia de performance expone la relación entre el nivel de performance de la ontología y los recursos utilizados bajo condiciones establecidas,

tomando en cuenta elementos tales como tiempo de respuesta o consumo de memoria. La calidad en uso refiere al grado con el cual la ontología satisface los requerimientos de usuarios específicos. Sin embargo, tales dimensiones no fueron consideradas dado la inexistencia de métricas asociadas a sus correspondientes sub-características. La Figura B.4 muestra los niveles de calidad de las ontologías desarrolladas por medio del enfoque basado en el mapeo de expresiones SBVR. La Figura B.5 muestra los niveles de calidad de las ontologías desarrolladas mediante el enfoque UML.

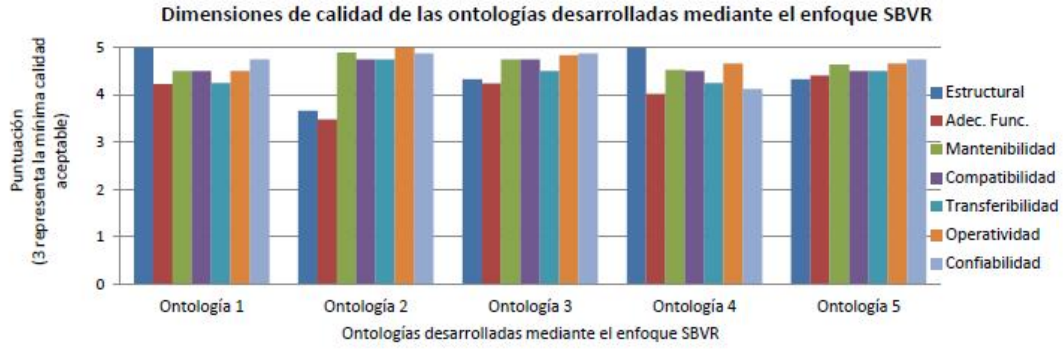

Figura B.4: Puntuación de las dimensiones de calidad de las ontologías desarrolladas mediante el enfoque SBVR

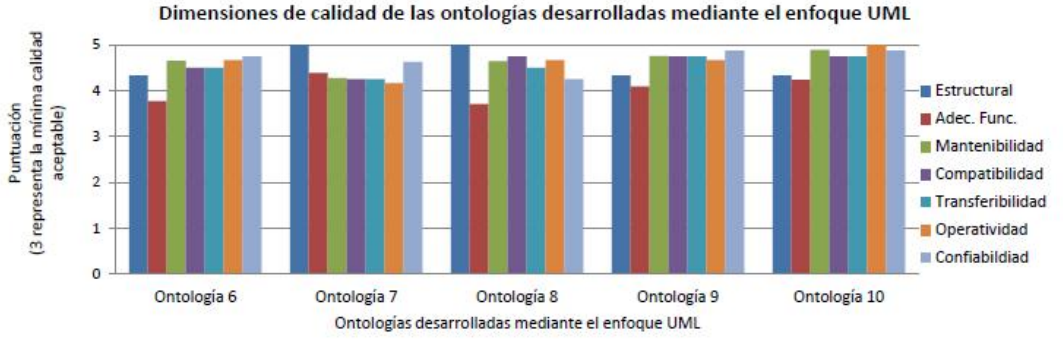

Figura B.5: Puntuación de las dimensiones de calidad de las ontologías desarrolladas mediante el enfoque UML

B.6. Análisis de resultados

Un rápido vistazo a las Figuras B.4 y B.5 permite reconocer un primer resultado importante: de acuerdo a OQuaRE, todas las ontologías superan la

mínima calidad aceptable.

La Tabla B.3 muestra el valor medio de cada una de las dimensiones de calidad evaluadas: la primer columna muestra el nivel de calidad de las ontologías desarrolladas mediante el enfoque basado en SBVR mientras la segunda presenta los valores de las ontologías obtenidas por medio de ODM. Además, la última fila de la tabla muestra el valor medio de la calidad de la ontología de acuerdo al enfoque seguido. Una comparación fila por fila permite observar que las medias de las dimensiones de calidad son muy similares entre ambos enfoques, incluso en sus valores medios globales. Nuevamente, el test MWW permitió evaluar las diferencias entre los valores obtenidos. La Tabla B.4 muestra los intervalos críticos de U para dos muestras de tamaño $n = 5$, tanto para test direccionales como no-direccionales y para los niveles de significancia más comúnmente considerados. Finalmente, la Tabla B.5 muestra los valores de U para cada enfoque en cada una de las dimensiones de calidad evaluadas. Los valores observados permiten concluir que no existen diferencias estadísticamente significativas entre ambos enfoques, en ninguna de las dimensiones evaluadas.

	Enfoque SBVR	Enfoque ODM
Estructural	4.47	4.60
Adecuación Funcional	4.05	4.05
Mantenibilidad	4.67	4.64
Compatibilidad	4.60	4.60
Transferibilidad	4.45	4.55
Operatividad	4.73	4.63
Confiabilidad	4.68	4.68
Media Global	4.52	4.54

Tabla B.3: Valores medios por dimensión de calidad y enfoque aplicado

Intervalos críticos de U			
	Nivel de significancia para:		
	Tests Direccionales		
	0.05	0.025	0.01
	Tests no-Direccionales		
	-	0.05	0.02
Límite inferior	4	2	1
Límite superior	21	23	24

Tabla B.4: Intervalos críticos de U para dos muestras de tamaño $n = 5$

	Valores de U	
	Enfoque SBVR	Enfoque ODM
Estructural	14	11
Adecuación Funcional	11.5	13.5
Mantenibilidad	12.5	12.5
Compatibilidad	13.5	11.5
Transferibilidad	16	9
Operatividad	10.5	14.5
Confiabilidad	12	9

Tabla B.5: Valores de U por dimensión de calidad y enfoque aplicado

B.7. Discusión

El experimento presentado se encuentra sesgado en forma negativa con respecto al enfoque basado en el mapeo de expresiones SBVR, dado que los participantes poseían conocimientos previos en el uso de artefactos UML mientras el conocimiento acerca del lenguaje SBVR era inexistente. Además, el enfoque basado en UML provee guías metodológicas para cada etapa del proceso de desarrollo de la ontología, mientras que el enfoque basado en SBVR se encuentra focalizado únicamente en la generación automatizable de una ontología OWL 2 mediante la aplicación de un conjunto de mapeos. En consecuencia, las unidades experimentales que aplicaron esta alternativa debieron identificar las reglas de negocio en forma previa a la ejecución de las transformaciones, sin disponer de ninguna guía metodológica para ejecutar dicha tarea.

Los resultados obtenidos por el experimento confirman investigaciones previas en el área: las técnicas basadas en UML son altamente efectivas como medio de conceptualización de "ontologías livianas" (lightweight ontologies) (Gómez-Pérez y otros, 2004). En cambio, es interesante notar el potencial del lenguaje SBVR para expresar nociones complejas de un dominio de interés: mientras las ontologías desarrollas aplicando el enfoque basado en mapeos aprovechan la totalidad del poder expresivo del lenguaje OWL, las desarrolladas mediante el enfoque UML sólo comprenden los aspectos básicos de OWL.

La Tabla B.6 muestra el poder expresivo de cada una de las ontologías mientras la Tabla B.7 presenta los constructores de la Lógica Descriptiva y sus nombres asociados (Baader y otros, 2003).

Las ontologías desarrolladas fueron examinadas por un experto del dominio para

su posterior evaluación de calidad, a fin de obtener cierta retroalimentación respecto a la brecha semántica existente entre la realidad y el modelado realizado. Más allá de las diferencias esperables entre cada modelado de la misma realidad, la totalidad de las ontologías representaban el dominio en forma adecuada.
Aunque cualquier experimento de laboratorio sufre una cierta falta de realismo, un estudio de campo de estas características resulta complejo de diseñar y ejecutar. Debe notarse además la limitación existente en la generalización de los resultados obtenidos, dada la naturaleza exploratoria del experimento. Sin embargo, incluso con tales limitaciones, el experimento ha permitido a los autores formular nuevas hipótesis as ser evaluadas en experimentos subsecuentes (Apéndice C).

Expresividad DL		
Enfoque SBVR	Ontología 1	$\mathcal{ALCHQ}(\mathcal{D})$
	Ontología 2	$\mathcal{ALCOQ}(\mathcal{D})$
	Ontología 3	$\mathcal{ALCHOQ}(\mathcal{D})$
	Ontología 4	$\mathcal{ALCIQ}(\mathcal{D})$
	Ontología 5	$\mathcal{ALEF}(\mathcal{D})$
Enfoque UML	Ontología 6	$\mathcal{ALCHI}(\mathcal{D})$
	Ontología 7	$\mathcal{AL}(\mathcal{D})$
	Ontología 8	$\mathcal{ALF}(\mathcal{D})$
	Ontología 9	$\mathcal{ALE}(\mathcal{D})$
	Ontología 10	$\mathcal{ALEHF}(\mathcal{D})$

Tabla B.6: Expresividad de la Lógica Descriptiva por ontología y por enfoque aplicado

B.8. Conclusiones

El presente apéndice describe un experimento exploratorio que compara la performance de estudiantes de ingeniería aplicando técnicas basadas en UML y SBVR para el desarrollo de ontologías. La utilización del test MWW ha permitido afirmar que no existen diferencias estadísticamente significativas entre los participantes del experimento en lo que refiere a conocimientos previos en las áreas involucradas en la ejecución del experimento; lo cual permite descartar dicho aspecto en la comparación de la calidad de las ontologías resultantes.
La performance de los enfoques considerados ha sido evaluada mediante las dimensiones de calidad de las ontologías definidas por OQuaRE, un marco de trabajo automatizable basado en el estándar SQuaRE para la evaluación de

Constructor	Sintaxis	Lenguaje
Concepto	A	$\mathcal{FL}_0$
Nombre de rol	R	$\mathcal{FL}_0$
Intersección	C ∩ D	$\mathcal{FL}_0$
Restricción de valor	∀ R.C	$\mathcal{FL}^-$
Cuantificación existencial limitada	∃ R	$\mathcal{AL}$
Universal	⊤	$\mathcal{S}$
Vacío	⊥	$\mathcal{S}$
Negación atómica	¬ A	$\mathcal{S}$
Negación	¬ C	$\mathcal{C}$
Unión	C ∪ D	$\mathcal{U}$
Restricción existencial	∃ R.C	$\mathcal{E}$
Restricción numérica	$(\geq nR)$ $(\leq nR)$	$\mathcal{N}$
Nominales	$\{a_1...a_n\}$	$\mathcal{O}$
Jerarquía de rol	$(R \subseteq S)$	$\mathcal{H}$
Rol inverso	R^-	$\mathcal{I}$
Restricción cuantificada numéricamente	$(\geq nR.C)$ $(\leq nR.C)$	$\mathcal{Q}$

Tabla B.7: Constructores DL y nombres de lenguaje: A refiere a conceptos atómicos, C y D a cualquier definición de concepto, R a roles atómicos y S a definición de roles

calidad de software. El test MWW también ha sido utilizado en la evaluación de las diferencias en las medidas de calidad de las ontologías, concluyendo que no existen diferencias estadísticamente significativas entre las técnicas evaluadas en ninguna de las dimensiones de calidad consideradas. Tales resultados permiten subrayar la factibilidad técnica de mapear expresiones de negocio SBVR a sentencias ontológicas OWL 2 al verificar la hipótesis del experimento: las ontologías desarrolladas por medio de un enfoque basado en SBVR al menos igualan la calidad de las ontologías desarrolladas mediante un método basado en UML.

Apéndice C

Evaluación empírica del mapeo de reglas de negocio para el desarrollo de ontologías

Este apéndice presenta una réplica diferenciada del experimento descripto en el Apéndice B, permitiendo evaluar el grado de generalización de los resultados obtenidos con respecto a la factibilidad técnica de SBVR como lenguaje de modelado para el desarrollo de ontologías.

Mientras un experimento aislado es útil por sí mismo, los estudios replicados resultan claves en el diseño de estudios científicos dado la naturaleza repetible del conocimiento científico. Además, la primera réplica es la más importante dado que establece si una generalización más amplia es posible. La idea de que una réplica implica la repetición exacta del estudio original constituye un error. En la práctica, la repetición de un experimento debería consistir en llevar a cabo un estudio distinto y evaluar si los resultados se mantienen o no. Esto permite distinguir dos tipos generales de réplicas: (1) las similares y (2) las diferenciadas. Las réplicas similares permiten establecer rápidamente si un resultado se repite o no. En cambio, el objetivo de un réplica diferenciada consiste en extender el rango de condiciones bajo las cuales los resultados se mantienen (Lindsay y Ehrenberg, 1993).

En el experimento descripto en el Apéndice B la hipótesis a evaluar fue:

Hipótesis 1: *Las ontologías desarrolladas por medio de un enfoque basado en la utilización del lenguaje SBVR al menos igualan la calidad de las ontologías desarrolladas mediante UML, considerando que las tareas de desarrollo son ejecutadas por recursos humanos con perfil de ingenieros de software sin*

conocimientos previos en ingeniería ontológica.
UML ha sido la base de diversas propuestas metodológicas para el desarrollo de ontologías; dada su amplia aceptación en la comunidad de ingeniería de software, su representación gráfica estandarizada de los modelos, la gran disponibilidad de herramientas que le brindan soporte y la naturaleza extensible del lenguaje. Pero la falta de una semántica precisa de conjuntos y la ausencia de una teoría de modelos ha dado origen a la definición de ODM, el cual ataca tales inconvenientes mediante un conjunto de meta-modelos formales, perfiles y mapeos.

C.1. Objetivo e hipótesis

El experimento descripto en el presente trabajo analiza la factibilidad técnica del mapeo de expresiones de negocios en sentencias ontológicas con el propósito de extender el rango de condiciones bajo las cuales se mantienen los resultados del primer experimento, evaluando la siguiente hipótesis:
Hipótesis 2: *Las ontologías desarrolladas por medio de un enfoque basado en la utilización del lenguaje SBVR al menos igualan la calidad de las ontologías desarrolladas mediante ODM, considerando que las tareas de desarrollo son ejecutadas por recursos humanos con perfil de ingenieros de software sin conocimientos previos en ingeniería ontológica.*
La hipótesis es evaluada mediante el análisis de calidad de las ontologías desarrolladas por distintos grupos de estudiantes avanzados de ingeniería.

C.2. Enfoques bajo estudio

A continuación se describen las dos propuestas consideradas en el experimento.

- El desarrollo de ontologías mediante la aplicación de ODM (OMG, 2009).
- El desarrollo de ontologías mediante la aplicación del enfoque de mapeo basado en SBVR propuesto en Reynares y otros (2013) y Reynares y otros (2014c).

ODM es una familia de meta-modelos MOF (OMG, 2014a), mapeos entre dichos meta-modelos, y mapeos desde y hacia el lenguaje UML (OMG, 2011), además de un conjunto de perfiles que permiten el modelado de ontologías mediante herramientas basadas en UML. Los meta-modelos reflejan la sintaxis abstracta de varios lenguajes estándares utilizados en el modelado conceptual y la representación de conocimiento. Sin embargo, lo que hace a un buen modelado de software orientado a objetos no necesariamente hace a una buena ontología: una vez que un modelo particular ha sido transformado en una ontología, es necesario tomar precauciones que permitan asegurar que el modelo resultante soportará las aserciones requeridas. A menudo son necesarias reestructuraciones significativas de la ontología para satisfacer tal punto. ODM provee dos maneras de relacionar los modelos a fin de superar ese problema: (1) mediante perfiles UML y (2) mediante mapeos entre los modelos.

El objetivo de un perfil UML es brindar un puente entre UML y las comunidades de representación de conocimiento en una base semántica bien fundamentada, lo cual constituye un medio de relacionar enfoques de software y de modelado lógico para representar conocimiento. ODM define los perfiles para la primera versión de OWL (W3C, 2004a), RDFS (W3C, 2014), y Topic Maps (ISO, 2003)). Aunque los perfiles proporcionan ciertas facilidades para que los usuarios puedan utilizar UML como base para el desarrollo de ontologías para un lenguaje de representación de conocimiento determinado, no facilitan una transformación completa a través del conjunto de paradigmas de representación incluidos en los metamodelos ODM. Tales necesidades son satisfechas por los mapeos de un meta-modelo a otro.

Por otra parte, el enfoque de mapeo de reglas de negocio propuesto por Reynares y otros (2013) y Reynares y otros (2014c) permite la generación automatizable de una ontología OWL 2 (W3C, 2009a) mediante la aplicación de un conjunto de transformaciones sobre las especificaciones SBVR de un dominio. SBVR ha sido concebido para expertos del negocio y diseñado para ser utilizado a fines de satisfacer los requerimientos del negocio. La naturaleza lingüística del lenguaje permite la expresión del conocimiento del negocio mediante sentencias en lugar de diagramas, lo cual responde a la idea de que los diagramas permiten describir estructuralmente los conceptos pero no resultan prácticos a los fines de definir vocabularios y expresar reglas de negocio. SBVR está basado en lógica

de predicados de primer orden con algunas extensiones restringidas en lógica de alto nivel y lógica modal.

C.3. Contexto y unidades experimentales

El experimento fue realizado en el contexto del curso de grado denominado *"Desarrollo de Sistemas de Información basados en Ontologías"*, el cual forma parte del último nivel de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional en la ciudad de Santa Fe, Argentina. Las unidades experimentales consistieron en 10 grupos de igual tamaño, conformados por 20 estudiantes del curso. Un proceso aleatorio fue utilizado para la generación de muestras de igual tamaño y la asignación de las unidades experimentales a los distintos enfoques. De esta forma, 5 grupos aplicaron el enfoque de mapeo basado en SBVR, mientras los restantes 5 siguieron la propuesta ODM Los participantes obtuvieron créditos académicos por formar parte del experimento. Además, respondieron una ncuesta anónima acerca de su conocimiento previo en relación a los temas involucrados en el experimento, evaluando las mismas en un rango de valores de 1 - conocimientos previos nulos - a 10 conocimientos previos de nivel profesional -.

Aunque la naturaleza anónima de la encuesta buscaba obtener respuestas sinceras, es necesario considerar el sesgo subjetivo presente en cualquier auto evaluación. En las Figuras C.1-C.3 se muestran las comparaciones de frecuencia sobre conocimientos previos en (P1) el uso de artefactos UML, (P2) el desarrollo de sentencias lógicas y (P3) la ingeniería ontológica, respectivamente1. Para evaluar las diferencias en las respuestas de los enfoques comparados fue utilizado el test de Mann-Whitney-Wilcoxon (MWW). Este test (también denominado estadístico U, o simplemente U) es un test no paramétrico de la hipótesis nula - donde dos poblaciones resultan iguales - contra un a hipótesis alternativa - donde una población particular tiende a tener valores mayores que la otra -. El test posee una eficiencia mayor al test *"t"* en distribuciones no-normales y una eficiencia cercana a la del test *"t"* en distribuciones normales Mann y Whitney (1947)(Fay y Proschan, 2010).

La Tabla C.1 muestra los intervalos críticos de U para dos muestras de igual

tamaño A y B (donde el tamaño $n = 10$), para test direccionales y no direccionales y para los niveles de significancia más comúnmente utilizados. La hipótesis nula- ambos poblaciones tienden a poseer los mismo valores - se acepta si los valores observados de U se encuentran en el rango comprendido entre los límites superiores e inferiores. La Tabla C.2 presenta el valor observado de U para ambos enfoques en cada una de las preguntas de la encuesta. Los valores observados permiten concluir que no existe diferencia estadísticamente significativa entre ambos enfoques en relación a su conocimiento previo sobre P1, P2, y P3. Tales resultados permitieron concluir que ambos grupos poseían el mismo nivel de conocimientos previos en relación a los temas involucrados en el experimento, los cuales se asemejan al conocimiento que un ingeniero de software sin formación previa en ingeniería ontológica posee inicialmente.

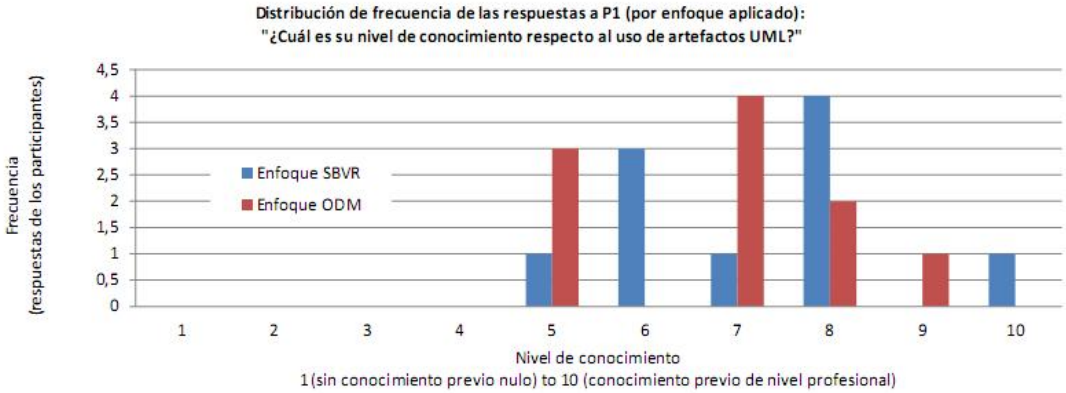

Figura C.1: Distribución de frecuencia de las respuestas a P1 (por enfoque aplicado): "¿Cuál es su nivel de conocimiento respecto al uso de artefactos UML?"

Intervalos críticos de U			
	Nivel de significancia para:		
	Tests Direccionales		
	0.05	0.025	0.01
	Tests no-Direccionales		
	-	0.05	0.02
Límite inferior	27	23	19
Límite superior	73	77	81

Tabla C.1: Intervalos críticos de U para dos muestras de tamaño $n = 10$)

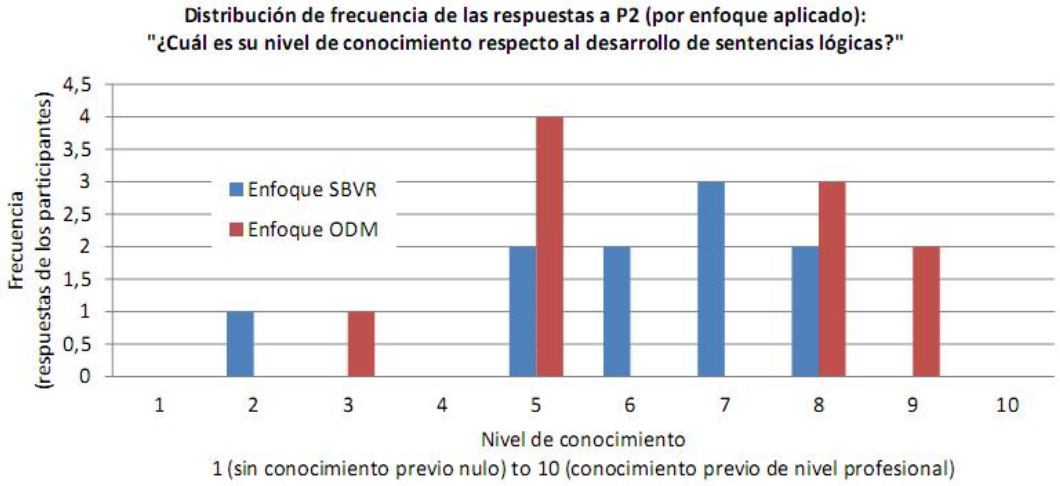

Figura C.2: Distribución de frecuencia de las respuestas a P2 (por enfoque aplicado): "¿Cuál es su nivel de conocimiento respecto al desarrollo de sentencias lógicas?"

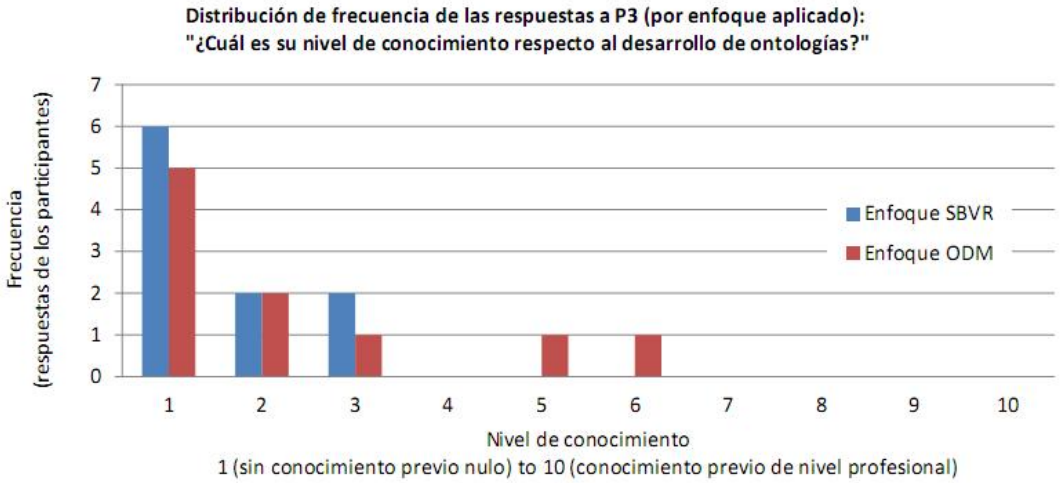

Figura C.3: Distribución de frecuencia de las respuestas a P3 (por enfoque aplicado): "¿Cuál es su nivel de conocimiento respecto al desarrollo de ontologías?"

	Valores de U	
	Enfoque SBVR	Enfoque ODM
Conocimientos previos en:		
(P1) el uso de artefactos UML	42.5	57.5
(P2) el desarrollo de sentencias lógicas	56	44
(P3) ingeniería ontológica	58	42

Tabla C.2: Valores de U por enfoque aplicado

C.4. Tarea y materiales

La tarea del experimento consistió en la especificación ontológica de las políticas de asignación de becas estudiantiles de la universidad, especificadas originalmente en un documento oficial de la institución y descriptas en lenguaje natural. Con respecto a las herramientas, el enfoque basado en SBVR utilizó editores de texto convencionales para modelar el dominio del negocio, mientras aquellos que aplicaron ODM utilizaron la herramienta Microsoft Visio para el desarrollo de los modelos gráficos. Ambos enfoques implementaron la ontología en OWL 2 por medio de Protégé, un editor libre y de código abierto.

C.5. Resultados

La performance de los dos enfoques fue comparada mediante la evaluación de la calidad de las ontologías desarrolladas. Tal evaluación fue realizada por los autores de este trabajo mediante OQuaRE (Duque-Ramos y otros, 2011)(Duque-Ramos y otros, 2013), un marco de trabajo basado en el estándar SQuaRE para la evaluación de la calidad del software (ISO, 2005)). OQuaRE evalúa las ontologías independientemente de cualquier proceso de desarrollo particular, lo cual consituye un proceso objetivo de evaluación capaz de ser replicado.

OQuaRE define un modelo de calidad y sus respectivas métricas para la evaluación de las ontologías. El modelo de calidad es dividido en dimensiones - o características - organizadas en sub-dimensiones - o sub-características -, las cuales son evaluadas por medio de un conjunto de métricas. OQuaRE define el criterio para normalizar los valores de las métricas en un rango de 1 a 5: calidad no aceptable es asociada al valor 1, la mínima calidad aceptable responde al valor 3, y el valor 5 excede los requerimientos de calidad. Luego de la normalización, el puntaje asociado a cada una de las características es la media de las sub-características asociadas, cuyo valor es calculado como la media de sus métricas. El resultado final de la evaluación de calidad consiste en el conjunto de valores para cada una de las características evaluadas. Esto permite la identificación de las fortalezas y debilidades de las ontologías consideradas

en lugar de señalar la *"mejor"* ontología, como se propone en los trabajos de Lozano-Tello y Gómez-Pérez (2004), Park y otros (2011), y Vrandečić (2010).
Las dimensiones de calidad evaluadas en el presente experimento se describen a continuación:

- La *dimensión estructural* involucra propiedades formales y semánticas que resultan importantes al evaluar ontologías, tales como consistencia, nivel de formalización, redundancia o complejidad.
- La *dimensión de adecuación funcional* refiere al grado de alineamiento de la ontología para su finalidad prevista, de acuerdo a las categorías identificadas por (Stevens y Lord, 2009).
- La *dimensión de mantenibilidad* está relacionada a la capacidad de las ontologías de ser modificadas de acuerdo a cambios en el entorno, en los requerimientos o en las especificaciones funcionales.
- La *dimensión de compatibilidad* refiere a la habilidad de dos o más ontologías para intercambiar información y/o cumplir su función mientras comparten el mismo entorno de hardware o software. Esta dimensión puede ser evaluada sobre una única ontología - aunque intuitivamente involucra las propiedades de más de una -, dado que es cuantitativamente evaluada por medio de un conjunto de métricas aplicadas a cada ontología en forma separada.
- La *dimensión de transferibilidad* es el grado con el cual la ontología puede ser transferida de un entorno a otro.
- La *dimensión de operatividad* refiere al esfuerzo necesario para utilizar la ontología por un determinado conjunto de usuarios.
- La *dimensión de confiabilidad* es la capacidad de la ontología de mantener su nivel de performance bajo condiciones establecidas, por un período dado de tiempo.

Tres métricas no fueron consideradas en la evaluación de las dimensiones de calidad. Las métricas denominadas *riqueza de las anotaciones* - número medio

de anotaciones por clase - y *riqueza de clases* - número medio de instancias por clase - no fueron evaluadas dado que la anotación e instanciación de ontologías no formaba parte del experimento. La métrica denominada *riqueza de atributos* - número medio de atributos por clase - no fue evaluada dado que los atributos no forman parte de la especificación estructural de lenguaje de implementación de las ontologías (W3C, 2009b).

OQuaRE define también las dimensiones denominadas *eficiencia de performance* y *calidad de uso*. La eficiencia de performance expone la relación entre el nivel de performance de la ontología y los recursos utilizados bajo condiciones establecidas, tomando en cuenta elementos tales como tiempo de respuesta o consumo de memoria. La calidad en uso refiere al grado con el cual la ontología satisface los requerimientos de usuarios específicos. Sin embargo, tales dimensiones no fueron consideradas dado la inexistencia de métricas asociadas a sus correspondientes sub-características. La Figura C.4 muestra los niveles de calidad de las ontologías desarrolladas por medio del enfoque basado en el mapeo de expresiones SBVR. La Figura C.5 muestra los niveles de calidad de las ontologías desarrolladas mediante el enfoque UML.

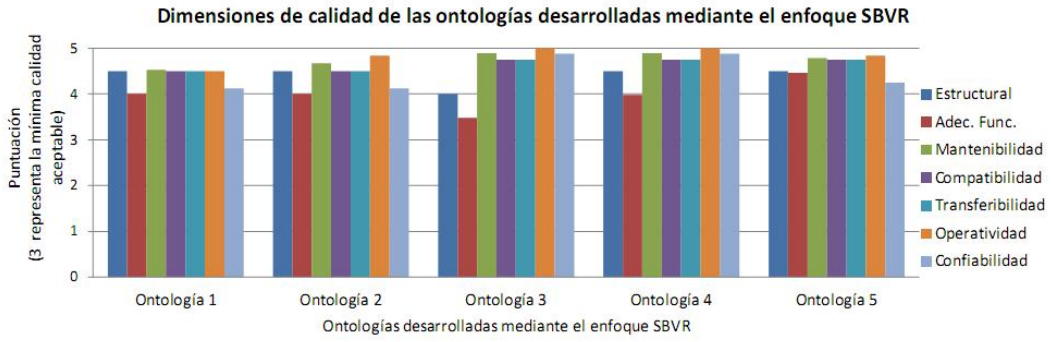

Figura C.4: Puntuación de las dimensionaes de calidad de las ontologías desarrolladas mediante el enfoque SBVR

C.6. Análisis de resultados

Un rápido vistazo a las Figuras C.4 y C.5 permite reconocer un primer resultado importante: de acuerdo a OQuaRE, todas las ontologías superan la mínima calidad aceptable. La Tabla C.3 muestra el valor medio de cada una

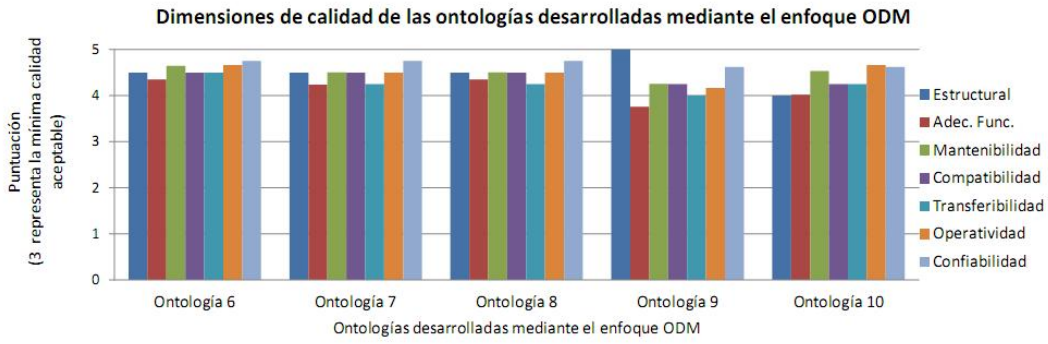

Figura C.5: Puntuación de las dimensiones de calidad de las ontologías desarrolladas mediante el enfoque ODM

de las dimensiones de calidad evaluadas: la primer columna muestra el nivel de calidad de las ontologías desarrolladas mediante el enfoque basado en SBVR mientras la segunda presenta los valores de las ontologías obtenidas por medio de ODM. Además, la última fila de la tabla muestra el valor medio de la calidad de la ontología de acuerdo al enfoque seguido. Una comparación fila por fila permite observar que las medias de las dimensiones de calidad son muy similares entre ambos enfoques, incluso en sus valores medios globales. Nuevamente, el test MWW permitió evaluar las diferencias entre los valores obtenidos. La Tabla C.4 muestra los intervalos críticos de U para dos muestras de tamaño $n = 5$, tanto para test direccionales como no-direccionales y para los niveles de significancia más comúnmente considerados. Finalmente, la Tabla C.5 muestra los valores de U para cada enfoque en cada una de las dimensiones de calidad evaluadas. Los valores observados permiten concluir que no existen diferencias estadísticamente significativas entre ambos enfoques, en ninguna de las dimensiones evaluadas.

	Enfoque SBVR	Enfoque ODM
Estructural	4.40	4.50
Adecuación Funcional	3.99	4.14
Mantenibilidad	4.75	4.49
Compatibilidad	4.65	4.40
Transferibilidad	4.65	4.25
Operatividad	4.83	4.50
Confiabilidad	4.45	4.70
Media Global	4.53	4.43

Tabla C.3: Valores medios por dimensión de calidad y enfoque aplicado

Intervalos críticos de U			
	Nivel de significancia para:		
	Tests Direccionales		
	0.05	0.025	0.01
	Tests no-Direccionales		
	-	0.05	0.02
Límite inferior	4	2	1
Límite superior	21	23	24

Tabla C.4: Intervalos críticos de U para dos muestras de tamaño $n = 5$

	Valores de U	
	Enfoque SBVR	Enfoque ODM
Estructural	14.5	10.5
Adecuación Funcional	17	8
Mantenibilidad	2	23
Compatibilidad	11.5	13.5
Transferibilidad	3	22
Operatividad	3	22
Confiabilidad	15	10

Tabla C.5: Valores de U por dimensión de calidad y enfoque aplicado

C.7. Discusión

El experimento presentado se encuentra sesgado en forma negativa con respecto al enfoque basado en el mapeo de expresiones SBVR, dado que los participantes poseían conocimientos previos en el uso de artefactos UML mientras el conocimiento acerca del lenguaje SBVR era inexistente.

Resulta interesante destacar el potencial del lenguaje SBVR para expresar nociones complejas de un dominio de interés: mientras las ontologías desarrollas aplicando el enfoque basado en mapeos aprovechan la totalidad del poder expresivo del lenguaje OWL, las desarrolladas mediante ODM sólo comprenden los aspectos básicos de OWL. Aunque no se vea reflejado en los resultados formales del experimento, un aspecto importante de resaltar lo constituyen las sensaciones de los participantes del experimento. Aquellos que aplicaron el enfoque SBVR manifestaron la simpleza del proceso de desarrollo basado en la descripción de un dominio por medio del lenguaje natural. En cambio, aquellos que utilizaron ODM expresaron su descontento con un proceso que consideraron dificultoso de ejecutar y cuyos resultados mostraban una complejidad y dimensión que excedía

la del dominio real que era objeto del modelado.

La Tabla C.6 muestra el poder expresivo de cada una de las ontologías mientras la Tabla C.7 presenta los constructores de la Lógica Descriptiva y sus nombres asociados. Las ontologías desarrolladas fueron examinadas por un experto del dominio para su posterior evaluación de calidad, a fin de obtener cierta retroalimentación respecto a la brecha semántica existente entre la realidad y el modelado realizado. Más allá de las diferencias esperables entre cada modelado de la misma realidad, la totalidad de las ontologías representaban el dominio en forma adecuada. Sin embargo, una evaluación más sistemática y compleja eliminaría cualquier sesgo subjetivo.

Aunque cualquier experimento de laboratorio sufre una cierta falta de realismo, un estudio de campo de estas características resulta complejo de diseñar y ejecutar. En cuanto a los sujetos experimentales, existen buenas razones para realizar experimentos con estudiantes. Por ejemplo, a fin de evaluar diseños experimentales y posibilitar un rápido descubrimiento de resultados negativos en los cuales no tenga sentido profundizar (Tichy, 2000). Los estudiantes también resultan representativos de profesionales de escasa experiencia (Sjøberg y otros, 2005), lo cual es actualmente la condición general de los profesionales de ingeniería de software en relación al uso de tecnologías semánticas.

Expresividad DL		
Enfoque SBVR	Ontología 1	$\mathcal{ALCHIC}(\mathcal{D})$
	Ontología 2	$\mathcal{ALCIC}(\mathcal{D})$
	Ontología 3	$\mathcal{ALCRQ}(\mathcal{D})$
	Ontología 4	$\mathcal{ALIQ}(\mathcal{D})$
	Ontología 5	$\mathcal{ALCROIQ}(\mathcal{D})$
Enfoque ODM	Ontología 6	$\mathcal{ALU}(\mathcal{D})$
	Ontología 7	$\mathcal{ALUI}(\mathcal{D})$
	Ontología 8	$\mathcal{ALCF}(\mathcal{D})$
	Ontología 9	$\mathcal{ALCO}(\mathcal{D})$
	Ontología 10	$\mathcal{ALCIQ}(\mathcal{D})$

Tabla C.6: Expresividad de la Lógica Descriptiva por ontología y por enfoque aplicado

Constructor	Sintaxis	Lenguaje			
Concepto	A	$\mathcal{FL}_0$	$\mathcal{FL}^-$	$\mathcal{AL}$	$\mathcal{S}$
Nombre de rol	R				
Intersección	C ∩ D				
Restricción de valor	∀ R.C				
Cuantificación existencial limitada	∃ R				
Universal	⊤				
Vacío	⊥				
Negación atómica	¬ A				
Negación	¬ C	$\mathcal{C}$			
Unión	C ∪ D	$\mathcal{U}$			
Restricción existencial	∃ R.C	$\mathcal{E}$			
Restricción numérica	$(\geq nR)$ $(\leq nR)$	$\mathcal{N}$			
Nominales	$\{a_1...a_n\}$	$\mathcal{O}$			
Jerarquía de rol	$(R \subseteq S)$	$\mathcal{H}$			
Rol inverso	R^-	$\mathcal{I}$			
Restricción cuantificada numéricamente	$(\geq nR.C)$ $(\leq nR.C)$	$\mathcal{Q}$			

Tabla C.7: Constructores DL y nombres de lenguaje: A refiere a conceptos atómicos, C y D a cualquier definición de concepto, R a roles atómicos y S a definición de roles

C.8. Conclusiones

Este apéndice presenta una primera réplica diferenciada del experimento previo descripto en el Apéndice B, evaluando el grado de generalización de los resultados originales y extendiendo el rango de condiciones bajo las cuales tales resultados se mantienen. Los resultados obtenidos en la réplica presentada permiten sostener aquellos obtenidos en el estudio original. La factibilidad técnica de mapear expresiones SBVR en ontologías OWL 2 ha sido demostrada, incluso variando el tamaño y composición de las unidades experimentales y los enfoques bajo estudio. Tales resultados refuerzan el potencial de los mapeos SBVR a OWL 2 como una técnica de desarrollo de ontologías digno de un estudio más profundo.

Bibliografía

Alberts, R. y Franconi, E. (2012). «An Integrated Method Using Conceptual Modelling to Generate an Ontology-based Query Mechanism.» En: Pavel Klinov y Matthew Horridge (Eds.), *OWL: Experiences and Directions Workshop 2012 (OWLED)*, volumen 849 de *CEUR Workshop Proceedings*. Greece.

Alexander, Christopher (1979). *The timeless way of building (Vol 1 of Center for Environmental Structure Series)*. Oxford University Press, New York.

Alexander, Christopher; Ishikawa, Sara y Silverstein, Murray (1977). *A pattern language: towns - buildings - construction, (Vol 2 of Center for Environmental Structure Series)*. Oxford University Press, New York.

Alrajeh, D.; Ray, O.; Russo, A. y Uchitel, S. (2009). «Using abduction and induction for operational requirements elaboration». *Journal of Applied Logic*, **7(3)**, pp. 275 – 288. ISSN 1570-8683. doi: http://dx.doi.org/10.1016/j.jal.2008.10.002.
http://www.sciencedirect.com/science/article/pii/S1570868308000633

Anand, Saswat; Burke, Edmund K.; Chen, Tsong Yueh; Clark, John; Cohen, Myra B.; Grieskamp, Wolfgang; Harman, Mark; Harrold, Mary Jean y McMinn, Phil (2013). «An orchestrated survey of methodologies for automated software test case generation». *Journal of Systems and Software*, **86(8)**, pp. 1978 – 2001. ISSN 0164-1212. doi: http://dx.doi.org/10.1016/j.jss.2013.02.061.
http://www.sciencedirect.com/science/article/pii/S0164121213000563

Arcuri, Andrea y Yao, Xin (2014). «Co-evolutionary automatic programming

for software development». *Information Sciences*, **259(0)**, pp. 412 – 432. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2009.12.019.
http://www.sciencedirect.com/science/article/pii/S0020025509005490

Auer, Sören y Herre, Heinrich (2007). «RapidOWL An Agile Knowledge Engineering Methodology». En: *Perspectives of Systems Informatics*, pp. 424–430. Springer.

Ayub, M.C; Cian, A.N.; Caliusco, M.L. y Reynares, E. (2013). «An Experience Report on using the EDON Method for Building a Team Recommender System». En: *Proceedings of 41st Argentine Conference on Informatics - 13th Argentine Symposium on Software Engineering (JAIIO - ASSE 2012)*, Córdoba, Argentina.
http://www.42jaiio.org.ar/proceedings/simposios/trabajos/ASSE/02.pdf

Ayub, M.C; Cian, A.N.; Caliusco, M.L. y Reynares, E. (2014). «Developing an Ontology-Based Team Recommender System using EDON Method: An Experience Report», **13(1)**, pp. 1–13.
http://www.sadio.org.ar/wp-content/uploads/2014/06/1-Caliusco.pdf

Baader, Franz; Calvanese, Diego; McGuinness, Deborah L.; Nardi, Daniele y Patel-Schneider, Peter F. (Eds.) (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA. ISBN 0-521-78176-0.

Beck, Kent (2000). *Extreme programming explained: embrace change*. Addison-Wesley Professional.

Berners-Lee, T.; Hendler, J. y Lassila, O. (2001). «The Semantic Web». *Scientific American*, **284(5)**, pp. 34–43.

Beydoun, G.; Low, G.; Tran, N. y Bogg, P. (2011). «Development of a peer-to-peer information sharing system using ontologies». *Expert Systems with Applications*, **38(8)**, pp. 9352–9364. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2011.01.104.
http://www.sciencedirect.com/science/article/pii/S0957417411001242

Blomqvist, E.; Gangemi, A. y Presutti, V. (2009). «Experiments on pattern-based ontology design». En: Y. Gil y N.F. Noy (Eds.), *5th International Conference on Knowledge Capture (K-CAP 2009)*, pp. 41–48. ACM, California, USA. ISBN 978-1-60558-658-8.

Breitman, Karin Koogan y do Prado Leite, Julio Cesar Sampaio (2004). «Lexicon based ontology construction». En: *Software Engineering for Multi-Agent Systems II*, pp. 19–34. Springer.

Carroll, Jeremy J; Dickinson, Ian; Dollin, Chris; Reynolds, Dave; Seaborne, Andy y Wilkinson, Kevin (2004). «Jena: implementing the semantic web recommendations». En: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pp. 74–83. ACM.

Catelani, Marcantonio; Ciani, Lorenzo; Scarano, Valeria L. y Bacioccola, Alessandro (2011). «Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use». *Computer Standards & Interfaces*, **33(2)**, pp. 152 – 158. ISSN 0920-5489. doi: http://dx.doi.org/10.1016/j.csi.2010.06.006. `http://www.sciencedirect.com/science/article/pii/S092054891000084X`

Ceravolo, P.; Fugazza, C. y Leida, M. (2007). «Modeling Semantics of Business Rules». En: *EcoSystems, Digital Technologies Conference. DEST 07*, pp. 171–176.

Clark, P.; Thompson, J. y Porter, B. (2004). «Knowledge Patterns». En: *Handbook on Ontologies*, pp. 191–207. Springer Berlin Heidelberg.

Corcho, Óscar; Gómez-Pérez, Asunción; González-Cabero, Rafael y Suárez-Figueroa, M. Carmen (2004). «ODEval: A Tool for Evaluating RDF(S), DAML+OIL, and OWL Concept Taxonomies». En: Max Bramer y Vladan Devedzic (Eds.), *Artificial Intelligence Applications and Innovations*, volumen 154 de *IFIP International Federation for Information Processing*, pp. 369–382. Springer US.

de Nicola, A.; Missikoff, M. y Navigli, R. (2009). «A software engineering

approach to ontology building». *Information Systems*, **34(2)**, pp. 258–275. ISSN 0306-4379.

Duque-Ramos, Astrid; Fernández-Breis, Jesualdo Tomás; Iniesta, Miguela; Dumontier, Michel; EgañA Aranguren, Mikel; Schulz, Stefan; Aussenac-Gilles, Nathalie y Stevens, Robert (2013). «Evaluation of the {OQuaRE} framework for ontology quality». *Expert Systems with Applications*, **40(7)**, pp. 2696 – 2703. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2012.11.004.
http://www.sciencedirect.com/science/article/pii/S0957417412012146

Duque-Ramos, Astrid; Fernández-Breis, Jesualdo Tomás; Stevens, Robert y Aussenac-Gilles, Nathalie (2011). «OQuaRE: A SQuaRE-based Approach for Evaluating the Quality of Ontologies». *Journal of Research and Practice in Information Technology*, **43(2)**, pp. 159–176.

Euzenat, Jérôme; Shvaiko, Pavel y otros (2007). *Ontology matching*. Springer.

Farahbod, Roozbeh; Gervasi, Vincenzo y Glasser, Uwe (2014). «Executable formal specifications of complex distributed systems with CoreASM». *Science of Computer Programming*, **79(0)**, pp. 23 – 38. ISSN 0167-6423. doi: http://dx.doi.org/10.1016/j.scico.2012.02.001.
http://www.sciencedirect.com/science/article/pii/S0167642312000160

Farré, Carles; Queralt, Anna; Rull, Guillem; Teniente, Ernest y Urpí, Toni (2013). «Automated reasoning on {UML} conceptual schemas with derived information and queries». *Information and Software Technology*, **55(9)**, pp. 1529 – 1550. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2013.02.010.
http://www.sciencedirect.com/science/article/pii/S0950584913000438

Fay, Michael P y Proschan, Michael A (2010). «WilcoxonMannWhitney or ttest? On assumptions for hypothesis tests and multiple interpretations of decision rules». *Statistics surveys*, **4**, p. 1.

Franconi, E. y Mosca, A. (2012). «The formalisation of ORM2 and its encoding in OWL2». *Informe técnico*, KRDB Research Centre

Technical Report KRDB12-2, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy.

Fuchs, N.E. (1992). «Specifications are (preferably) executable». *Software Engineering Journal*, **7(5)**, pp. 323–334. ISSN 0268-6961.

Gamma, Erich; Helm, Richard; Johnson, Ralph y Vlissides, John (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.

Gangemi, A. y Presutti, V. (2009). *Ontology Design Patterns*. pp. 221–243. Springer, Berlin, 2ª edición.

Gangemi, Aldo; Catenacci, Carola; Ciaramita, Massimiliano y Lehmann, Jos (2006). «Modelling Ontology Evaluation and Validation». En: York Sure y John Domingue (Eds.), *The Semantic Web: Research and Applications*, volumen 4011 de *Lecture Notes in Computer Science*, pp. 140–154. Springer Berlin Heidelberg.

Gaspoz, C.; Bertossi, V.; Reynares, E. y Caliusco, M.L. (2013). «Applying EDON Methodology and SBVR2OWL Mappings for Building an Ontology-Aware Software». En: *Proceedings of XIX Congreso Argentino de Ciencias de la Computación CACIC 2013 - V Workshop Innovación en Sistemas de Software*, Mar del Plata, Argentina.
`http://sedici.unlp.edu.ar/handle/10915/31610`

Gómez-Pérez, A.; Fernández-López, M. y Corcho, O. (2004). *Ontological Engineering*. Springer/Heidelberg.

Grüninger, Michael y Fox, Mark S. (1995). «Methodology for the Design and Evaluation of Ontologies». En: *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, .

Guarino, Nicola y Welty, ChristopherA. (2009). «An Overview of OntoClean». En: Steffen Staab y Rudi Studer (Eds.), *Handbook on Ontologies*, International Handbooks on Information Systems, pp. 201–220. Springer Berlin Heidelberg. ISBN 978-3-540-70999-2.

Guizzardi, G.; Herre, H. y Wagner, G. (2002). «Towards Ontological Foundations for UML Conceptual Models». En: *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pp. 1100–1117. Springer-Verlag, London, UK, UK. ISBN 3-540-00106-9.

Hailpern, Brent y Tarr, Peri (2006). «Model-driven development: The good, the bad, and the ugly». *IBM systems journal*, **45(3)**, pp. 451–461.

Hepp, Martin (2007). «Possible ontologies: How reality constrains the development of relevant ontologies». *Internet Computing, IEEE*, **11(1)**, pp. 90–96.

Hevner, A.; March, S. T.; Park, J. y Ram, S. (2004). «Design science in information systems research». *MIS Quarterly*, **28(1)**, pp. 75–105.

Hoekstra, R. (2009). *Ontology Representation - Design Patterns and Ontologies that Make Sense*. volumen 197 de *Frontiers in Artificial Intelligence and Applications*. IOS Press. ISBN 978-1-60750-013-1.

Hristozova, Maia y Sterling, Leon (2002). «An eXtreme method for developing lightweight ontologies». En: *In Workshop on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, .

Iqbal, R.; Murad, M.A.A.; Mustapha, A. y Sharef, N.M. (2013). «An Analysis of Ontology Engineering Methodologies: A Literature Review». *Research Journal of Applied Sciences, Engineering and Technology*, **6(16)**, pp. 2993–3000.

Iscoe, N. (1991). «Domain Modeling-Evolving Research». En: *Proceedings of Knowledge-Based Software Engineering Conference*, pp. 234–236. IEEE.

ISO (2003). «International Organization for Standardization. Topic Maps. ISO/IEC 13250».
`http://www.isotopicmaps.org/`

ISO (2005). «International Organization for Standardization ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)».

Jacobson, Ivar; Booch, Grady; Rumbaugh, James; Rumbaugh, James y Booch, Grady (1999). *The unified software development process*. volumen 1. AddisonWesley Reading.

Kalyanpur, Aditya; Pastor, Daniel J; Battle, Steven y Padget, Julian (2004). «Automatic mapping of OWL ontologies into Java». En: *Proceedings of Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE)*, .

Karpovic, Jaroslav; Krisciuniene, Gintare; Ablonskis, Linas y Nemuraite, Lina (2014). «The Comprehensive Mapping of Semantics of Business Vocabulary and Business Rules (SBVR) to OWL 2 Ontologies». *Information Technology And Control*, **43(3)**, pp. 289–302.

Keet, C Maria y Artale, Alessandro (2008). «Representing and reasoning over a taxonomy of part–whole relations». *Applied Ontology*, **3(1)**, pp. 91–110.

Keim, Tobias (2007). «Extending the applicability of recommender systems: a multilayer framework for matching human resources». En: *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pp. 169–169. IEEE.

Keller, RichardM.; Rimon, Michal y Das, Aseem (1994). «A knowledge-based prototyping environment for construction of scientific modeling software». *Automated Software Engineering*, **1(1)**, pp. 79–128. doi: 10.1007/BF00871693.
`http://dx.doi.org/10.1007/BF00871693`

Kendall, E. y Linehan, M. H. (2013). «Mapping SBVR to OWL2». *Informe técnico*, Technical report, IBM Research Division, New York, NY.

Kitchenham, Barbara A; Pfleeger, Shari Lawrence; Pickard, Lesley M; Jones, Peter W; Hoaglin, David C.; El Emam, Khaled y Rosenberg, Jarrett (2002). «Preliminary guidelines for empirical research

in software engineering». *IEEE Transactions on Software Engineering*, **28(8)**, pp. 721–734.

Knublauch, Holger (2002). *An Agile Development Methodology for Knowledge-Based Systems Including a Java Framework for Knowledge Modeling and Appropriate Tool Support.* Universität Ulm, Fakultät für Informatik.

Leinberger, Martin; Scheglmann, Stefan; Lämmel, Ralf; Staab, Steffen; Thimm, Matthias y Viegas, Evelyne (2014). «Semantic web application development with liteq». En: *The Semantic Web - ISWC 2014*, pp. 212–227. Springer.

Lindsay, R Murray y Ehrenberg, Andrew SC (1993). «The design of replicated studies». *The American Statistician*, **47(3)**, pp. 217–228.

Lozano-Tello, Adolfo y Gómez-Pérez, Asunción (2004). «Ontometric: A method to choose the appropriate ontology». *Journal of Database Management*, **2(15)**, pp. 1–18.

Ma, Yinglong; Ma, Xinyv; Liu, Shaohua y Jin, Beihong (2009). «A Proposal for Stable Semantic Metrics Based on Evolving Ontologies». En: *International Joint Conference on Artificial Intelligence, 2009. IJCAI09.*, pp. 136–139. doi: 10.1109/JCAI.2009.42.

Mann, Henry B y Whitney, Donald R (1947). «On a test of whether one of two random variables is stochastically larger than the other». *The annals of mathematical statistics*, pp. 50–60.

Martínez, Agustín; Sosa, Santiago; Reynares, Emiliano y Caliusco, María Laura (2014). «Implementación de Sistemas de Información Basados en Ontologías: Análisis de Tecnologías». En: *Anales de 2do. Congreso Nacional de Ingeniería Informática/Sistemas de Información (CoNaIISI) - Simposio de Ingeniería en Sistemas y de Software*, pp. 1114–1118.

Masolo, C.; Borgo, S.; Gangemi, A.; Guarino, N. y Oltramari, A. (2003). «Ontology Library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003)».

Meth, Hendrik; Brhel, Manuel y Maedche, Alexander (2013). «The state of the art in automated requirements elicitation». *Information and Software Technology*, **55(10)**, pp. 1695 – 1709. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2013.03.008.
`http://www.sciencedirect.com/science/article/pii/S0950584913000827`

Nicholson, Jon; Eden, Amnon H.; Gasparis, Epameinondas y Kazman, Rick (2014). «Automated verification of design patterns: A case study». *Science of Computer Programming*, **80, Part B(0)**, pp. 211 – 222. ISSN 0167-6423. doi: http://dx.doi.org/10.1016/j.scico.2013.05.007.
`http://www.sciencedirect.com/science/article/pii/S0167642313001354`

OMG (2009). «Object Management Group. Ontology Definition Metamodel (ODM). Version 1.0».
`http://www.omg.org/spec/ODM/1.0`

OMG (2011). «Object Management Group. Unified Modeling Language (UML). Version 2.4.1».
`http://www.omg.org/spec/UML/2.4.1/`

OMG (2012). «Object Management Group. Object Constraint Language (OCL). Version 2.3.1».
`http://www.omg.org/spec/OCL/2.3.1/`

OMG (2013). «Object Management Group. Semantics of Business Vocabulary and Business Rules (SBVR). Version 1.1: Formal Specification».
`http://www.omg.org/spec/SBVR/1.1/`

OMG (2014a). «Object Management Group. MetaObject Facility (MOF). Version 2.4.2».
`http://www.omg.org/mof/`

OMG (2014b). «Object Management Group. Model Driven Architecture (MDA)».
`http://www.omg.org/mda/`

Oren, Eyal; Delbru, Renaud; Gerke, Sebastian; Haller, Armin y Decker, Stefan (2007). «ActiveRDF: Object-oriented semantic web

programming». En: *Proceedings of the 16th international conference on World Wide Web*, pp. 817–824.

Paar, Alexander y Vrandecic, Denny (2011). «Zhi# - OWL Aware Compilation». En: *Proceedings of The Semanic Web: Research and Applications - 8th Extended Semantic Web Conference Part II*, pp. 315–329. Springer.

Panti, M; Cucchiarelli, A y Valenti, S (1995). «Computer-Supported Domain MOdeling: A Step Toward Intelligent Case». *Managing Information and Communications in a Changing Global Environment*.

Park, Jinsoo; Oh, Sunjoo y Ahn, Joongho (2011). «Ontology selection ranking model for knowledge reuse». *Expert Systems with Applications*, **38(5)**, pp. 5133–5144.

Pinker, S. (2007). *The Stuff of Thought*. Allen Lane, Penguin Books, London, England.

Pohl, Klaus (2010). *Requirements engineering: fundamentals, principles, and techniques*. Springer.

Presutti, Valentina; Daga, Enrico; Gangemi, Aldo y Blomqvist, Eva (2009). «eXtreme design with content ontology design patterns». En: *Proceedings of Workshop on Ontology Patterns*, Citeseer, Washington, DC, USA.

Reiss, S.P. (2012). «Automatic programming as code search: A research agenda». En: *Search-Driven Development - Users, Infrastructure, Tools and Evaluation (SUITE), 2012 ICSE Workshop on*, pp. 1–4. doi: 10.1109/SUITE.2012.6225473.

Reynares, E.; Caliusco, M.L. y Galli, M.R. (2012). «EDON: A Method for Building an Ontology as Software Artefact». En: *41st Argentine Conference on Informatics - 13th Argentine Symposium on Software Engineering (JAIIO - ASSE 2012)*, La Plata, Buenos Aires, Argentina.

Reynares, E.; Caliusco, M.L. y Galli, M.R. (2013). «An Automatable Approach for SBVR to OWL 2 Mappings». En: *XVI Ibero-American Conference on Software Engineering (CIbSE 2013)*, Montevideo, Uruguay.

Reynares, E.; Caliusco, M.L. y Galli, M.R. (2014a). «Approaching the feasibility of SBVR as modeling language for ontology development: An exploratory experiment». *Expert Systems with Applications*, **41(4, Part 2)**, pp. 1576–1583. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2013.08.054.
`http://www.sciencedirect.com/science/article/pii/S0957417413006751`

Reynares, E.; Caliusco, M.L. y Galli, M.R. (2014b). «Evaluación empírica del mapeo de reglas de negocio para el desarrollo de ontologías». *Revista Ibérica de Sistemas y Tecnologías de Información (RISTI)*, **(14)**, pp. 83–99. ISSN 1646-9895. doi: http://dx.doi.org/10.17013/risti.14.83-99.
`http://www.aisti.eu/risti/risti14.pdf`

Reynares, E.; Caliusco, M.L. y Galli, M.R. (2014c). «SBVR to OWL 2 Mappings: An Automatable and Structural-Rooted Approach». *CLEI Electronic Journal (CLEIej)*, **17(3)**.

Robinson, Alan JA y Voronkov, Andrei (2001). *Handbook of automated reasoning*. volumen 2. ELSEVIER.

Rogers, Jeremy E. (2006). «Quality assurance of medical ontologies». *Methods of information in medicine*, **45(3)**, pp. 267–274.

Romli, R.; Sulaiman, S. y Zamli, Kamal Zuhairi (2010). «Automatic programming assessment and test data generation a review on its approaches». En: *Information Technology (ITSim), 2010 International Symposium in*, volumen 3, pp. 1186–1192. ISSN 2155-897. doi: 10.1109/ITSIM.2010.5561488.

Ruotsalo, T. (2010). *Methods and Applications for Ontology-Based Recommender Systems (PhD. Thesis)*. Tesis doctoral, Alto University School of Science and Technology, Finland.

Sampaio do Prado Leite, JC y Franco, Ana Paula M (1993). «A strategy for conceptual model acquisition». En: *Proceedings of IEEE International Symposium on Requirements Engineering, 1993*, pp. 243–246. IEEE.

Saripalle, Steven A. Demurjian Alberto De la Rosa AlgarÃn, Rishi Kanth y Blechner., Michael (2014). «A Software Modeling

Approach to Ontology Design via Extensions to ODM and OWL». *International Journal on Semantic Web and Information Systems (IJSWIS)*, **9(2)**, pp. 62–97. doi: 10.4018/jswis.2013040103.

Scheglmann, Stefan; Gröner, Gerd; Staab, Steffen y Lämmel, Ralf (2013). «Incompleteness-aware programming with RDF data». En: *Proceedings of the 2013 workshop on Data driven functional programming*, pp. 11–14. ACM.

Sharifloo, Amir Azim y Shamsfard, Mehrnoush (2008). «Using Agility in Ontology Construction». En: *Proceedings of the 2008 conference on Formal Ontologies Meet Industry*, pp. 109–119. IOS Press.

Shvaiko, P. y Euzenat, J. (2013). «Ontology Matching: State of the Art and Future Challenges». *IEEE Transactions on Knowledge and Data Engineering*, **25(1)**, pp. 158–176.

Sjøberg, Dag IK; Hannay, Jo Erskine; Hansen, Ove; Kampenes, Vigdis By; Karahasanovic, Amela; Liborg, N-K y Rekdal, Anette C (2005). «A survey of controlled experiments in software engineering». *Software Engineering, IEEE Transactions on*, **31(9)**, pp. 733–753.

Stevens, Robert y Lord, Phillip (2009). «Application of ontologies in bioinformatics». En: *Handbook on Ontologies*, pp. 735–756. Springer.

Suárez-Figueroa, MariCarmen; Gómez-Pérez, Asunción y Villazón-Terrazas, Boris (2009). «How to Write and Use the Ontology Requirements Specification Document». En: Robert Meersman; Tharam Dillon y Pilar Herrero (Eds.), *On the Move to Meaningful Internet Systems: OTM 2009*, volumen 5871 de *Lecture Notes in Computer Science*, pp. 966–982. Springer Berlin Heidelberg. ISBN 978-3-642-05150-0.

Svatek, Vojtech (2004). «Design patterns for semantic web ontologies: Motivation and discussion». En: *7th Conference on Business Information Systems*, Poznan.

Tichy, Walter F (2000). «Hints for reviewing empirical work in software engineering». *Empirical Software Engineering*, **5(4)**, pp. 309–312.

Uschold, M. (2008). «Ontology-driven information systems: Past, present and future». En: *Proceedings of the 2008 conference on Formal Ontology in Information Systems: Proceedings of the Fifth International Conference (FOIS 2008)*, pp. 3–18. IOS Press.

Vieira, T.A.S.C.; Casanova, M.A. y Ferrao, L.G. (2004). «An Ontology-Driven Architecture for Flexible Workflow Execution». En: *Proceedings of WebMedia and LA-WEB*, pp. 70–77. IEEE Computer Society.

Villazón-Terrazas, B. M. (2011). *A Method for Reusing and Re-engineering Non-ontological Resources for Building Ontologies*. Tesis doctoral, Universidad Politécnica de Madrid, Madrid. España.

Völker, Johanna; Vrandečić, Denny y Sure, York (2005). «Automatic Evaluation of Ontologies (AEON)». En: Yolanda Gil; Enrico Motta; V. Richard Benjamins y Mark A. Musen (Eds.), *The Semantic Web ISWC 2005*, volumen 3729 de *Lecture Notes in Computer Science*, pp. 716–731. Springer Berlin Heidelberg.

Vrandečić, Denny (2010). *Ontology Evaluation*. Tesis doctoral.

W3C (2004a). «World Wide Web Consortium. OWL Web Ontology Language. Guide».
`http://www.w3.org/TR/owl-guide/`

W3C (2004b). «World Wide Web Consortium. SWRL: A Semantic Web Rule Language Combining OWL and RuleML».
`http://www.w3.org/Submission/SWRL/`

W3C (2009a). «World Wide Web Consortium. OWL 2 Web Ontology Language. Document Overview».
`http://www.w3.org/TR/owl2-overview/`

W3C (2009b). «World Wide Web Consortium. OWL 2 Web Ontology Language. Structural Specification and Functional-Style Syntax».
`http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/`

W3C (2012a). «World Wide Web Consortium. OWL 2 Web Ontology Language Direct Semantics (Second Edition)».
http://www.w3.org/TR/owl2-direct-semantics/

W3C (2012b). «World Wide Web Consortium. OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition)».
http://www.w3.org/TR/owl2-rdf-based-semantics/

W3C (2014). «World Wide Web Consortium. RDF Schema. Version 1.1».
http://www.w3.org/TR/rdf-schema/

Wang, Taowei David (2006). «Gauging ontologies and schemas by numbers». En: *Proceedings of the Workshop EON Evaluation of Ontologies for the Web*, ACM Press. USA..

Wang, X. y Chan, C. (2001). «Ontology Modeling Using UML». Calgary, Canada.

Whittle, Jon; Hutchinson, John y Rouncefield, Mark (2013). *Model-Driven Development: A Practical Approach*. Chapman & Hall, 1st[a] edición. ISBN 1466501952, 9781466501959.

Wos, L.; Overbeck, R.; Lusk, E. y Boyle, J. (1984). *Automated reasoning: Introduction and applications*.

Zacharias, V. (2008). «Rules as simple way to model knowledge: Closing the gap between promise and reality». En: *Proceedings of the 10th International Conference on Enterprise Information Systems*, .