UNIVERSIDAD NACIONAL DEL SUR

Tesis de Doctorado en Ciencias de la Computación

Metodologías y Herramientas Visuales para Ingeniería Ontológica

Germán Alejandro Braun

Bahía Blanca Argentina

2018

Prefacio

Esta Tesis se presenta como parte de los requisitos para optar al grado académico de Doctor en Ciencias de la Computación, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el ámbito del Laboratorio de Investigación & Desarrollo en Ingeniería de Software y Sistemas de Información (LISSI), durante el período comprendido entre el 1 de junio de 2013 y el 31 de diciembre de 2018, bajo la dirección del Dr. Pablo Fillottrani, Profesor Titular del Departamento de Ciencias e Ingeniería de la Computación.

Germán Alejandro Braun
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, 15 de diciembre de 2018.

A Caro, por su amor incondicional.

Agradecimientos

Quiero comenzar agradeciendo a mi director Dr. Pablo Fillottrani y a la Dra. Laura Cecchi, quienes tuvieron la paciencia para guiarme, quienes me enseñaron que la investigación es divertida y quienes hicieron que ame mi trabajo. A ellos debo agradecer el empuje, la motivación y mi formación en investigación.

Estoy en deuda con todos mis compañeros docentes de nuestra Facultad de Informática, de la Universidad Nacional del Comahue, por ser parte de mi formación de grado y por fomentar el sueño de ser investigador. Gracias a mis compañeros del Departamento de Teoría de la Computación, por el alegre entorno de trabajo y por las discusiones enriquecedoras que tenemos en forma permanente.

Gracias a mis padres, a quienes debo mi educación. A mi hermano, Fede, una gran persona y profesional.

Gracias a todos mis amigos, de la vida, y a los que el estudio me hizo conocer.

Finalmente, a Caro, la persona más importante de mi vida.

Resumen

El objetivo principal de esta tesis es estudiar la retroalimentación entre los sistemas de representación visual de conocimiento y los formalismos lógicos, y definir la teoría subyacente a esta interacción, mediante la manipulación de ontologías gráficas basadas en Lógicas Descriptivas (DLs) y las características principales de los ambientes de ingeniería ontológica que la soporten.

Las tecnologías semánticas son cada vez más preponderantes en la integración de datos e interoperabilidad de sistemas de información y, en este contexto, las ontologías son centrales para la definición de vocabularios compartidos y modelos conceptuales. De esta manera, proveer ambientes para el desarrollo de ontologías de calidad es esencial, potenciando la integración del conocimiento de los expertos de dominio con la semántica formal de los lenguajes de ontologías.

En esta Tesis se presentan la formalización de los sistemas de manipulación de ontologías gráficas, a los cuales notamos como *GOMS*, y de un proceso de visualización de conocimiento basado en ontologías, que articula esta teoría junto con aspectos de visualización en el contexto de herramientas gráficas. A partir de los resultados obtenidos, se diseñó y documentó una arquitectura de referencia web y se implementó una herramienta concreta, llamada *crowd*, para tareas de ingeniería ontológica, por medio de representaciones gráficas de dominios y sus reconstrucciones en DL.

Esta infraestructura ha sido concebida como un sistema visual integrando fuertemente los modelos gráficos con sus representaciones lógicas, interfaces con múltiples razonadores lógicos para validarlos y cumplimiento con estándares relevantes de la W3C. Asimismo, se incorporó la gestión de espacios de nombres para obtener modelos ontológicos listos para documentar y publicar.

Evaluaciones basadas en experiencias de usuarios y en la formalización de un sistema para visualización de contenido semántico, son también presentadas y han sido ejecutadas sobre la implementación de *crowd*, actualmente en linea en `http://crowd.fi.uncoma.edu.ar`.

Abstract

The aim of this thesis is to study how both knowledge representation visual systems and logic-based formalisms feed each other, theorise about this interaction through manipulating graphical ontologies based on Description Logics (DLs), and thus defining the main requirements of ontology engineering environments to support them.

Semantics technologies are increasingly important in data integration and information system interoperability. In this sense, ontologies are key for shared vocabularies and conceptual models. Thus, providing environments for high quality ontologies becomes an essential issue, empowering the integration of domain experts' knowledge with the semantics of ontology languages.

In this thesis, we formalise a system for manipulating graphical ontologies, named as *GOMS*, and an ontology-based knowledge visualisation process, which orchestate the theoretical and visualisation aspects in the context of visual tools. From these results, we have designed and documented a reference web architecture. Moreover, we have implemented a concrete tool, named *crowd*, for ontology engineering tasks based on visual representations of domains and their logic-based reconstructions.

This infrastructure has been conceived as a visual system intragrating closely visual models and their logical representations, interfacing with diverse reasoning tools to validate them and compliancing to all relevant W3C recommendations. The tool also support namespaces definition in order to get ontologies ready to be documented and published.

Evaluations based on user experiences and the formalisation as a system for visualising semantic content are also presented, which have been run on the current implementation of *crowd*, hosted at `http://crowd.fi.uncoma.edu.ar`.

Índice general

Capítulo 1

Introducción

Contents

En este capítulo, se introducen el contexto y la motivación que guiaron el desarrollo de esta investigación y que dieron como resultado un ambiente web visual para ingeniería ontológica y el estudio de la formalización de ontologías gráficas, por medio de un sistema para manipularlas que gestiona los aspectos visuales de estos modelos y su representación semántica basada en lógica. Se presentan también los requerimientos que fueron identificados para la herramienta, los cuales serán abordados en el transcurso de esta investigación. Asimismo, se sintetizan las principales contribuciones que se han obtenido y que han sido publicadas en congresos nacionales e internacionales. Finalmente, se resume la organización de esta Tesis.

1.1. Motivación

Los sistemas basados en redes semánticas[1] [Qui67, BL85] ofrecen un enfoque centrado en humanos para representar conocimiento en términos de conjuntos de individuos y sus relaciones. Estos enfoques están basados en el uso de interfaces gráficas dónde el razonamiento es definido en base a la manipulación de las estructuras de las redes. En este sentido, si bien son consideradas más efectivas que los sistemas lógicos, la ausencia de una semántica bien definida redunda en caracterizaciones ambiguas de dominios. Como consecuencia, surgió la necesidad de definir formalismos lógicos que den semántica a este tipo de sistemas pero que, además, garanticen razonamiento completo para validar su consistencia. Este contexto dio lugar a una familia de lógicas, llamadas Lógicas Descriptivas (DLs), [BHLS17, BCM$^+$03a] un conjunto de formalismos con diferente poder expresivo y propiedades computacionales.

Una ontología [SMJ02] define un vocabulario compartido para un dominio de interés, modelado, a diferencia de las bases de datos, como un conjunto de axiomas, los cuales deben verificarse en la situación descripta. Las DLs son lenguajes de representación ampliamente usados en el desarrollo de ontologías y la base lógica que subyace a los lenguajes de ontologías web, tales como OWL [HKP$^+$09]. Las DLs están equipadas con una especificación precisa y propiedades computacionales bien definidas. Esto permite que humanos y sistemas de computaciones puedan intercambiar ontologías sin ambigüedades, haciendo posible el uso de deducción lógica para inferir propiedades adicionales a partir de la información explícitamente definida en una ontología.

Sin embargo, la necesidad de dar soporte al diseño, edición y mantenimiento de sistemas basados en DLs, hizo resurgir nuevamente la motivación principal de las redes semánticas, es decir, el uso de interfaces gráficas amigables, junto con servicios de razonamiento, para que modeladores puedan realizar tareas de ingeniería de esto sistemas. Estos ambientes deben posibilitar la integración de las intenciones de los expertos y la semántica formal

[1] *Semantic Network-based systems*

de los lenguajes de ontologías, además de ofrecer un manejo comprensivo del conocimiento implícito.

Esto motivó la formalización teórica y desarrollo de un ambiente web y gráfico, a partir de esta formalización, para tareas de ingeniería ontológica [Miz04] que articulen ambas motivaciones: la representación de conocimiento basado en interfaces gráficas orientadas a humanos y el uso de formalismos lógicos para una definición precisa de la semántica de modelos gráficos. En este contexto, los requerimientos principales para este tipo de ambientes son el soporte de lenguajes gráficos como fuente primaria de abstracción y la integración son sistemas basados en lógica para explorar, componer y validar los modelos, explicitando a los usuarios su semántica subyacente [VBJS14].

A través de una revisión exhaustiva de la bibliografía, un gran ecosistema de herramientas ha sido relevado, los cuales presentan diversos requerimientos específicos y tareas de ingeniería ontológica [ABAG17, DLSP18]. Sin embargo, ninguno de ellos presenta la integración de soporte web e interactivo en un mismo ambiente. Entre los más referenciados y ampliamente aceptados, se destacan Protégé [KFNM04] y su versión web [TNNM13], aunque la falta de soporte visual más expresivo es una de sus mayores desventajas, limitando su uso a usuarios con cierto conocimiento en DLs. Herramientas más recientes tales como VOWL [LNB14], eddy [CLSS14] and OWLGrEd [COLS12], presentan novedosas interfaces visuales basados en grafos y en UML, pero la integración con herramientas lógicas para su validación y verificación es limitada o ausente.

Como resultado de nuestra investigación, hemos podido plasmar nuestros aportes en un sistema formal para manipulación de ontologías gráficas, denominado *GOMS*, y una herramienta *open-source* web y gráfica para tareas de ingeniería ontológica e integrada a razonadores lógicos, llamada *crowd*, la cual se encuentra publicada en `http://crowd.fi.uncoma.edu.ar/`.

1.2. Objetivos

Durante varias décadas, un esfuerzo considerable ha sido invertido en ampliar el horizonte teórico de las DLs y los lenguajes de ontologías, basados sobre estás lógicas, sin embargo, relegando a un segundo plano (en algún grado) el desarrollo de ambientes de ingeniería ontológica para simplificar la formalización y la validación de las propiedades de los dominios de aplicación.

Este trabajo de tesis busca investigar, formalizar y caracterizar la relación identificada entre los sistemas visuales y los sistemas lógicos para representación de conocimiento. Asimismo, proveer las bases para el desarrollo e implementación de herramientas web para tareas de ingeniería ontológica basados en la teorización de esta relación.

Nuestra principal pregunta de investigación **(PI)** es la siguiente:

¿Cómo la formalización de ontologías y los lenguajes visuales pueden ser integrados en una herramienta para ingeniería ontológica?

Para esto, es necesario estudiar la literatura subyacente a ambas dimensiones, cómo es su relación, cómo estás influyen en la ingeniería ontológica y cómo deben diseñarse las herramientas que las soporten. Para esto, se trabajará sobre un conjunto de preguntas de investigación que nos conducirán hacía nuestra pregunta principal.

En primer lugar, examinaremos las tareas involucradas en la ingeniería ontológica y las metodologías que las soportan. Asimismo, profundizaremos en conceptos teóricos fundamentales: la familia de lógicas descriptivas, sus características, los servicios de razonamiento y los lenguajes para especificar ontologías. A partir de este análisis abordaremos las primeras dos PI:

- **(PI1)** ¿Cómo integrar efectivamente la formalización de ontologías en las herramientas de ingeniería ontológica?

- **(PI2)** ¿Cómo puede aprovecharse el razonamiento automático en los procesos de ingeniería ontológica?

Tal como presentamos en la motivación de esta Tesis, los sistemas visuales son consideramos más efectivos que los sistemas lógicos, aunque los primeros carecen de una semántica bien definida y, por lo tanto, su interpretación es ambigua. Sin embargo, el uso de interfaces gráficas amigables es importante en tareas de modelado que ayuden en la comprensión de los modelos y en su comunicación, decrementando la carga cognitiva de los modeladores y usuarios finales. Para esto, exploraremos los lenguajes visuales para ontologías y los procesos visualización de datos, en general. Así una nueva PI es la siguiente:

- **(PI3)** ¿Cómo influyen los lenguajes visuales de ontologías en herramientas de la ingeniería ontológica?

Por último, poner en práctica estos conceptos teóricos relevados previamente, requiere de ambientes software avanzamos que orquesten librerías para manipulación gráfica junto con procesamiento y acceso a los modelos generados por parte de humanos y computadoras. La identificación de requerimientos necesarios para tales ambientes forma parte del abordaje de la última PI.

- **(PI4)** ¿Cuáles son los requerimientos principales que una herramienta visual, para tareas de ingeniería ontológica, debe satisfacer?

Las cuatro PI en conjunto nos proveerán un marco conceptual que será utilizado para responder a nuestro principal PI. Además, nos permitirán analizar en profundidad la relación existente entre los sistemas visuales y lógicos y cómo ambos colaboran en tareas automáticas de ingeniería ontológica.

El campo de las Lógicas Descriptivas tuvo grandes e importantes aportes teóricos y prácticos durante las últimas dos décadas, relegando así los

avances respecto a los sistemas basados en interfaces visuales, considerando que la complejidad alcanzada por las DLs en cuanto a poder expresivo dificulta su uso como lenguaje de modelado, para el que también se requieren visualizaciones más complejas y expresivas.

Finalmente, las herramientas con soporte de razonamiento automático proveen funcionalidades vitales para tareas de ingeniería ontológica. Algunas de ellas, tales como Protégé, presentan requerimientos mínimos para soporte de estas tareas, aunque también diversas herramientas adicionales deben ser consideradas, aumentando así la complejidad del modelado [VBJS14, Hor11].

1.3. Metodología

Durante el desarrollo de la investigación propuesta, se relevaron y analizaron lenguajes de ontologías, sus propiedades computacionales y cómo estos pueden ser integrados en un proceso de visualización para herramienta gráficas. Como resultado, se obtuvo la formalización del concepto de ontología gráfica a partir de la formalización de la interacción entre los lenguajes de ontologías y los de modelado conceptual gráfico.

Se relevaron arquitecturas y herramientas existentes y se cotejaron tomando en cuenta los aspectos relevados previamente y, en particular, el soporte gráficos subyacente de cada una. En este sentido, se formalizó una arquitectura de referencia para gestionar ontologías gráficas y, basado en esta arquitectura, se implementó una herramienta llamada *crowd*, en la cual se utilizaron tecnologías libres: los lenguajes de programación PHP, JavaScript y otras off-the-shelf como razonadores y motores de consultas, como MariaDB. Para una correcta codificación, se hizo uso de técnicas de desarrollo guiado por testing (TDD, test-driven development), que además permite incrementar la complejidad del software de una manera orgánica y estable.

El análisis de los resultados incluyó la comparación con metodologías y herramientas de la literatura y las condiciones de aplicación. Se utilizaron

indicadores y usabilidad y encuestas a usuarios alumnos y docentes de la carrera de Ciencias de la Computación.

La documentación del trabajo se realizó de forma continua, y se recurrió a permanente revisión del estado del arte y actualización bibliográfica. Además, todo el proceso fue documentado mediante el uso de bitácoras y reportes internos de avance; como así también con publicaciones en congresos (nacionales e internacionales) y revistas indexadas de áreas afines.

1.4. Organización de la Tesis

La tesis incluye los contenidos necesarios para que su lectura sea autocontenida, asumiéndose conocimientos a nivel de grado de lógica clásica, representación de conocimiento y ontologías, nociones de complejidad computacional y de los lenguajes de modelado de datos estándar (UML, EER, ORM 2), y arquitecturas de software. A continuación se describe la estructura de esta Tesis, organizada en siete capítulos y un apéndice.

Capítulo 1: Introducción. Este capítulo describe las motivaciones y las contribuciones de la Tesis.

Capítulo 2: Lógicas Descriptivas y Ontologías. Se analiza el marco teórico subyacente al presente trabajo: Lógicas Descriptivas, lenguajes de ontologías, lenguajes gráficos de modelado conceptual e ingeniería ontológica.

Capítulo 3: Gestión de Ontologías Gráficas. Se introduce un framework teórico para la gestión de ontologías gráficas, considerando aspectos visuales, formalismos lógicos y su integración.

Capítulo 4: Arquitectura de Referencia para Ambientes Web de Ingeniería Ontológica. Se detalla el diseño arquitectónico para los sistemas gráficos. Se describen las diferentes vistas de la arquitectura y sus propiedades.

Capítulo 5: *crowd v1.0* y Evaluación. Se introduce la herramienta *crowd*, se presenta su descripción funcional, la evaluación basada en usuarios y la formalización de la herramienta como un modelo para gestionar contenido semántico.

Capítulo 6: Comparación con otras Herramientas. Se analiza el soporte de herramientas existentes para el desarrollo, mantenimiento y uso de ontologías gráficas. La intención es explorar otros enfoques y analizar su cubrimiento con respecto a los requerimiento definidos para *crowd*.

Capítulo 7: Conclusiones. Se detallan los resultados y conclusiones de esta Tesis y se delinean líneas de investigación aún pendientes de estudio.

Apéndice: Transformaciones Gráfico-Lógicas en *crowd v1.0*. Se detallas los mapeos implementados en la primer versión de la herramienta para reconstruir lógicamente una ontología gráfica y para extraer un modelo visual desde una ontología expresada en un lenguaje basado en lógica.

Bibliografía: Recopilación de las referencias bibliográficas de todos los capítulos y del apéndice, ordenadas alfabéticamente por autor.

1.5. Contribuciones

De este estudio, análisis y motivación, confluye la contribución central de nuestra investigación: *la integración de lenguajes visuales con razonamiento basado en lógica y su utilidad en cuanto a tareas automáticas de ingeniería ontológica, y la validación de modelos. La integración abarca no sólo a la invocación de servicios de razonamiento, sino también la generación de un nuevo modelo plasmando sus resultados y su posterior visualización.*

Hasta donde conocemos, la definición formal de ambientes en los cuales

se orquesten lenguajes visuales, sus reconstrucciones basadas en lógica y la integración de ambos para visualización de ontologías, junto con los respectivos servicios de razonamiento, es novedoso en el área de la Representación de Conocimiento y Razonamiento (KR). En este sentido, podemos distinguir los siguientes aportes:

1. En primer lugar, la formalización de los sistemas de manipulación de modelos visuales, dando un marco teórico para la definición del concepto de ontologías gráficas y un proceso de visualización de conocimiento, así como también para la caracterización de ambientes de ingeniería ontológica, especificando claramente su poder expresivo lógico y gráfico.

   Esta primer contribución tiene como resultados un *Sistema para la Manipulación de Ontologías Gráficas* (en inglés, *GOMS*), basado sobre la especificación de un modelo heterogéneo, independiente de un lenguaje visual y una lógica particular [BCF15]. La intención detrás de este enfoque es formalizar la reconstrucción lógica de un lenguaje gráfico, sus servicios de razonamiento y su posterior tratamiento de los resultados para ser visualizados nuevamente por modeladores. Para su desarrollo, se analizaron los diferentes perfiles expresivos de las DLs, sus propiedades computacionales y sus aplicaciones, principalmente, su rol central en la Web Semántica [BLHL01, Hor08] y en la definición de los lenguajes de ontologías, tales como OWL.

   En esta misma dirección, un *GOMS* es el concepto central de nuestra definición de un *Proceso de Visualización de Conocimiento basado en Ontologías* [BGCF16]. Dicho proceso articula los aspectos referidos a la integración gráfica y lógica, junto con requerimientos visuales complementarios para disminuir la carga cognitiva de los usuarios [War04] y de calidad [KS02].

2. La segunda contribución relevante es la definición, el diseño y la documentación de una *arquitectura de referencia* web para el Proceso de Visualización [BEF18], a través de la caracterización de sus módulos,

su interacción y su instalación. Para esto, se documentaron vistas complementarias de módulos, componentes y conectores, e implantación, junto con diagramas de este diseño [GBI+10, Gom11].

Para lograr esta arquitectura, se relevaron y analizaron herramientas y sistemas para el desarrollo de ontologías presentes en la literatura [ABAG17, DLSP18], considerando, principalmente, los aspectos referidos en nuestra hipótesis. Durante este proceso, se emplearon las definiciones de nuestro *GOMS* y el proceso de visualización.

Las características destacadas de nuestro enfoque son la flexibilidad y escalabilidad lograda por una arquitectura web, su capacidad para interactuar con razonadores lógicos a través de protocolos estándar, tales como OWLlink [LLR+08], el cumplimiento de las recomendación de la W3C[2], manipulando ontologías OWL 2, y el soporte para la definición de espacios de nombres para la documentación y publicación de las ontologías.

3. Como última contribución preponderante, se desarrolló, implementó, implantó y evaluó una herramienta web a partir de los aportes descriptos anteriormente. Esta herramienta, a la cual nombramos *crowd*, fue definida como una instancia concreta de un *GOMS*, a partir de la identificación de requerimientos funcionales, que serán descriptos en este documento, y que se encuentra actualmente en linea en el servidor `http://crowd.fi.uncoma.edu.ar`.

   Dicha herramienta fue evaluado mediante diversos enfoques y experimentos con usuarios reales: Factores de Usabilidad[3] [Lau05] y Escala de Usabilidad de Sistemas (en inglés, SUS)[4] [LS09], y comparación con otros enfoques similares. Para esto último, formalizamos a *crowd* como una instancia de un modelo *WYSIWYM* (*What You See Is What You Mean*) [KA15] que permite concluir sobre la relación entre los modelos semánticos, como las ontologías, y los respectivos elementos gráficos de

[2] `https://www.w3.org/`
[3] *Factors of Usability*
[4] *System Usability Scale (SUS)*

una herramienta para tareas de visualización, exploración y desarrollo de estos modelos.

crowd está basado sobre la noción de deducción completa de servicios de razonamiento lógicos relativos a la sintaxis gráfica de lenguajes visuales. Los usuarios de la herramienta visualizarán las ontologías gráficas con todas las deducciones expresadas en el mismo lenguaje gráfico. Esto incluye la validación de la consistencia de clases y relaciones, clases implicadas por el diagrama o restricciones de cardinalidad más precisas. Aserciones no gráficas también pueden expresarse mediante axiomas textuales. *crowd* se enfoca solamente en el modelado gráfico de esquemas terminológicos de ontologías. Asimismo, a partir de la formalización previa y como funcionalidad complementaria, es posible utilizar la herramienta para validar modelos conceptuales en procesos relacionados a la ingeniería de software.

Por último, durante el transcurso de esta investigación se presentaron enfoques y metodologías complementarias para la evolución y mantenimiento de modelos ontológicos en el contexto de una herramientas visual, que no serán detalladas en esta Tesis. Estos aportes incluyen la especificación de algoritmos y reglas para alcanzar patrones de diseño ontológico [BC15, BCF15] y una versión gráfica [GBCF18] para consultas SPARQL-DL [SP07] sobre ontologías. En este mismo sentido, también se exploraron las DLs en el contexto del análisis de variabilidad para el desarrollo de software basado en reuso [BPC$^+$17, OBCF18]. Estas contribuciones teóricas fueron presentadas en congresos nacionales e internacionales, y aún se encuentran en etapas de implementación y evaluación.

Capítulo 2

Lógicas Descriptivas y Ontologías

Contents

En Ciencias de la Computación, una *ontología* es la descripción del conocimiento sobre un dominio de interés. Sin embargo, a diferencia de un modelo conceptual de datos, una ontología provee una representación independiente de una determinada aplicación. Es posible representar el conocimiento de una aplicación de dominio, de forma estructurada y formalmente bien comprendida, expresándola por medio de una Lógica Descriptiva (DL) [BHS08]. Esto dota a los modelos resultantes de capacidades de razonamiento para validarlos y consultar sobre sus propiedades.

Las DLs han sido adoptadas como el principal paradigma para la des-

cripción ontológica, que luego redundó en la estandarización del lenguaje de ontologías web OWL [GHM+08, Krö12], por parte del *World Wide Web Consortium (W3C)*. Además, dicha adopción posibilitó la construcción de herramientas para inferencias automáticas [MLH+15], definiendo así la base teórica para los sistemas de información. Las DLs son una evolución de las redes semánticas [Qui67] y los sistemas basados en frames [Min74] que, aunque carentes de una semántica precisa, su ventaja era la facilidad de comprensión [BCM+03a].

En este contexto, éstas lógicas surgen para dotar a estas representaciones de una semántica formal. Además, las DLs son una familia de lenguajes de representación de conocimiento que pueden ser consideradas como un subconjunto decidible de la lógica de primer orden (FOL). Ellas pueden ser usadas para describir el dominio de una aplicación en términos de clases y relaciones y admitiendo razonamiento decidible. En la misma dirección, surge OWL [OWL09, AH11] como un lenguaje formal de la Web Semántica [BLHL01], para expresar ontologías. Basado en la semántica de DL, OWL ha sido diseñado para aplicaciones que requieren procesar información, en lugar de sólo presentarla a los humanos, además de incrementar la interoperabilidad web.

El propósito del presente capítulo es hacer un breve repaso de las lógicas descriptivas básicas, tomando como referencia las lógicas $\mathcal{ALC}$ y $\mathcal{ALCQI}$, y el lenguaje OWL, a través de sus características principales y sus sublenguajes subyacentes. Primeramente, detallamos la sintaxis y semántica de las lógicas $\mathcal{ALC}$ y $\mathcal{ALCQI}$. También, describimos algunas lógicas más expresivas y resumimos sus propiedades computacionales. A continuación, analizamos los lenguajes de ontologías OWL 1 y OWL 2 y realizamos una breve comparación de ambos. Finalmente, plasmamos los conceptos principales de la ingeniería de ontologías, metodologías y herramientas para soportar el diseño ontológico.

Este capítulo y los temas vertidos fundamentan y concluyen la necesidad de dar soporte a las tareas de ingeniería ontológica y de compatibilizar

nuestro enfoque con la Web Semántica, gestionando de espacios de nombres, documentación y publicación de las ontologías, como detallan los requerimientos siguientes:

- ***(R1)*** *Soporte para Ingeniería Ontológica*: Los modeladores requieren ambientes para poder editar, mantener, evaluar, usar y documentar ontologías por medio de diversas metodologías y lenguajes [Miz04, Kee18, FLGPJ97].
- ***(R4)*** *Compatibilidad con la Web Semántica*: Las ontologías son los principales componentes de la Web Semántica, de manera que ellas deben ser modeladas utilizando esquemas globales de nombres (espacios de nombres) y documentadas para compartir con otros interesados para lograr el consenso necesario para este tipo de modelos [BLHL01].

2.1. Lógicas Descriptivas

Investigaciones en el campo de la representación de conocimiento y el razonamiento, usualmente se enfocan en métodos que proveen una descripción de alto nivel de un mundo, que pueda ser usado de manera efectiva, para construir aplicaciones inteligentes. En este caso, con "inteligente" nos referimos a la posibilidad de encontrar consecuencias implícitas de una representación de conocimiento [BCM$^+$03a]. A pesar de que las lógicas de primer orden (FOL) ofrecen una maquinaria muy poderosa y general, no es preciso utilizar toda su expresividad para la representación de conocimientos en dominios reales sino fragmentos de ella, como queda demostrado por la utilización de redes semánticas y frames descripto en [BL85]. A partir de esto, se deduce que son posibles los razonamientos basados en diferentes fragmentos de FOL que conllevan a problemas de diversas complejidades.

En consecuencia, surgen las Lógicas Descriptivas (DLs), como una familia de lenguajes para la representación de conocimiento de una forma estructural y formalmente bien comprendida [BHS08]. Su nombre proviene del hecho de

que las nociones más importantes del dominio son descriptos como *descripciones* de conceptos, en otras palabras, de expresiones que se construyen por medio de conceptos y roles atómicos usando constructores provistos por la DL particular.

Semánticamente, un *concepto* es interpretado como un conjunto de individuos y los *roles* como conjuntos de pares de individuos. En el aspecto terminológico, una base de conocimiento, llamada TBox, describe nociones relevantes de un dominio de aplicación estableciendo propiedades sobre los conceptos y roles, y relaciones entre ellos. La parte de aserciones de la base de conocimiento, denominada ABox, es usada para describir una situación concreta detallando las propiedades para individuos.

Por lo tanto, en este contexto, podemos considerar a una ontología como equivalente a una base de conocimiento en DL, consistente de un TBox y un ABox [Hor08, HPSvH03a].

2.1.1. $\mathcal{ALC}$ y $\mathcal{ALCQI}$

$\mathcal{ALC}$ (Attributive concept Language with Complements) significa "lenguaje de conceptos con atributos y complementos". Se introdujo inicialmente en [SSS91], donde se investigaron la incorporación de la unión ($\sqcup$), intersección ($\sqcap$) y complemento ($\neg$) a un lenguaje de conceptos con atributos y su impacto en la complejidad computacional.

A continuación, se brindarán definiciones formales de la sintaxis y semántica de los constructores según se encuentran en [BCM$^+$03a, BHLS17].

Definición 2.1 (Sintaxis $\mathcal{ALC}$) : Sea N_C un conjunto de nombres de conceptos y N_R un conjunto de nombres roles. El conjunto de descripciones de $\mathcal{ALC}$-conceptos es el conjunto más pequeño de forma que:

1. $\top$, $\bot$ y cada concepto $A \in N_C$ es un $\mathcal{ALC}$-concepto atómico.

2. Si C y D son $\mathcal{ALC}$-conceptos y $r \in N_R$, entonces $C \sqcap D, C \sqcup D, \neg C, \forall r.C$ y $\exists r.C$ son $\mathcal{ALC}$-conceptos.

■

Definición 2.2 (Semántica $\mathcal{ALC}$) : Una interpretación $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consiste de un conjunto no vacío $\Delta^{\mathcal{I}}$ denominado el dominio de $\mathcal{I}$ y la función $\cdot^{\mathcal{I}}$ que asocia cada $\mathcal{ALC}$-concepto con un subconjunto de $\Delta^{\mathcal{I}}$, y cada nombre de rol a un subconjunto de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, para todos los $\mathcal{ALC}$-conceptos C, D y para todo nombre de rol r:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} & \bot^{\mathcal{I}} &= \emptyset, \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} & (C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} & \neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\ (\exists r.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \ \exists\ y \in \Delta^{\mathcal{I}} & (x,y) \in r^{\mathcal{I}},\ y \in C^{\mathcal{I}}\} \\ (\forall r.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \ \forall\ y \in \Delta^{\mathcal{I}}, \text{ si } (x,y) \in r^{\mathcal{I}}, \rightarrow y \in C^{\mathcal{I}}\} \end{aligned}$$

Decimos que $C^{\mathcal{I}}$ ($r^{\mathcal{I}}$) es una extensión del concepto C (rol r) en la interpretación $\mathcal{I}$. Si $x \in C^{\mathcal{I}}$, entonces decimos que x es una instancia de C en $\mathcal{I}$.

■

El dominio de la interpretación puede ser elegido de forma arbitraria y puede ser infinito. Las características de los dominios, en relación a finito/infinito y la consideración del mundo abierto, son distintivas de las lógicas descriptivas, con respecto a los lenguajes de modelado desarrollados en el estudio de bases de datos. Los conceptos atómicos son interpretados como subconjuntos del dominio de interpretación, mientras que la semántica para los demás constructores son especificados definiendo un conjunto de individuos denotado por cada constructor.

Definición 2.3 (GCI y Sintaxis y Semántica TBox) : La inclusión ge-

neral de conceptos (GCI por sus siglas en inglés "General Concept Inclusion") es de la forma $C \sqsubseteq D$, donde C y D son $\mathcal{ALC}$-conceptos. Un conjunto finito de GCIs es llamado un TBox. Una interpretación $\mathcal{I}$ es un modelo de un GCI $C \sqsubseteq D$ si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ también lo es de un TBox T si éste lo es de cada GCI en T.

■

El uso de $C \equiv D$ es una abreviación para el par simétrico de GCIs $C \sqsubseteq D$ y $D \sqsubseteq C$.

Por otra parte, los axiomas para indicar que un individuo es una instancia de un concepto dado y para indicar que un par de individuos es una instancia de un rol serán incluidos en lo que se denomina ABox.

Definición 2.4 (Axioma de Aserción) : Un axioma de aserción es de la forma $x : C$ o $(x, y) : r$ donde C es un $\mathcal{ALC}$-concepto, r es un $\mathcal{ALC}$-rol y x e y son nombres de individuos. Un conjunto finito de axiomas de aserción es llamado un ABox. Una interpretación $\mathcal{I}$ es un modelo de un axioma de aserción $x : C$ si $x^{\mathcal{I}} \in C^{\mathcal{I}}$ e $\mathcal{I}$ es un modelo de un axioma de aserción $(x, y) : r$ si $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$; $\mathcal{I}$ es un modelo de un ABox $\mathcal{A}$ si es un modelo para todos los axiomas en $\mathcal{A}$.

■

Existen varias anotaciones para describir un ABox en la literatura, en este trabajo utilizaremos en forma indistinta $C(x)$ o $x : C$ para conceptos, y $r(x, y)$ o $\langle x, y \rangle : r$ para roles.

Definición 2.5 (Base de Conocimiento) : Una base de conocimiento (KB) es un par (T, A), donde T es un TBox y A es un ABox. Una interpretación $\mathcal{I}$ es un modelo de la KB $K = (T, A)$ si $\mathcal{I}$ es un modelo para T e $\mathcal{I}$ es un modelo para A.

■

Definición 2.6 (Consecuencia Lógica) : Un axioma α es una consecuencia de una KB, escrito como KB $\models \alpha$ si todo modelo de la KB es también un modelo de α. Por lo tanto, para todo $\mathcal{I}$, donde $\mathcal{I}$ satisface a KB, también se cumple $\mathcal{I} \models \alpha$.

■

Ejemplo 2.1 Consideremos los siguientes $\mathcal{ALC}$-conceptos que describen una KB (un TBox y un ABox).

TBox:
$T = \{\, Padre \equiv Humano \sqcap Hombre \sqcap \exists tieneHijos$
$PadreAlegre \sqsubseteq Padre \sqcap \forall tieneHijos.(Doctor \sqcup Abogado \sqcup PersonaAlegre)$
$AlegreAnc \sqsubseteq \forall descendiente.PadreAlegre$
$Maestro \sqsubseteq \neg Doctor \sqcap \neg Abogado \,\}$

ABox:
$A = \{\, Maestro(mary), tieneHijos(john, mary), AlegreAnc(john) \,\}$

En este ejemplo se presenta una ontología expresada en $\mathcal{ALC}$-conceptos. En la primer línea se expresa que el concepto "padre" coincide con el de los humanos y que son de género masculino, además que deben tener hijos. Luego, un "padre alegre" es un subconjunto de los padres en los que todos sus hijos son doctores, abogados o son personas alegres.

■

La lógica descriptiva $\mathcal{ALCQI}$ aumenta la expresividad de la lógica $\mathcal{ALC}$ ampliando los constructores que define utilizando los roles inversos, denotado como $\mathcal{I}$, y con restricción numérica cualificada, notada como $\mathcal{Q}$. En el primer caso, en esta lógica más expresiva, ahora es posible modelar la inversa de una relación. Por ejemplo, la definición $\exists hijo^{-}.Doctor$ establece que alguien tiene un padre doctor, mediante el uso de la inversa del rol "hijo". Asimismo, las

restricciones numéricas cualificadas son una forma de restricciones numéricas, permitiendo la definición de cardinalidades sobre roles, en particular, sobre roles atómicos y su inversa. Usando esta nueva primitiva podemos modelar relaciones como $BuenPadre \sqsubseteq Padre \sqcap (\geq 2\ tieneHijo.Hijo)$, dónde expresamos que un "buen padre" es un padre con al menos 2 hijos.

De forma análoga a las definiciones de sintaxis y semántica $\mathcal{ALC}$, podemos describir la extensión $\mathcal{ALCQI}$ incorporando las definiciones de $\mathcal{I}$ y $\mathcal{Q}$.

Definición 2.7 (Sintaxis $\mathcal{ALCQI}$) : Sea N_C un conjunto de nombres de conceptos y N_R un conjunto de nombre de roles. El conjunto de descripciones de $\mathcal{ALCQI}$-conceptos es el conjunto más pequeño de forma que:

1. $\top$, $\bot$ y cada concepto $A \in N_C$ es un $\mathcal{ALCQI}$-concepto atómicos.
2. Si C y D son $\mathcal{ALCQI}$-conceptos y $r \in N_R$, entonces $C \sqcap D, C \sqcup D, \neg C, \forall r.C$ y $\exists r.C$ son $\mathcal{ALCQI}$-conceptos.
3. Si C es un $\mathcal{ALCQI}$-concepto, $r \in N_R$ y $n \in \mathbb{Z}^+$, entonces r^-, $(\geq n\ r.C)$, $(\leq n\ r.C)$ y $(= n\ r.C)$ también son $\mathcal{ALCQI}$-conceptos.

■

Definición 2.8 (Semántica $\mathcal{ALCQI}$) : Una interpretación $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consiste de un conjunto no vacío $\Delta^{\mathcal{I}}$ denominado el dominio de $\mathcal{I}$ y la función $\cdot^{\mathcal{I}}$ que asocia cada $\mathcal{ALCQI}$-concepto con un subconjunto de $\Delta^{\mathcal{I}}$, y cada nombre de rol a un subconjunto de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ de forma que, para todos los $\mathcal{ALCQI}$-conceptos C, D y para todo nombre de rol r:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset, \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \qquad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} |\ \text{existe algún } b \in \Delta^{\mathcal{I}} \quad (a,b) \in r^{\mathcal{I}} \text{ e } b \in C^{\mathcal{I}}\} \\
(\forall r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} |\ \text{para todo } b \in \Delta^{\mathcal{I}},\ \text{si } (a,b) \in r^{\mathcal{I}}, \text{ entonces } b \in C^{\mathcal{I}}\} \\
(r^-)^{\mathcal{I}} &= \{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} | (a,b) \in r^{\mathcal{I}}\} \\
(\geq n\ r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \left|\{b \in \Delta^{\mathcal{I}} | (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\right| \geq n\} \\
(\leq n\ r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \left|\{b \in \Delta^{\mathcal{I}} | (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\right| \leq n\} \\
(= n\ r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \left|\{b \in \Delta^{\mathcal{I}} | (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\right| = n\}
\end{aligned}
$$

Decimos que $C^{\mathcal{I}}$ ($r^{\mathcal{I}}$) es una extensión del concepto C (rol r) en la interpretación $\mathcal{I}$. Si $a \in C^{\mathcal{I}}$, entonces decimos que a es una instancia de C en $\mathcal{I}$.

■

Las definiciones subsecuentes de la sección 2.1.1, correspondiente a la inclusión general de conceptos, al axioma de aserción, a la base de conocimiento y consecuencias lógicas en $\mathcal{ALC}$ (definición 2.3, 2.4, 2.5 y 2.6 respectivamente), son análogas para $\mathcal{ALCQI}$.

Ejemplo 2.2 Considere un dominio en el cual Gestores de Proyectos tiene a su cargo Empleados que, a su vez, participan en Proyectos. En un proyecto deben participar como mínimo 3 empleados. Un proyecto debe ser gestionado por al menos dos gestores. Una representación $\mathcal{ALCQI}$ de este dominio es la siguiente:

TBox:

(1) $Manager \sqsubseteq Empleado$
(2) $AreaManager \sqsubseteq Manager$
(3) $TopManager \sqsubseteq Manager$
(4) $Manager \sqsubseteq AreaManager \sqcup TopManager$
(5) $AreaManager \sqcap TopManager \sqsubseteq \bot$
(6) $\exists trabajaPara \sqsubseteq Empleado$
(8) $\exists trabajaPara^{-} \sqsubseteq Proyecto$
(9) $Proyecto \sqsubseteq (\geq 3\ trabajaPara^{-})$
(10) $\exists gestiona \sqsubseteq TopManager$
(11) $\exists gestiona^{-} \sqsubseteq Proyecto$
(12) $Proyecto \sqsubseteq (\geq 2\ gestiona^{-})$

La siguiente interpretación I es un modelo de este **ABox**:

$Empleado^{\mathcal{I}} = \{german, laura, christian, gerardo, claudio\}$
$Manager^{\mathcal{I}} = \{german, gerardo, claudio\}$
$AreaManager^{\mathcal{I}} = \{laura\}$
$TopManager^{\mathcal{I}} = \{german, gerardo\}$

$Proyecto^{\mathcal{I}} = \{crowd, linkeddata\}$
$gestiona^{\mathcal{I}} = \{(german, crowd), (gerardo, crowd)\}$
$trabajaPara^{\mathcal{I}} = \{(christian, crowd), (claudio, crowd), (german, crowd), (gerardo, crowd)\}$

Del mismo modo, podemos también dar una nueva interpretación que no sea modelo del ABox:

$Empleado^{\mathcal{I}} = \{german, laura, christian, gerardo, claudio\}$
$Manager^{\mathcal{I}} = \{german, gerardo, claudio\}$
$AreaManager^{\mathcal{I}} = \{laura, gerardo\}$
$TopManager^{\mathcal{I}} = \{german, gerardo\}$
$Proyecto^{\mathcal{I}} = \{crowd, linkeddata\}$
$gestiona^{\mathcal{I}} = \{(german, crowd), (gerardo, crowd)\}$
$trabajaPara^{\mathcal{I}} = \{(christian, crowd), (claudio, crowd), (german, crowd), (gerardo, crowd)\}$

Dónde observamos que la inclusión de `gerardo` como `AreaManager` viola el axioma 5, ya que `gerardo` es también incluido en la extensión de `TopManager`.

■

2.1.2. Familia de Lógicas Descriptiva

Para muchas aplicaciones, el poder expresivo de las lógicas $\mathcal{ALC}$ y $\mathcal{ALCQI}$ puede no ser suficiente. Por lo tanto, para dar lugar a lógicas más expresivas, otros constructores fueron introducidos en [SSS91] junto con una propuesta de un primer esquema de nombres para DLs. Partiendo de los provistos por

la lógica $\mathcal{AL}$, la incorporación de otros constructores es indicado, mediante la concatenación al nombre de la lógica, las letras que representan a cada constructor agregado. Así, $\mathcal{ALC}$ es obtenido desde $\mathcal{AL}$ junto con el constructor de complemento ($\neg$).

Sin detallar totalmente la sintaxis y semántica de cada una, en la Tabla 2.1 se resumen dichas lógicas junto con sus símbolos representativos. Además, en [BCM+03b] se describen lógicas que no poseen todos los constructores dados por $\mathcal{AL}$, por ejemplo, la DL $\mathcal{FL}^-$ se obtiene a partir de $\mathcal{AL}$ pero sin permitir la negación atómica, $\mathcal{FL}_0$ se obtiene a partir de $\mathcal{FL}^-$, pero quitando los cuantificadores existenciales limitados. Las DLs mencionadas hasta ahora poseen como constructores de conceptos la intersección y restricciones de valor como núcleo común, pero la familia de lógicas $\mathcal{EL}$ excluye las restricciones de valor, sólo la intersección de conceptos y el cuantificador existencial se pueden utilizar. Con la finalidad de evitar nombres muy largos para expresar DLs, la abreviatura $\mathcal{S}$ fue introducida para referirse a las lógicas $\mathcal{ALC}_{R^+}$ (en otras palabras, la lógica $\mathcal{ALC}$ extendida con roles transitivos).

Otros constructores y propiedades de relevancia se describen también con letras, como la letra $\mathcal{H}$ que representa subroles o jerarquía de roles, $\mathcal{O}$ representa nominales, $\mathcal{I}$ representa roles inversos, $\mathcal{N}$ restricciones y $\mathcal{Q}$ restricciones numéricas calificadas. La integración con un dominio/tipo de dato concreto es indicado agregando su nombre entre paréntesis, o una **D** "genérica". Por ejemplo, la DL correspondiente para el lenguaje de ontologías OWL DL que incluye todos los constructores es denominado $\mathcal{SHOIN}(\mathbf{D})$.

2.1.3. Servicios de Razonamiento y Propiedades Computacionales

La importancia de las DLs y su tratabilidad se expresa en la capacidad de consultar el conocimiento representado a través de un conjunto de tareas o servicios de razonamiento para tal fin. Dichas tareas, además pueden ser reducidas, desde un punto de vista lógico, a otras tareas facilitando el

Nombre	Sintaxis	Semántica	Símbolo
Top	$\top$	$\Delta^{\mathcal{I}}$	$\mathcal{AL}$ / $\mathcal{EL}$
Bottom	$\bot$	$\emptyset$	$\mathcal{AL}$ / $\mathcal{EL}$
Intersección	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{AL}$ / $\mathcal{EL}$
Cuantificador existencial limitado	$\exists R.\top$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}$	$\mathcal{AL}$ / $\mathcal{EL}$
Restricción de valor	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$	$\mathcal{AL}$
Negación	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$	$\mathcal{C}$
Restricción num. calificada	$\geq nR.C$ $\leq nR.C$ $= nR.C$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\mid \geq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\mid \leq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\mid = n\}$	$\mathcal{Q}$
Inverso	r^-	$\{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}}\}$	$\mathcal{I}$
Unión	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
Cuantificador existencial	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	$\mathcal{E}$ / $\mathcal{EL}$
Restricción num. no calificada	$\geq nR$ $\leq nR$ $= nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}\mid \geq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}\mid \leq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}\mid = n\}$	$\mathcal{N}$
Mapeo rol-valor	$R \subseteq S$ $R = S$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow (a,b) \in S^{\mathcal{I}}\}$ $\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \leftrightarrow (a,b) \in S^{\mathcal{I}}\}$	
Acuerdo y desacuerdo	$u_1 \doteq u_2$ $u_1 \neq u_2$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}.u_i^{\mathcal{I}}(a) = b = u_2^{\mathcal{I}}(a)\}$ $\{a \in \Delta^{\mathcal{I}} \mid \exists b_1, b_2 \in \Delta^{\mathcal{I}}.u_i^{\mathcal{I}}(a) = b_1 \neq b_2 = u_2^{\mathcal{I}}(a)\}$	$\mathcal{F}$
Nominal	I	$I^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ con $\mid I^{\mathcal{I}} \mid = 1$	$\mathcal{O}$

Cuadro 2.1: Sintaxis y semántica de constructores para la familia de DL. Extraída de [BCM+03a] y ampliada según su texto.

desarrollo de herramientas como la que presentamos en este trabajo. A continuación, detallaremos los servicios más relevantes, a partir de los cuales, clasificaremos las propiedades computacional de cada lógica.

- **Satisfacibilidad de la Base de Conocimiento**. Una KB es satisfacible (o consistente) si tiene un modelo. Por lo tanto, existe una interpretación $\mathcal{I}$ tal que $\mathcal{I}$ es modelo de la KB [BCM+03c, HM01].
- **Inferencia de Axiomas**. Una KB infiere un axioma α si todo modelo de la KB es también un modelo de α. El problema de chequear la inferencia de axiomas puede reducirse a decidir si una KB es satisfacible [HM01].
- **Satisfacibilidad de Conceptos**. Dado una KB, un concepto C es

satisfacible con respecto a la KB si puede ser instanciado. Por lo tanto, existe un modelo $\mathcal{I}$ de KB que mapea C a un conjunto no vacío: $C^{\mathcal{I}} \neq \emptyset$. El problema de decidir la satisfacibilidad de un concepto C puede ser reducido al servicio anterior determinando si KB $\models C \sqsubseteq \bot$ [BCM+03c, HM01].

Decidir si una base de conocimiento es consistente es importante *per se*, ya que un KB inconsistente puede llevar a errores de modelado severos y a la derivación de axiomas inválidos. Una KB donde todos los conceptos son satisfacibles es llamada *coherente*.

Es posible clasificar la complexidad computacional de las lógicas descriptivas en términos de satisfacibilidad de conceptos y axiomas generales de la forma $C \sqsubseteq D$, para conceptos arbitrarios C y D. La Tabla 2.2 presenta una clasificación no exhaustiva con sus respectivas referencias.

DL	Complejidad	Comentario
$\mathcal{ALC}$	PSPACE-completo [SSS91]	
$\mathcal{ALCI}$	PSPACE-completo [Tob01]	
$\mathcal{ALCQI}$	PSPACE-completo [Tob01]	
$\mathcal{SHIF}$	ExpTime-completo [Tob01, Hla04]	
$\mathcal{SHIN}$	ExpTime-completo [Tob01, Hla04]	OWL-Lite [BHS08]
$\mathcal{SHOIQ}(\mathcal{D})_n^-$	$\mathcal{SHOIQ}$ es NExpTime-completo [Tob01, Tob00]	Usado para DIG [BMC03]
$\mathcal{SHOIN}$	NExpTime-completo [Tob01, Tob00]	Usado para OWL-DL [BHS08]
$\mathcal{SROIQ}$	NExpTime-hard [Tob00, HS04, HKS06]	Usado para OWL 1.1 [PSH06, HKS06]

Cuadro 2.2: Complejidad Computacional para satisfacibilidad de conceptos de algunas Lógicas Descriptivas

2.2. Lenguaje de Ontologías Web: OWL 2

La Web Semántica [BLHL01] es una evolución de la web actual donde la información tiene un significado explícito posibilitando el procesamiento automático por parte de las máquinas y la integración de la información

disponible. La Web Semántica intenta integrar la capacidad del lenguaje XML para definir etiquetas y la flexibilidad de RDF para representar datos. Sin embargo, un nivel superior es necesario para describir el significado de los términos usados en estos nuevos documentos web. En este contexto y, debido a la necesidad de ejecutar tareas de razonamiento sobre tales documentos, surge el Lenguaje de Ontologías Web (OWL).

OWL es un lenguaje de representación de conocimiento para la Web Semántica con un significado definido formalmente, ya que está basado en Lógicas Descriptivas. A partir de él, es posible definir ontologías especificando clases, individuos y valores de datos que pueden ser almacenados como documentos web semánticos. Además, estas ontologías pueden incluir información escrita en otros lenguajes, por ejemplo RDF, para la descripción de recursos web, versionado y anotaciones.

Debido a sus grados de expresividad, el estándar OWL fue dividido en los siguientes sublenguajes: OWL Full, OWL DL y OWL Lite. El primero, OWL Full, contiene a los restantes sublenguajes y su expresividad lo vuelve indecidible. Una de las razones de ésta indecidibilidad es la falta de restricciones al momento de combinar individuos, clases y roles. Diferente es el caso de OWL DL el cual es soportado por varias herramientas de software debido a su decidibilidad. Si bien OWL DL incluye todos los constructores OWL, ellos no pueden ser combinados de maneras aleatorias. En este contexto, definiciones de clases como instancias de otras clases no se permiten, entre otras. Finalmente OWL Lite, mantiene la decidibilidad pero es aún menos expresivo que los anteriores. OWL Lite restringe aún más sus definiciones permitiendo sólo, por ejemplo, valores de cardinalidades 0 y 1. Las definiciones completas de estos sublenguajes pueden ser consultadas en [Wor04, Con04].

En particular, OWL 2 [HKP$^+$09, OWL09] es esencialmente una pequeña extensión de su predecesor, OWL 1. Desde un punto de vista sintáctico, OWL 2 introduce una nueva sintaxis llamada *Sintaxis Funcional*, que reemplaza a la sintaxis abstracta de OWL 1. El propósito de esta nueva sintaxis es facilitar la lectura estructural de las ontologías. Otras características sintácticas que

se incluyen son la posibilidad de definir una clase como la unión disjunta de otras (de a pares), del mismo modo que una clase puede definirse a partir de un conjunto de clases disjuntas. Finalmente, también incluye la posibilidad de declarar valores de una propiedad que un individuo no satisface.

Nueva expresividad es adicionada a partir de los siguientes constructores, que no serán profundizados en este documento: claves, cadena de propiedades, mayor expresividad en rangos y tipos de datos, propiedades asimétricas, reflexivas y disjuntas y mejoras en anotaciones. Una restricción importante incluida en OWL 2 y que ha sido considerada para nuestro trabajo es la posibilidad de definir cardinalidades sobre propiedades cualificadas, denotada con $\mathcal{Q}$. A diferencia de las restricciones no cualificadas, $\mathcal{N}$, esta nueva restricción permitirá establecer las clases o rangos para los cuales las instancias son contadas. Por ejemplo, en OWL 2 es posible especificar clases de personas que tiene una cantidad determinada de hijos que son además mujeres.

OWL 2 también mantiene la compatibilidad hacia atrás de OWL 2 DL con respecto al sublenguaje OWL 1 DL, mientras que define los siguientes perfiles explorando otras expresividades y propiedades computacionales, cuyos resultados son tres perfiles explayados en [MGH+12]: OWL 2 EL, OWL 2 QL y OWL 2 RL. En tanto que OWL 2 Full continua con las mismas limitaciones que el OWL 1 Full en referencia a su indecidibilidad.

Con respecto a los nuevos perfiles OWL 2 [MGH+12], OWL EL es utilizado en aplicaciones con ontologías que contienen un gran número de propiedades y/o clases. Captura un poder expresivo utilizado por la mayoría de las ontologías siendo un subconjunto de OWL 2 por lo que las tareas básicas de razonamiento pueden ser realizadas en tiempo polinomial con respecto al tamaño de la ontología. El acrónimo "EL" refleja la familia $\mathcal{EL}$ [BBL05, BLB08] de la lógica descriptiva, las cuales proveen solamente cuantificadores existenciales. A pesar de su acotado conjunto de primitivas, OWL 2 EL provee los constructores suficientemente expresivos para grandes ontologías biomédicas, tal como SNOMED CT [CdK08].

OWL QL se utiliza especialmente para aplicaciones que usan un gran número de instancias de datos, donde las consultas es la tarea de razonamiento más importante. Basado en la familia DL-Lite, usando técnicas de razonamiento consistentes y completas, una consulta conjuntiva puede ser realizada en LOGSPACE [CGL+07] con respecto al tamaño de los datos. El acrónimo "QL" refleja el hecho de que, responder consultas en este perfil, puede ser implementado por medio de una reescritura de las consultas a un lenguaje de consultas("Query Language" en inglés) relacional estándar. OWL 2 QL provee algunas de las características necesarias para expresar modelos conceptuales, diagramas de clases UML y EER. Asimismo, está basado sobre DL-Lite [CGL+07] y tiene importantes aplicaciones en acceso a datos basado en ontologías (OBDA) [GGH+13, XCK+18].

Por último, el objetivo de OWL 2 RL es tratar con aplicaciones que requieren razonamiento escalable sin sacrificar demasiado poder expresivo. Está diseñado para una fácil adopción por parte de los sistemas de inferencia basados en reglas y restringe el uso de constructores a ciertas posiciones sintácticas. El acrónimo "RL" refleja el hecho de que el razonamiento en este perfil puede ser implementado usando un lenguaje de reglas estándares [CNS11].

Sintaxis y Semántica de OWL

La sintaxis primaria para almacenar ontologías es la sintaxis *RDF/XML* [bec04], también basada sobre RDF y por lo tanto, orientada al intercambio de datos. OWL 2 también incluye otras definiciones sintácticas: serializaciones RDF tales como *Turtle* [BBLPC14] cuyo propósito es facilitar la lectura/escritura de tripletas RDF; una serialización XML, *OWL/XML* [OWL09, mot12] para un manejo eficiente de herramientas XML y una sintaxis llamada *Manchester* [OWL09, HPS12] la cual intenta hacer más entendibles los documentos ontológicos.

Para poder razonar sobre las ontologías OWL 2, es necesario especificar

$\mathcal{ALCQI}$	OWL 2	$\mathcal{ALCQI}$	OWL 2

$C, \top$	`<owl:Class IRI="C"/>`
	`<owl:Class`
	`abbreviatedIRI="owl:Thing"/>`
		$A \sqsubseteq B$	`<owl:SubClassOf>`
			`<owl:Class IRI="A"/>`
			`<owl:Class IRI="B"/>`
			`</owl:SubClassOf>`

$A \sqcap B$	`<owl:ObjectIntersectionOf>`
	`<owl:Class IRI="A"/>`
	`<owl:Class IRI="B"/>`
	`</owl:ObjectIntersectionOf>`
		$A \sqcup B$	`<owl:ObjectUnionOf>`
			`<owl:Class IRI="A"/>`
			`<owl:Class IRI="B"/>`
			`</owl:ObjectUnionOf>`

R^-	`<owl:ObjectInverseOf>`
	`<owl:ObjectPropertyOf IRI="R"/>`
	`</owl:ObjectInverseOf>`
		$\neg C$	`<owl:ObjectComplementOf>`
			`<owl:Class IRI="C"/>`
			`</owl:ObjectComplementOf>`

$(\leq n\, R.\top)$	`<owl:ObjectMinCardinality`
	`cardinality="n">`
	`<owl:ObjectProperty IRI="R"/>`
	`<owl:Class`
	`abbreviatedIRI="owl:Thing"/>`
	`</owl:ObjectMinCardinality>`
		$(\geq n\, R.\top)$	`<owl:ObjectMaxCardinality`
			`cardinality="n">`
			`<owl:ObjectProperty IRI="R"/>`
			`<owl:Class`
			`abbreviatedIRI="owl:Thing"/>`
			`</owl:ObjectMaxCardinality>`

$\forall R.C$	`<owl:ObjectAllValuesFrom>`
	`<owl:ObjectProperty IRI="R"/>`
	`<owl:Class IRI="C"/>`
	`</owl:ObjectAllValuesFrom>`
		$\exists R.C$	`<owl:ObjectSomeValuesFrom>`
			`<owl:ObjectProperty IRI="R"/>`
			`<owl:Class IRI="C"/>`
			`</owl:ObjectSomeValuesFrom>`

Cuadro 2.3: Referencias de los axiomas OWL 2 y su significado en $\mathcal{ALCQI}$. Extraído de [bao12, HKP+09].

su semántica. Actualmente, OWL 2 presenta dos semánticas estrechamente relacionadas: *Semántica Directa* (Direct Semantics) y *Semántica Basada en RDF* (RDF-based Semantics).

La semántica directa define el significado de los axiomas OWL mediante relaciones directas a las Lógicas Descriptivas. Esto resulta en una semántica compatible con la semántica de modelos de $\mathcal{SROIQ}$ y en la posibilidad de utilizar herramientas para modelado ontológico integradas a sistemas de razonamiento. Por otro lado, la semántica basada en RDF primero traduce axiomas OWL en grafos dirigidos expresados en RDF. Esta semántica es completamente compatible con la de RDF y puede ser también aplicada a cualquier ontología OWL 2 sin restricciones. Finalmente, la relación entre ambas es estrecha ya que dada una ontología OWL 2 DL, las inferencias hechas usando la semántica directa serán también válidas si dicha ontología es mapeada a grafos RDF e interpretada usando la semántica basada en RDF. Este último resultado ha sido publicado en [sch12]. La Tabla 2.3 muestra un correspondencia entre la lógica $\mathcal{ALCQI}$ y los constructores de OWL 2.

Ejemplo 2.3 Considere la Tabla 2.4 donde se identifica las partes relevantes de un documento OWL/RDF por medio de una ontología usando dicha sintaxis. Los nombres e identificadores que se utilizan son mayormente IRIs. La primer sección son los espacios de nombres XML a utilizar y las abreviaturas asociadas a ellas. Los espacios de nombres de este ejemplo corresponden a OWL, RDFS (RDF Schema), RDF, XSD (XML Schema).

En "Información Básica" se encuentra información que se considera útil para varias aplicaciones. También se provee el nombre, que usualmente es la ubicación en la Web en la que está disponible. Es común reutilizar información que está dentro de otra ontología importándola. En el ejemplo la ontología se denomina `http://example.com/owl/families` y se reutiliza los datos contenidos en `http://example.org/otherOntologies/families.owl`.

Luego, se describe el nivel conceptual como los conceptos (`owl:Class`) y los roles (`owl:ObjectProperty` y `owl:DatatypeProperty`). En este caso, se define el concepto "Woman" como incluido dentro de "Person", los roles "hasParent" y su inverso "hasChild", el cual debe ser disjunto con "hasSpouse". También, se describe otro rol denominado "hasAge" cuyo dominio es "Person" y rango un tipo de dato "nonNegativeInteger" descripto en XML Schema. Finalmente, el ejemplo describe un individuo del concepto "Woman" llamado "Mary". Indica que, en la ontología importada, "Mary" es el mismo individuo descripto allí como "MaryBrown".

En DL, se describen los conceptos y roles de la siguiente manera:

$$Woman \sqsubseteq Person$$
$$hasParent^{-} \equiv hasChild$$
$$hasParent \sqcap hasSpouse \equiv \emptyset$$
$$(\exists hasAge.\top \sqsubseteq Person)$$
$$(\exists hasAge^{-}.\top \sqsubseteq nonNegativeInteger)$$
$$Woman(Mary)$$

■

Espacios de Nombres	`<rdf:RDF xml:base="http://example.com/owl/families/"`
	`  xmlns="http://example.com/owl/families/"`
	`  xmlns:otherOnt=`
	`  "http://example.org/otherOntologies/families/"`
	`  xmlns:owl="http://www.w3.org/2002/07/owl#"`
	`  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"`
	`  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`
	`  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">`

Información Básica	`<owl:Ontology rdf:about="http://example.com/owl/families">`
	`  <owl:imports rdf:resource=`
	`  "http://example.org/otherOntologies/families.owl" />`
	`</owl:Ontology>`

Conceptos	`<owl:Class rdf:about="Woman">`
	`  <rdfs:subClassOf rdf:resource="Person"/>`
	`</owl:Class>`

Roles	`<owl:ObjectProperty rdf:about="hasParent">`
	`  <owl:inverseOf rdf:resource="hasChild"/>`
	`  <owl:propertyDisjointWith rdf:resource="hasSpouse"/>`
	`</owl:ObjectProperty>`
	`<owl:DatatypeProperty rdf:about="hasAge">`
	`  <rdfs:domain rdf:resource="Person"/>`
	`  <rdfs:range rdfs:Datatype=`
	`  "http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>`
	`  <owl:equivalentProperty rdf:resource="&otherOnt;age"/>`
	`</owl:DatatypeProperty>`

Individuos	`<Person rdf:about="Mary">`
	`  <rdf:type rdf:resource="Woman"/>`
	`  <owl:sameAs rdf:resource="&otherOnt;MaryBrown"/>`
	`</Person>`

	`</rdf:RDF>`

Cuadro 2.4: Partes de un documento OWL RDF para una ontología simple. Extractos de OWL RDF obtenidos de [HKP+09].

Para finalizar esta sección, resumimos en la Tabla 2.5, los distintos perfiles de OWL junto con su complejidad computacional para cada situación.

Lenguaje	Problema de Razonamiento	Complejidad Taxonómica	Complejidad Datos	Complejidad Consultas	Complejidad Combinada
OWL 2 con semántica basada en RDF	Consistencia ontológica, Subsume en expresiones de clases, Chequeo de instancia, Consulta conjuntiva	Indecidible	Indecidible	Indecidible	Indecidible
OWL 2 con semántica directa	Consistencia ontológica, Satisfacibilidad de expresiones de clases, Subsume en expresiones de clases, Chequeo de instancias	N2EXPTIME-completo	Decidible pero de complejidad abierta (NP-Hard)	No Aplicable	N2EXPTIME-completo
	Consulta conjuntiva	Decibilidad Abierta	Decibilidad Abierta.	Decibilidad Abierta	Decibilidad Abierta
OWL 2 EL	Consistencia ontológica, Satisfacibilidad de expresiones de clases, Subsume en expresiones de clases, Chequeo de instancias	PTime-completo	PTime-completo	No Aplicable	PTime-completo
	Consulta conjuntiva	EXPTIME	PTime-completo	NP-completo	EXPTIME
OWL 2 QL	Consistencia ontológica, Satisfacibilidad de expresiones de clases, Subsume en expresiones de clases, Chequeo de instancias	NLogSpace-completo	AC^0	No Aplicable	NLogSpace-completo
	Consulta conjuntiva	NLogSpace-completo	AC^0	NP-completo	NP-completo
OWL 2 RL	Consistencia ontológica	PTime-completo	PTime-completo	No Aplicable	PTime-completo
	Satisfacibilidad de expresiones de clases, Subsume en expresiones de clases, Chequeo de instancias	PTime-completo	PTime-completo	No Aplicable	co-NP-completo
	Consulta conjuntiva	PTime-completo	PTime-completo	NP-completo	NP-completo
OWL 1 DL	Consistencia ontológica, Satisfacibilidad de expresiones de clases, Subsume en expresiones de clases, Chequeo de instancias	NEXPTIME-completo	Decidible pero de complejidad abierta (NP-Hard)	No Aplicable	NEXPTIME-completo
	Consulta conjuntiva	Decibilidad Abierta	Decibilidad Abierta	Decibilidad Abierta	Decibilidad Abierta

Cuadro 2.5: Complejidad computacional para los distintos perfiles de OWL 2. Tabla extraída de [OWL12].

2.3. Ingeniería Ontológica

Las ontologías son tecnologías claves para el desarrollo de la Web Semántica, sin embargo, su éxito se ha diversificado y se han vuelto esenciales en otros campos de aplicación: biología [ZBF17], astronomía [CKA+18], medicina [CdK08], *e-science* [PS18], entre muchos otros. Asimismo, en las principales empresas de tecnología, tales como Oracle, tiene aplicaciones en la gestión de datos semánticos.

Estas novedosas y cada vez más críticas aplicaciones de las ontologías, requieren asegurar ciertos niveles de calidad de estos modelos. El conjunto de actividades que gobiernan su construcción es lo que se conoce como *ingeniería ontológica* y refiere a principios, métodos, metodologías y herramientas que guían y den soporte al desarrollo de ontologías.

Metodologías

Como en la ingeniería de software, existen métodos y metodologías para guiar el desarrollo de ontologías. Si bien existen varias propuestas, abarcan las siguientes actividades, esquematizadas en la figura 2.1:

- *Gestión de Ontologías*: incluye actividades de planificación, control y aseguramiento de la calidad. Las primeras identifican las tareas que serán ejecutadas y los recursos necesarios. El control garantiza que las tareas planificadas sean realizadas y finalmente, la última actividad asegura que se cumplan ciertos parámetros de calidad.

- *Desarrollo y Soporte*: abarca actividades relacionadas al análisis del dominio, esto implica la definición de escenarios, preguntas de competencia, y la búsqueda de soluciones existentes; conceptualización del modelo y planteo de una integración y/o extensión de soluciones existentes; e implementación, la cual se refiere al desarrollo de la ontología en un lenguaje de representación basado en lógica. Durante estas acti-

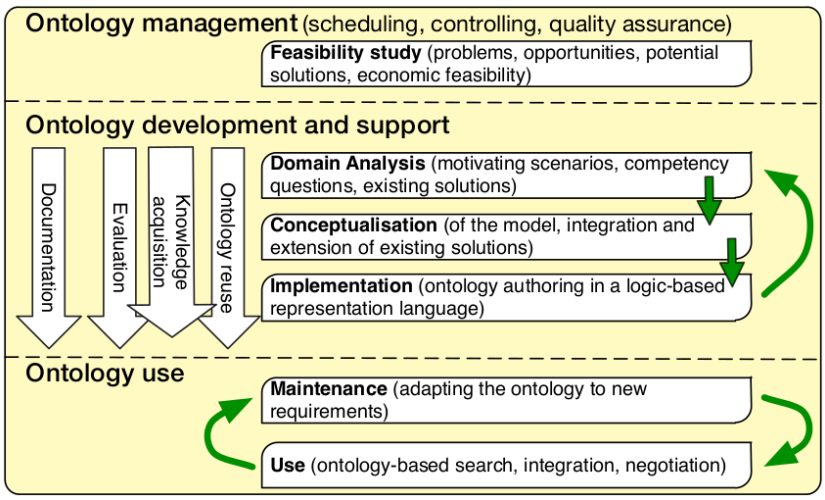


Figura 2.1: Principales tareas de ingeniería ontológica, extraída de [Kee18].

vidades se puede proceder de manera secuencial (en un primer intento), y luego comenzar a iterar sobre ellas en base a los resultados obtenidos.

- *Uso de la Ontología*: estas últimas actividades también se deben considerar dentro de un ciclo iterativo. El mantenimiento es la actividad de actualizar y corregir/perfeccionar el modelo ontológico según nuevos requerimientos y que depende, en ocasiones, del uso/reuso de este modelo por otras ontologías y/o aplicaciones.

- *Actividades de Soporte*: pueden ejecutarse durante la etapa de desarrollo y soporte e incluyen: adquisición de conocimiento [LV14], evaluación [Vra10], documentación [Gar17], y reuso de ontologías [OCRMGR15].

Existen varias propuestas de metodologías en la literatura. Entre ellas, vamos a desatacar a METHONTOLOGY [FLGPJ97], cuyo objetivo es el desarrollo de ontologías desde cero, reusando otras o haciendo una reingeniería de ellas. Esta metodología provee una ciclo de vida basado en prototipos evolutivos y ofrece también técnicas particulares para cada actividad. Su fase más importante en el proceso de desarrollo es la tarea de conceptualización ya que es donde se organiza y estructura el conocimiento adquirido (en las actividades de soporte correspondientes), usando una representación cercana

a los expertos en el dominio. Una vez que el modelo conceptual fue construido, METHONTOLOGY permite la generación automática de un modelo computable.

Una metodología similar es la presentada en [dAF14], llamada SABiO, y enfocada en el desarrollo de ontologías de referencia de dominios. Estás ontologías de referencia proveen modelos conceptuales independientes de una solución y permiten hacer descripciones más precisas. Una vez que los usuarios acuerdan sobre esta ontología, se procede a generar modelo operativo para ser procesado. Las principales actividades de SABiO son: Elicitación de Requerimientos e Identificación del Propósito de la Ontología, Captura y Formalización, Diseño, Implementación y Testing. La primera, involucra a los ingenieros de ontologías, expertos del dominio y potenciales usuarios y su salida es un conjunto de preguntas de competencia (CQ), que ayudarán a la conceptualización. Luego, en Captura y Formalización, deben identificarse los principales conceptos y relaciones, y la salida es un diagrama OntoUML basado en un modelo general llamado UFO [Gui05, GW10]. Este modelo es transformado en un modelo operativo, expresado en un lenguaje de ontologías estándar (por ejemplo, OWL 2) para finalmente ser implementada. Por último, el aseguramiento de calidad es SABiO es gestionada por las CQs definidas previamente. Estos tests son implementadas como consultas sobre una ontología instanciada.

Herramientas

Las herramientas para el soporte al desarrollo de ontologías puede dividirse en dos grupos: aquellas cuyo modelos de conocimiento mapea a un lenguaje de ontologías, como por ejemplo OilEd [BHGS01], SWOOP [KPS+05] y KAON2 [MS05], entre otras. El segundo grupo abarca herramientas que integran diversas características sobre una arquitectura extensible y que, usualmente, son independientes de los lenguajes de ontologías. En este grupo se pueden incluir a Protégé [KFNM04] y WebODE [ACFLGP01]. El capítulo 6 incluye un análisis comparativo de muchas de estas herramientas, junto con

la descripción y alcance de cada una.

Otras herramientas, además de los ambientes de ingeniería descriptos, son las que implementan las actividades de soporte de las metodologías ontológicas. En particular, vamos a resaltar DL-Learner [BLW16], para adquisición de conocimiento basado en aprendizaje automático; WIDOCO [Gar17] para documentación; y herramientas de alineación para reuso de ontologías [OCRMGR15].

Finalmente, un grupo importante de herramientas, que soportan el desarrollo de ontologías, abarca a aquellas relacionadas al procesamiento de modelos en un lenguaje operativo. Los razonadores lógicos, tales como Racer [HM01], HermiT [SMH08], Pellet [SPG+07] y FaCT++ [TH06], son componentes de software que proveen servicios de razonamiento a otras aplicaciones e infieren consecuencias lógicas desde un conjunto de hechos explícitamente asertados. Estos razonadores tiene varios atributos para destacar: algoritmos de inferencias (generalmente, tableaux), expresividad lógica, protocolos de comunicación (DIG [BMC03], OWLlink [LLNW11]) para acceder a sus servicios, grado de completitud y lenguaje de implementación.

2.4. Conclusiones

Las DLs son las bases lógicas para el lenguaje de ontologías OWL 2, el cual es central para el desarrollo de la Web Semántica. Su objetivo es permitir a humanos y sistemas de computación intercambiar ontologías de manera no ambigua y posibilitar la inferencia de conocimiento implícito a partir del definido en dichas ontologías.

En este sentido, desarrollar y mantener ontologías es una tarea compleja para expertos del dominio de interés que no están familiarizados con formalismos lógicos. Para abordar estas tareas, las metodologías para ingeniería ontológica se presentan como un marco de trabajo para ordenarlas, junto con herramientas para que los usuarios puedan interactuar con sus modelos

y hacer uso de los servicios de razonamiento para validarlos. Estos ambientes deberán incorporar aspectos formales y metodológicos para integrar las intenciones de los expertos con la semántica formal de los lenguajes de ontologías para favorecer la comprensión de los modelos.

Este capítulo es la base teórica sobre DL, ontologías y sus actividades y herramientas para desarrollarlas, sobre las que se trabajará durante las siguientes instancias de esta investigación.

Capítulo 3

Gestión de Ontologías Gráficas

Contents

Responder a consultas tales como *¿cuál es una adecuada representación para un diagrama?* es complejo desde el punto de vista estrictamente formal, ya que no solo involucra el análisis de la sintaxis y semántica de los lenguajes sino que también su efectividad cognitiva [Moo06] y los aspectos de calidad [KS02]. Aún así, el tamaño de los diagramas y la capacidad de las herramientas gráficas para visualizarlos, agrega otra dimensión al análisis. En consecuencia, es esencial la definición de un proceso de visualización que coordine estas dimensiones. Dicho proceso debe considerar tres principios

claves para la producción de diagramas efectivos [Moo06]: discriminabilidad, control de la complejidad y simplicidad gráfica.

Este capítulo aborda el problema de acercar los usuarios, expertos de dominios y diseñadores en una herramienta visual que permita la producción y manipulación de ontologías gráficas que cumplen ciertos criterios de calidad. El objetivo es formalizar estos aspectos en un proceso para luego ser implementado en un herramienta web para ingeniería. Asimismo, aborda y fundamente los siguientes requerimientos planteados para nuestro trabajo:

- ***(R2)*** *Edición Visual*: La edición visual es una tarea orientada a humanos desde la cual los expertos del dominio, que poseen un conocimiento profundo de la semántica de sus entornos, podrían tomar ventaja para reducir la carga cognitiva involucrada en tareas de modelado complejo [FFT12, War04].

- ***(R6)*** *Servicios de Razonamiento Integrados*: Las aplicaciones de las ontologías se extendieron más allá de la web, convirtiéndose en un artefacto clave en diversos campos y en empresas de tecnología para la gestión de datos. En este sentido, explotar las técnicas de razonamiento automático para asegurar la correctitud de modelos es esencial en el desarrollo de ontologías de calidad. Asimismo, es de importancia reflejar los resultados de dicho razonamiento no solo en el modelo formal, sino también en su visualización [FFT12].

- ***(R7)*** *Independencia de los Lenguajes Visuales*: Los lenguajes de modelado conceptual (CDMLs) pueden variar de un usuario a otro dependiendo su experiencia y diferencias culturales. A pesar de esto, estos presentan características similares, compartiendo una semántica común para un subconjunto de elementos de dichos lenguajes [KF15].

- ***(R8)*** *Manejo Simultáneo de Modelos Visuales*: La independencia de los lenguajes visuales permite dotar a las herramientas de diversos CDMLs para representar la misma ontología e integrar modelos expresados en

diferentes lenguajes mediante la definición de restricción intermodelos [FFT12].

Resultados del proceso de visualización presentado aquí fueron publicados en [BGCF16], mientras que conceptos teóricos preliminares referidos a la manipulación de ontologías gráficas fueron presentados en [BCF15].

Previo a detallar la contribución central de esta Tesis, y habiendo descripto la teoría subyacente a las DLs, vamos a definir la terminología que utilizaremos en lo que resta del documento. En este trabajo, una **ontología** es definida como equivalente a una base de conocimiento en DL [HPSvH03b]. Asimismo, un **modelo conceptual** describe la semántica de los datos de un sistema en un nivel alto de abstracción, utilizando los **lenguajes de modelado conceptual de datos (CMDLs)** estándar: UML , ER/EER y/o ORM/ORM2.

Por último, las ontologías difieren de los modelos conceptuales de datos en que las primeras deben ser aprobadas por una comunidad de uso, mientras que los modelos conceptuales son cerrados a un dominio particular. Sin embargo, desde el punto de vista de este trabajo, utilizaremos ambos conceptos de manera indistinta.

3.1. Visualización de Modelos Conceptuales

El modelado conceptual es considerada una actividad cognitiva y, como tal, está relacionada al proceso de conocer y entender modelos. Las personas que interactúan con sistemas cognitivos son capaces de comprender grandes cantidades de datos, utilizando capacidades inherentes a los humanos para reconocer patrones y tomar decisiones con ellos [War04]. En este sentido, la visualización juega un rol crucial colaborando en la percepción de estos patrones, que no han sido previamente capturados. Además, la visualización permite identificar problemas en los modelos y los datos y focalizar la aten-

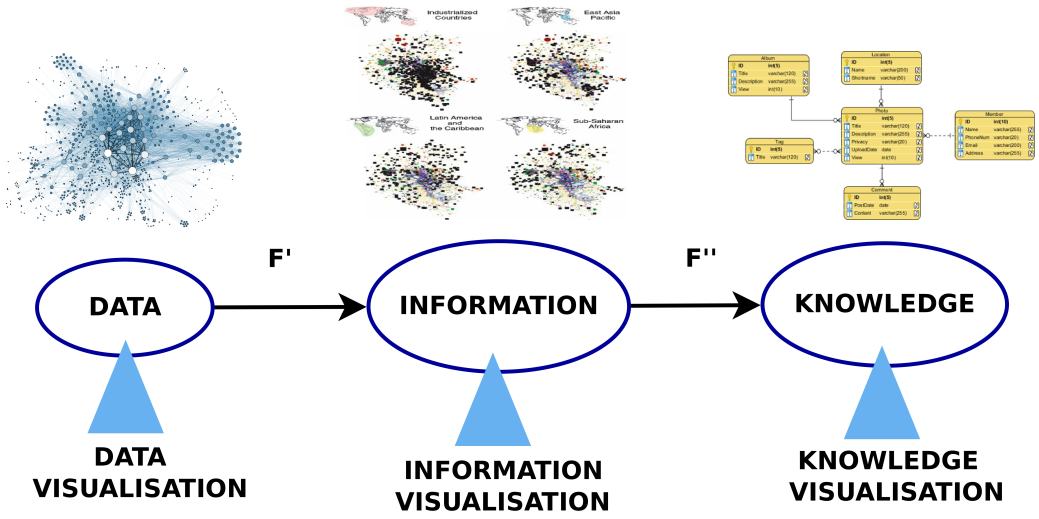


Figura 3.1: Visualizaciones: F' y F'' representan cualquier transformación entre datos, información y conocimiento.

ción sobre el dominio siendo modelado, decrementando la carga cognitiva de los diseñadores.

Los aspectos de calidad de los modelos conceptuales deben también ser considerados debido a su efecto sobre los productos finales. En este sentido, diversas evaluaciones sobre la correspondencia entre el dominio real y su modelo conceptual están basadas en interpretaciones sociales y técnicas, las cuales a su vez definen la calidad pragmática [ST05, KS02]. De esta forma, la cognición humana tiene un rol central en asegurar la calidad de los modelos y la visualización es una manera efectiva para lograrlo.

Según Card [CMS99], la visualización es *"el uso de representaciones de datos visuales, interactiva y soportada por una computadora, con el fin de amplificar la cognición"*. En consecuencia, la visualización es clave para obtener "buenos" modelos conceptuales [Moo06], para los cuales debemos considerar "buenas" notaciones visuales [Moo09] y su semántica apropiada [HR04]. Además, permite comprender grandes cantidades de datos, percibir propiedades anticipadas y errores, y facilitar el definición de hipótesis.

En este contexto, los datos son solo símbolos y hechos no interpretados,

mientras que se convierten en información cuando son interpretados y procesados, asignándole significado. El conocimiento es un concepto un poco más sofisticado debido a que es información *"cognitivamente procesada e integrada en una estructura de conocimiento existente en un humano"* [KT05]. Mediante esta distinción, es posible definir claramente niveles de visualización [MVC+10], ilustrados en la Figura 3.1, que se corresponden con los conceptos previos.

Los niveles de datos e información son conceptos más cercanos ya que ellos refieren al uso de representaciones gráficas para mostrar relaciones en datos, aunque la visualización de información es restringida al soporte computacional [CMS99, War04]. A diferencia de los anteriores, la visualización del conocimiento usa representaciones gráficas para transferir, crear y compartir conocimiento. Como consecuencia, su gestión en el diseño de los sistemas de información impulsa la comunicación intensiva entre los involucrados en el proceso de modelado y la aplicación de nuevas metodologías en dominios reales.

La conceptualización de un dominio puede ser formalizado por medio de ontologías y así evaluar representaciones concretas de dichos dominios, en términos de un lenguaje de modelado conceptual [Gui05] y la capacidad de éstos para razonar automáticamente [CGL+98]. El modelado conceptual basado en ontologías se vuelve esencial para definir modelos de calidad, decrementando tiempos y costos, y asistiendo a usuarios durante el proceso.

En este sentido, las herramientas son deseables para ayudar a los modeladores de aplicaciones reales, definiendo criterios para manipular las representaciones gráficos, evitar redundancias en la visualización de restricciones implícitas y controlar el impacto de los diagramas en la capacidad de los humanos para comprenderlos, sin generar modelos aún más complejos [BCLW12].

Estos conceptos previos son integrados por Burkhard [Bur05] en un modelo abstracto, basado en el uso de visualizaciones para integrar nuevo cono-

cimiento al de los receptores. El objetivo es analizar y definir los efectos de proveer representaciones visuales y la forma más efectiva para hacerlo.

En este trabajo trasladamos este concepto hacia el modelado conceptual basado en ontologías, al que nombramos *proceso de visualización para modelado ontológico*, a partir de los siguientes lineamientos: integración de visualizaciones con razonamiento lógico para validación; definición de aspectos relevantes para visualizar y cómo esto debería realizarse; y por último, el uso de diagramas para evaluar la calidad del modelado y su correspondencia con los respectivos dominios reales. A continuación, explicitamos las bases para este proceso de visualización y cómo sus componentes interactúan, de manera tal que pueda ser implementado en un herramienta para tareas de modelado.

3.2. Proceso de Visualización de Conocimiento basado en Ontologías

Introducimos la definición de nuestro proceso y detallamos sus componentes e interacción. En la sección siguiente, introducimos los aspectos formales que complementan el proceso y la definición de ontologías gráficas.

Definición 3.1 (Proceso de Visualización de Conocimiento basado en Ontologías) : Un *proceso de visualización de conocimiento basado en ontologías* es una transformación de aserciones ontológicas, posiblemente extraídos desde los datos, en revelaciones por medio de mapeos gráficos y lógicos. El objetivo es capturar el conocimiento desde dominios reales; entenderlos mediante el descubrimiento de relaciones, patrones y anomalías; y finalmente, comunicarlos entre usuarios interesados.

■

Concretamente, nuestro proceso es un bucle iterativo para modelado gráfico de dominios e interacción con usuarios. El primer paso es describir asercio-

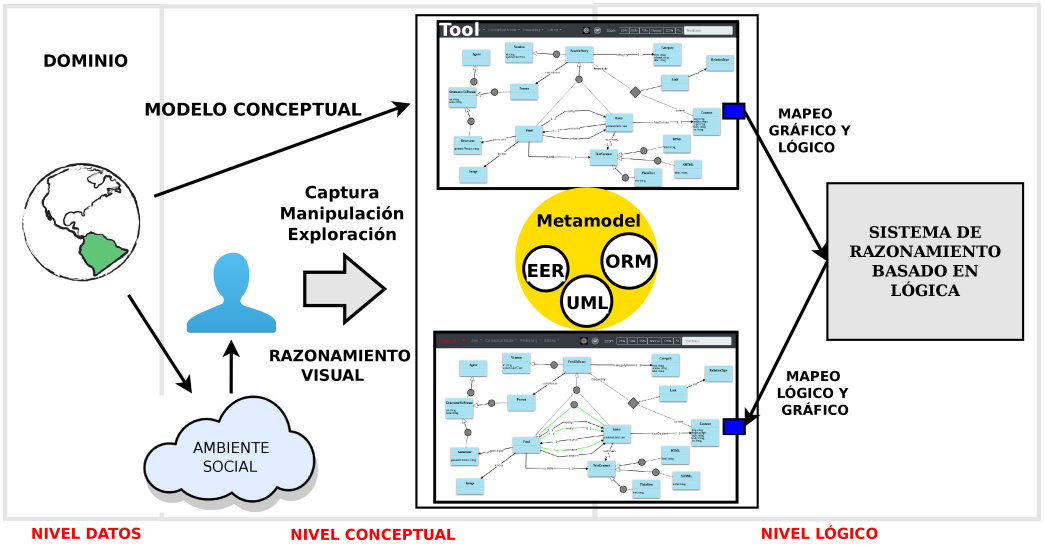


Figura 3.2: Vista Esquemática del Proceso de Visualización de Conocimiento basado en Ontologías.

nes ontológicas como modelos conceptuales en un lenguaje visual y mapear estas visualizaciones en un artefacto lógico (una ontología) y viceversa, posibilitando razonamiento decidible para validarlos. El último paso es mostrar los modelos nuevamente con sus deducciones en el lenguaje gráfico mismo, y permitir al usuarios su manipulación y exploración.

El proceso de visualización está dividido en niveles de datos, conceptual y lógico. Datos incluyen las aserciones ontológicas, relacionados al dominio a modelar, el cual puede también ser afectado por la interpretación del usuario sobre la realidad. Niveles conceptuales involucran a los modeladores y usuarios durante tareas de captura, exploración de conocimiento y manipulación de la visualización. Niveles lógicos incluyen mapeos tanto gráfico-lógicos como lógico-gráficos para proveer suporte formal y guiar el proceso de modelado.

Como muestra la figura 3.2, la notación gráfica está basada sobre los lenguajes estándar para modelado conceptual: EER [Gog94], UML [BRJ05] y ORM [HM08], los cuales a su vez promueven usabilidad e interoperabilidad de modelos y herramientas. Las diferencias existentes entre las ontologías y los modelos de datos han sido abordados por diversos enfoques [AGK06],

Metamodelo	UML	EER	ORM 2
tipo de objeto	clase	entidad	tipo de entidad
relación	asociación	relación	tipo de hecho
Rol	Rol	Rol	Rol
subclasificación	generalización	IsA	subtipo
atributo	atributo	atributo	✗
tipos de objetos disjuntos	disjuntos	disjuntos	subtipos exclusivos
completitud	completo	total	subtipos exhaustivos

Cuadro 3.1: Correspondencia entre las principales primitivas del metamodelo KF y las de los respectivos lenguajes que lo integran [KF15] .

sin embargo, todas confluyen en la necesidad de formalizar modelos (UML, EER, ORM) para expresar ontologías y, por otro lado, en las capacidades de razonamiento lógico, provisto por los lenguajes de ontologías, sobre diagramas.

A pesar de las obvias diferencias entre la expresividad de los lenguajes de modelado conceptual y los lenguajes de ontologías, herramientas existentes validan esta convergencia [FFT12, CH10, COLS12, HJ12], y la efectividad de usar sintaxis gráfica basada en estos lenguajes gráficos para expresar ontologías y soportar su ingeniería [ABAG17, VBJS14]. Finalmente, evidencias de esta afirmación indican que el núcleo común de los lenguajes UML, EER y ORM 2 puede ser expresado con la DL $\mathcal{ALNI}$ garantizando razonamiento tratable sobre ellos [FK15]. Además, cada lenguaje puede expresar $\mathcal{ALCQI}$, manteniendo también las capacidades de razonamiento.

Estos lenguajes forman parte de un metamodelo, llamado KF [KF15], el cual se encarga de capturar las diferencias y similitudes de estos lenguajes, a partir de un análisis ontológico de las respectivas primitivas. La unificación de estos CDMLs constituye una base teórica sólida que soporta estos tres lenguajes, facilitando la definición de aserciones intermodelos (representados en diferentes lenguajes) y la conversión entre ellos. Las principales entidades de este metamodelos son: *tipos de objetos, relaciones, relaciones de subclasificación, atributos* y *restricciones*. La tabla 3.1 resume estas entidades y sus correspondencias en los lenguajes UML, EER y ORM 2 que integran el

metamodelo.

Los tipos de objetos denotan conjuntos de objetos, llamados instancias, con características comunes. Se corresponden con clases, entidades y tipos de entidades en UML, EER y ORM 2, respectivamente. Asimismo, clases y entidades pueden tener atributos, los cuales son tipos de relaciones entre instancias de clases y tipos de datos.

Un relación denota un conjunto de tuplas compuestas por instancias de los diferentes tipos involucrados en tal relación. Se corresponden con asociaciones de UML, relaciones de EER y tipos de hechos de ORM 2. Cada tipo tiene un rol en la relación y una cardinalidad para limitar la cantidad de instancias de un tipo dado pueden participar en la relación.

La subclasificación es un tipo especial de relación entre tipos de objetos o relaciones en las cuales un tipo de objeto/relación subsume a otro/a. Estás se corresponden a la generalización en UML, IsA en EER y subtipos en ORM 2. Asimismo, es posible definir restricciones de disyunción y completitud entre los tipos que componen la subclasificación. En el primer caso los subtipos son mutualmente excluyentes, mientras que en el segundo, un subtipo padre es la unión de sus subtipos.

Con respecto a los mapeos especificados en el proceso, el objetivo es coordinar diferentes formas de codificar primitivas gráficas de los CDMLs en un formalismo lógico decidible. La ventaja principal de estas transformaciones es la formalización de la interacción gráfica y lógica para definir la semántica precisa de cada modelo, dotarlo de razonamiento lógico y mostrar al usuario posibles inconsistencias o restricciones ocultas. La formalización de esta interacción será detallada en la sección siguiente.

Desde una perspectiva de los modeladores, la captura de conocimiento en la figura 3.2, representa la tarea de tomar aspectos de un dominio real y representarlos gráficamente, dónde la percepción del humano sobre la realidad es central y condiciona la recolección e interpretación de su conocimiento. La manipulación considera el manejo de estructuras visuales, en particular,

por medio de interfaces gráficas. Los criterios para proveer interfaces amigables están basados en la disminución del esfuerzo cognitivo a través de técnicas de navegación, visualización y *layout* automático [War04]. Por último, la exploración no es un aspecto gráfico en el contexto del proceso de visualización e incluso no accesible para usuarios que no trabajan con lógica, sino que representa las definición o elección de otras transformaciones (o mapeos) que capturan la semántica de los modelos, posiblemente aumentando la expresividad de los lenguajes formales subyacentes.

A continuación, damos la formalización de los mapeos gráficos y lógicos para completar la descripción de nuestro proceso de visualización. Asimismo, introducimos el concepto de *ontología gráfica* y damos un marco teórico para definirlas y manipularlas.

3.3. Sistema para Manipulación de Ontologías Gráficas (*GOMS*)

Con el objetivo de integrar razonamiento a ambientes gráficos, presentamos en este capítulo un modelo teórico independiente del formalismo lógico y metodología gráfica. El principal aporte de este sistema teórico, al que llamaremos *Sistema de Manipulación de Ontologías Gráficas (GOMS)*, es la generación de transformaciones, basadas en lógica, de los lenguajes visuales y viceversa, que permiten la definición de *ontologías gráficas*. Esto redundará en una generalización de la coordinación entre la representación visual de una ontología y su descripción en un lenguaje formal, requerida por las metodologías de ingeniería descriptas en el capítulo anterior. Un *GOMS* fuerza la definición precisa para cada primitiva gráfica de un lenguaje visual para permitir razonamiento sobre tales lenguajes. Asimismo, generaliza un conjunto de servicios de razonamiento que, cualquier herramienta generada a partir de un *GOMS*, debe proveer.

Formalizar la interacción gráfica y lógica, como se propone en un *GOMS*,

tiene importantes beneficios. Primeramente, puede ser usada como una base para el diseño y la implementación de novedosas aplicaciones gráficas para ingeniería ontológica, aportando una terminología estándar y la comprensión de los requerimientos para este tipo de ambientes. Por otro lado, una formalización permite la evaluación y clasificación de ambientes existentes de acuerdo a como ellos gestionan estos aspectos y, por último y no menos importante, es útil para caracterizar, de manera precisa, el poder expresivo de cualquier herramienta en ambas direcciones: visual y formal.

El uso de elementos visuales para modelado es deseable por usuario, pero puede ocultar consecuencias formales que podrían no ser fácilmente reconocidos por diseñadores en dominios complejos. Por ejemplo, la inconsistencia de los modelos o algunos de sus elementos. Así, la integración de razonamiento lógico en herramientas es necesaria para identificar estas propiedades o notificar al usuario sobre estos problemas. Esto es logrado mediante la representación de la semántica de los modelos visuales en términos de un lenguaje formal y luego presentar los resultados a los usuarios en la misma notación gráfica. Este enfoque es formalizado a continuación en las siguientes definiciones.

3.3.1. Ontologías Gráficas

Las siguientes definiciones introducen los aspectos formales de los mapeos (o transformaciones) gráficos y lógicos, en ambas direcciones. Estos conceptos son necesarios para la caracterización de las ontologías gráficas.

Para esto, identificamos un conjunto de elementos gráficos que se corresponden con las primitivas de los CDMLs involucrados en el metamodel KF de referencia. Por lo tanto, cada entidad de dicho metamodelo es un elemento gráfico para nuestro sistema.

Definición 3.2 (Mapeo Gráfico Lógico) : Sea L un lenguaje de modelado conceptual, y Δ_L el conjunto de sus elementos gráficos, en un metamodelo

dado. $\theta_L^{\mathcal{L}}$ es una codificación para algunos elementos de Δ_L a un conjunto de aserciones en un lenguaje formal decidible $\mathcal{L}$. Un *mapeo gráfico lógico* $\Theta_L^{\mathcal{L}}$ es definido desde 2^{Δ_L} a un conjunto de aserciones en $\mathcal{L}$:

$$\Theta_L^{\mathcal{L}}(G) = \bigcup_{x \in G} \theta_L^{\mathcal{L}}(x)$$

■

Definición 3.3 (Mapeo Lógico Gráfico) : Sea L un lenguaje de modelado conceptual, y Δ_L el conjunto de sus elementos gráficos. $\psi_{\mathcal{L}}^L$ es una transformación desde un conjunto de aserciones en un lenguaje formal decidible $\mathcal{L}$ a Δ_L. Un *mapeo lógico gráfico* es definido como una transformación $\Psi_{\mathcal{L}}^L$ desde un conjunto de aserciones $\mathcal{L}$ a 2^{Δ_L}:

$$\Psi_{\mathcal{L}}^L(H) = \bigcup_{H \models x} \psi_{\mathcal{L}}^L(x)$$

■

Intuitivamente, $\theta_L^{\mathcal{L}}$ codifica cada elementos gráfico en una representación lógica capturando la semántica de este elemento, y $\Theta_L^{\mathcal{L}}(G)$ es una codificación de un subconjunto de los elementos gráficos de L. Además, G representa el conjunto de estos elementos soportados por una herramienta. Por otro lado, $\psi_{\mathcal{L}}^L$ transforma un conjunto aserciones lógicas en un elemento visual, y $\Psi_{\mathcal{L}}^L(H)$ asocia una representación visual a una base de conocimiento lógica H, siendo H conocimiento terminológico (en DL, los axiomas de un TBox). Finalmente, de manera pragmática, restringimos nuestra atención a lenguajes formales decidibles debido a que abordaremos implementaciones concretas de estos sistemas de manipulación.

Como consecuencia de las definiciones previas, introducimos ahora un nuevo concepto sobre definición de un modelo visual y cómo éste representa una ontología gráfica si puede ser trasformada a una ontología en un lenguaje

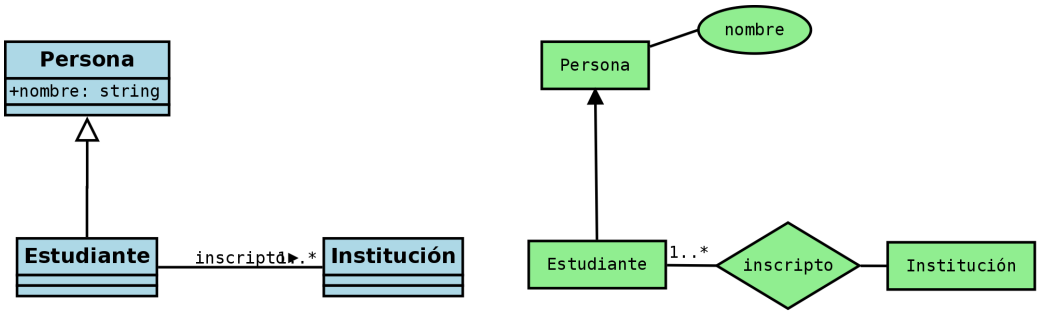


Figura 3.3: Modelos visuales con los mismos elementos gráficos $G \in \Delta$, pero diferente representación en UML y EER, respectivamente

formal.

Definición 3.4 (Ontologías Gráficas) : Sea G un subconjunto de Δ_L, y $\Theta_L^{\mathcal{L}}$ un mapeo gráfico lógico y $\Psi_{\mathcal{L}}^L$ un mapeo lógico gráfico, ambos preservando la satisfacibilidad. G es una *ontología gráfica* con respecto a $\Theta_L^{\mathcal{L}}$ and $\Psi_{\mathcal{L}}^L$ si y solo si $\Theta_L^{\mathcal{L}}(G)$ transforma G a una ontología $\mathcal{O}$ en $\mathcal{L}$, y $\Psi_{\mathcal{L}}^L(\mathcal{O})$ transforma $\mathcal{O}$ a un diagrama en Δ_L.

■

A partir de esta definición, es importante remarcar que una ontología, desde un lenguaje formal decidible, es definida como en [BCM+03a]. Por otro lado, ambos Θ y Ψ no son necesariamente operaciones inversas debido a las diferencias de poder expresivo entre los lenguajes visuales y los formalismos lógicos. Sin embargo, estas trasformaciones cumplen el concepto de *preservación de la satisfacibilidad* [BCD05]. Un elemento G_i es consistente en una ontología gráfica G si y solo si el correspondiente concepto es satisfacible en $\Theta_L^{\mathcal{L}}(G)$. Del mismo modo, un concepto C_i es satisfacible en una ontología $\mathcal{O}$ si y solo si el correspondiente elemento gráfico G_i es consistente en $\Psi_L^{\mathcal{L}}(\mathcal{O})$.

Ejemplo 3.1 Dado el metamodelo KF, los modelos de la figura 3.3 comparten los elementos gráficos $G \in \Delta_{L_i}$ para los lenguajes $L_1 =$ UML y $L_2 =$ EER: tipos de objeto (`Person, Student, Institución`), relaciones (`Inscripto`), atributos (`nombre`) y cardinalidades (`1..N`).

Asimismo, ambos modelos representan ontologías gráficas siguiendo las transformaciones [BCD05] para UML y [ACK+07] para el lenguaje EER.

■

Dada la formalización de ontologías gráficas, presentamos la definición de un sistema para manipular estos modelos visuales representando su semántica usando lógica. Un *GOMS* formaliza y generaliza la correspondencia entre modelos visuales para representación de ontologías, el uso de servicios de razonamiento para validación por medio de reconstrucciones lógicas y cómo la apariencia del diagrama original es modificado con resultados de posibles deducciones.

Definición 3.5 (Sistema para Manipulación de Ontologías Gráficas (GOMS)) : Un *sistema para manipulación de ontologías gráficas (GOMS)* es una tupla:

$$\langle \{G_i\}, \{\mathcal{L}_i\}, \{\Theta_{G_i}^{\mathcal{L}_i}\}, \{\Psi_{\mathcal{L}_i}^{G_i}\} \rangle$$

tal que:

- $\{G_i\}$ un conjunto no vacío de lenguajes de modelado conceptual dado por un metamodelo fijo, por ejemplo, KF [KF15],
- $\{\mathcal{L}_i\}$ un conjunto no vacío de lenguajes formales decidibles para la definición de ontologías,
- $\{\Theta_{G_i}^{\mathcal{L}_i}\}$ mapeos gráficos lógicos que preservan la satisfacibilidad,
- $\{\Psi_{\mathcal{L}_i}^{G_i}\}$ mapeos lógicos gráficos que preservan la satisfacibilidad.

■

Así, un *GOMS* permite generar ontologías desde lenguajes gráficos y formalismos lógicos, a partir de mapeos entre ellos y preservando la satisfacibi-

lidad. Como consecuencia, una ontología gráfica debe ser definida por medio de ambos mapeos, además de ser una instancia del metamodelo KF.

Para profundizar sobre este concepto, es importante destacar que no toda instancia de un *GOMS* define ontologías gráficas tal como las definidas en 3.4. Por ejemplo, no es posible derivar ontologías gráficas a partir de un *GOMS* instanciado con un lenguaje gráfico OVM (Modelos de Variabilidad Ortogonal para Líneas de Productos Software) y una DL. Un modelo OVM puede ser codificado en $\mathcal{ALCI}$, sin embargo, $\mathcal{ALCI}$ no puede ser transformado a un diagrama OVM (Orthogonal Variability Model) [PBvdL05], como demostrado en [BPC+17]. Estos modelos permiten representar la variabilidad en el contexto de una linea de productos software, definiendo restricciones entre servicios, para luego derivar respectivos productos. Dichas restricciones modelan la posibilidad que un servicio pueda o nos ser considerado como parte de un producto software. En este contexto, OVM no define relaciones que permitan modelar la subsumción de servicios, por lo tanto, axiomas DL como $A \sqsubseteq B$, donde A y B son servicios, no pueden ser modelados usando primitivas OVM.

3.3.2. Servicios de Razonamiento en un *GOMS*

La posibilidad de expresar diagramas visuales en lógica permite ampliar el soporte de herramientas gráficas con respecto a tareas (automáticas) de modelado conceptual. En este sentido, el modelador puede usar estas ventajas para validar, formalmente, las propiedades de sus modelos y asegurar su calidad. Algunas de estas propiedades son especificadas para un *GOMS* y están basadas en [BCD05, ACK+07]. Emplearemos las convenciones del nombres del metamodelo KF para nombrar primitivas gráficas, una ontología gráfica G definida por $\Theta_L^{\mathcal{L}}(G)$ y una ontología formal $\mathcal{O}$ a partir de $\Psi_L^{\mathcal{L}}(\mathcal{O})$

- *Consistencia de la Ontología Gráfica.* Una ontología gráfica es consistente si alguno de los tipo de objeto en la ontología puede ser poblada (instanciada) sin violar sus restricciones. Formalmente, G es una onto-

logía gráfica consistente si y solo si $\mathcal{O}$ admite un modelo no vacío en el cual al menos un elemento $x \in G$ tiene una extensión no vacía.

- *Consistencia de Tipos de Objetos.* Un tipo de objeto es consistente si la ontología gráfica admite una instanciación en la cual el tipo de objeto tiene un conjunto no vacío de instancias. Formalmente, $x \in G$ es un elemento gráfico consistente si y solo si $\mathcal{O}$ admite un modelo en el cual el elemento tiene una extensión no vacía.

- *Subclasificación de Tipos de Objetos.* Un tipo de objeto OT subsume un tipo de objeto OT_1 si una ontología gráfica implica que OT es un super conjunto de OT_1. De manera formal, OT *subsume* OT_1 en una ontología gráfica G si y solo si OT es un super conjunto de OT_1 para todos los modelos no vacíos de $\mathcal{O}$.

- *Equivalencia de Tipos de Objetos.* Dos tipos de objetos son equivalentes si ellos denotan el mismo conjunto de instancias siempre que las restricciones de la ontología gráfica no sean violadas. En notación *GOMS*, OT y OT_1 son equivalentes in G si y solo si OT y OT_1 tienen el mismo conjunto de instancias en todo modelo no vacío de $\mathcal{O}$.

- *Consecuencias Gráficas Implícitas.* Una propiedad es una consecuencia (implícita) de una ontología gráfica si se mantienen todos los requerimientos impuestos por la ontología. Casos de consecuencias implícitas incluyen subclasificaciones, equivalentes y cardinalidades. En particular, las cardinalidades pueden ser definidas como: Sea $OT, Rel, Role \in G$, donde OT es un tipo de objeto, Rel es una relación y $Role$ un rol. Entonces OT y Rel denotan a restricción de cardinalidad a través de $Role$ en G si y solo si OT, Rel y $Role$ especifican la cardinalidad más estricta de las instancias relacionadas en todo modelo no vacío de una ontología $\mathcal{O}$.

Las definiciones previas de los servicios de razonamiento concluyen la definición de un sistema completo para la manipulación de ontologías gráficas,

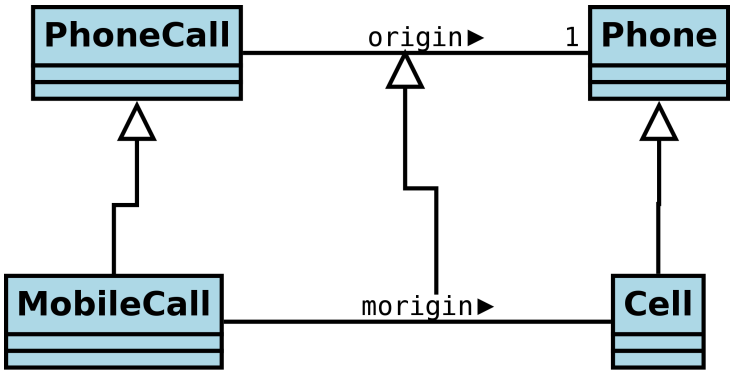


Figura 3.4: Ontología gráfica en UML inicial para ejemplo.

el cual puede ser usado para formalizar e implementar herramientas visuales con razonamiento basado en lógica para ingeniería ontológica.

Concluimos la sección, ilustrando con un ejemplo para un *GOMS* instanciado con el lenguaje gráfico UML y de ontologías OWL 2. Los mapeos incluidos son detallados completamente en el apéndice A.

3.3.3. Ejemplos de Manipulación de Ontologías Gráficas

Un *GOMS* permite formalizar operaciones de manipulación de ontologías gráficas, basadas en el uso de razonamiento automático para validación de ontologías y derivación de propiedades implícitas en ellas, y en la interoperabilidad de ontologías representadas mediante diferentes lenguajes visuales por medio de la aplicación de mapeos definidos en el metamodelo KF. A continuación detallamos ejemplos para cada una de las operaciones de un *GOMS*.

Razonamiento sobre Ontologías Gráficas

Consideramos la ontología gráfica adaptada de [FFT12], modelada en lenguaje UML, como muestra la figura 3.4, donde se representa que todo

`MobileCall` es un `PhoneCall` y que toda `Cell` es un `Phone`. Los primeros están relacionados a través de la asociación binaria `origin`, mientras que los segundos por `morigin`, las cuales, además tienen una relación de subclasificación entre ellas. La cardinalidad de la asociación `origin` expresa que todo `PhoneCall` debe tener exactamente un `Phone`.

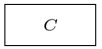
Para el desarrollo del ejemplo, definimos una instancia de un GOMS por medio de subconjuntos respectivos de UML, como lenguaje gráfico y OWL 2 como el lenguaje formal. Los mapeos correspondientes están definidos completamente en el apéndice A, sin embargo, en las tablas 3.3.3 y 3.3.3, damos la semántica de las primitivas UML en DL y aplicada para el ejemplo. Asimismo, los mapeos concretos Θ_{UML}^{OWL2} en sintaxis OWL 2 y los mapeos Ψ_{OWL2}^{UML} por medio de consultas SPARQL-DL [KGH11].

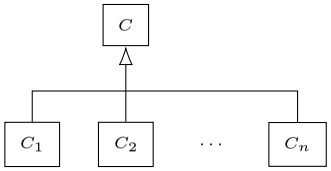
Una ontología $\mathcal{O}$ es derivada desde el modelo visual inicial, en la figura 3.4, por medio de la codificación Θ_{UML}^{OWL2}. Asimismo, el diagrama puede generarse nuevamente desde una ontología $\mathcal{O}'$ en OWL 2 por medio de Ψ_{OWL2}^{UML}, luego de ejecutar razonamiento automáticos sobre $\mathcal{O}$. Las consultas que implementan los mapeos lógico-gráficos, Ψ_{OWL2}^{UML}, retornan todas las clases y subclases, el dominio y rango para todas las propiedades (`objectProperty`), y la jerarquía de propiedades en la ontología $\mathcal{O}'$.

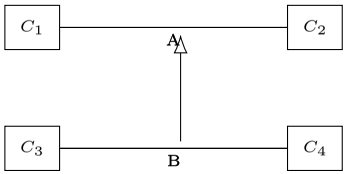
El razonamiento lógico revela una cardinalidad más precisa para la asociación `morigin` del modelo UML. Esta cardinalidad es una consecuencia lógica de la ontología ya que `morigin` es un subconjunto de `origin`, por lo tanto, `morigin` hereda la cardinalidad máxima de `origin`. No sucede lo mismo para la cardinalidad mínima debido a que las relaciones de generalización no son totales, de manera que no todos los elementos de `PhoneCall` están en `MobileCall`. La ontología gráfica resultante es ilustrada en la figura 3.5. Las ontologías en OWL 2 completas para este ejemplo están detallas en el apéndice A, junto con los mapeos implementados en nuestra herramienta.

Para finalizar, vamos a detallar cómo los mapeos y el razonamiento lógico son utilizados para la manipulación de cardinalidades. El algoritmo subya-

Primitiva	UML	$\mathcal{ALCQI}$	Θ_{UML}^{OWL2}	Ψ_{OWL2}^{UML}

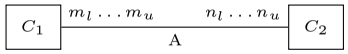
Tipo de Objeto		C	`<Declaration>`
			`<Class IRI="#PhoneCall"/>`
			`</Declaration>`
			`<Declaration>`
			`<Class IRI="#MobileCall"/>`
			`</Declaration>`
			`<Declaration>`
			`<Class IRI="#Phone"/>`
			`</Declaration>`
			`<Declaration>`
			`<Class IRI="#Cell"/>`
			`</Declaration>`
				`SELECT ?class`
				`WHERE {Class(?class)}`

Subclasif.		$C_1 \sqsubseteq C$
		$C_2 \sqsubseteq C$
		...
		$C_n \sqsubseteq C$
		$C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n \sqsubseteq C$
			`<SubClassOf>`
			`<Class IRI="#MobileCall"/>`
			`<Class IRI="#PhoneCall"/>`
			`</SubClassOf>`
			`<SubClassOf>`
			`<Class IRI="#Cell"/>`
			`<Class IRI="#Phone"/>`
			`</SubClassOf>`
				`// Subclases directas y estrictas`
				`  para cada Clase`
				`SELECT DISTINCT ?stsub ?stsupclass`
				`WHERE {`
				`  DirectSubClassOf(?stsub,?stsupclass),`
				`  StrictSubClassOf(?stsub,?stsupclass)`
				`}`
				
				`// Subclases directas para cada Clase`
				`SELECT DISTINCT ?directsub ?supclass`
				`WHERE {`
				`  DirectSubClassOf(?directsub,?supclass)`
				`}`

		$B \sqsubseteq A$	`<SubObjectPropertyOf>`
			`<ObjectProperty IRI="#morigin"/>`
			`<ObjectProperty IRI="#origin"/>`
			`</SubObjectPropertyOf>`
				`// Subpropiedades directas y`
				`  estrictas para cada propiedad`
				`SELECT DISTINCT ?subop ?strictsupop`
				`WHERE {`
				`  DirectSubPropertyOf(?subop,?strictsupop),`
				`  StrictSubPropertyOf(?subop,?strictsupop)`
				`}`

Cuadro 3.2: Semántica para las clases y subclasificaciones de UML, basados en [BCD05], junto con mapeos, definidos en el GOMS, para el ejemplo. La columna Θ_{UML}^{OWL2} incluye la representación OWL 2 del modelo UML inicial, mientras que la columna Ψ_{OWL2}^{UML} muestra las consultas para reconstruir el modelo visual a partir del documento OWL 2 luego del ejecutar razonamiento sobre la ontología formal.

Primitiva	UML	$\mathcal{ALCQI}$	Θ_{UML}^{OWL2}	Ψ_{OWL2}^{UML}

Relación binaria sin clase		$\exists A.\top \sqsubseteq C_1$
		$\exists A^-.\top \sqsubseteq C_2$
		
		$C_1 \sqsubseteq (\geq n_l\, A.\top) \sqcap (\leq n_u\, A.\top)$
		$C_2 \sqsubseteq (\geq m_l\, A^-.\top) \sqcap (\leq m_u\, A^-.\top)$
		
		$C_1\_A_{min_1} \equiv C_1 \sqcap (\geq n_1\, A)$
		$C_1\_A_{min_l} \equiv C_1 \sqcap (\geq n_l\, A)$
		$C_1\_A_{max_1} \equiv C_1 \sqcap (\leq n_1\, A)$
		$C_1\_A_{max_u} \equiv C_1 \sqcap (\leq n_u\, A)$
		$C_2\_A_{min_1} \equiv C_2 \sqcap (\geq m_1\, A^-)$
		$C_2\_A_{min_l} \equiv C_2 \sqcap (\geq m_l\, A^-)$
		$C_2\_A_{max_1} \equiv C_2 \sqcap (\leq m_1\, A^-)$
		$C_2\_A_{max_u} \equiv C_2 \sqcap (\leq m_u\, A^-)$
			<SubClassOf>
			<ObjectSomeValueFrom>
			<ObjectProperty IRI="#origin"/>
			<Class abbreviatedIRI="owl:Thing"/>
			</ObjectSomeValuesFrom>
			<Class IRI="#PhoneCall"/>
			</SubClassOf>
			<SubClassOf>
			<ObjectSomeValuesFrom>
			<ObjectInverseOf>
			<ObjectProperty IRI="#origin"/>
			</ObjectInverseOf>
			<Class abbreviatedIRI="owl:Thing"/>
			</ObjectSomeValuesFrom>
			<Class IRI="#Phone"/>
			</SubClassOf>
			<SubClassOf>
			<Class IRI="#PhoneCall"/>
			<ObjectMaxCardinality cardinality="1">
			<ObjectProperty IRI="#origin"/>
			<ObjectMaxCardinality>
			<SubClassOf>
			
			<EquivalentClasses>
			<Class IRI="#PhoneCall_origin_max_1"/>
			<ObjectIntersectionOf>
			<Class IRI="#PhoneCall"/>
			<ObjectMaxCardinality cardinality="1">
			<ObjectProperty IRI="#origin"/>
			</ObjectMaxCardinality>
			</ObjectIntersectionOf>
			</EquivalentClasses>
			<EquivalentClasses>
			<Class IRI="#PhoneCall_origin_max_9"/>
			<ObjectIntersectionOf>
			<Class IRI="#PhoneCall"/>
			<ObjectMaxCardinality cardinality="9">
			<ObjectProperty IRI="#origin"/>
			</ObjectMaxCardinality>
			</ObjectIntersectionOf>
			</EquivalentClasses>
				// Clases
				SELECT ?class
				WHERE {
				Class(?class)
				}
				
				// ObjectProperties
				SELECT ?objectproperty
				WHERE {
				ObjectProperty(?objectproperty)
				}
				
				// Dominios de ObjectProperty
				SELECT DISTINCT ?objectproperty
				?domainop
				WHERE {
				ObjectProperty(?objectproperty),
				Domain(?objectproperty,?domainop)
				}
				
				// Obtener de ObjectProperty
				SELECT DISTINCT ?objectproperty
				?rangeop
				WHERE {
				ObjectProperty(?objectproperty),
				Range(?objectproperty,?rangeop)
				}
				
				// Clases equivalentes
				SELECT DISTINCT ?classeq1 ?classeq
				WHERE {
				EquivalentClass(?classeq1,?classeq)
				},

Cuadro 3.3: Semántica para las asociaciones binarias sin clase de UML, basados en [BCD05], junto con mapeos, definidos en el GOMS, para el ejemplo. La columna Θ_{UML}^{OWL2} incluye la representación OWL 2 de la asociación `origin` y cardinalidades $n, m \geq 1$ y $n, m \leq 9$, mientras que la columna Ψ_{OWL2}^{UML} muestra las consultas para reconstruir el modelo visual a partir del documento OWL 2 luego del ejecutar razonamiento sobre la ontología formal.

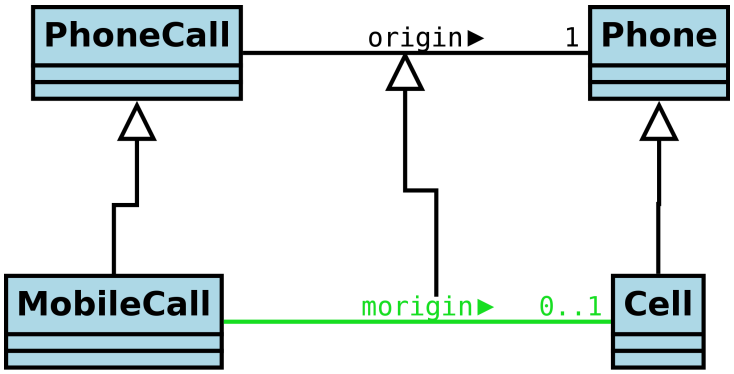


Figura 3.5: Ontología gráfica resultante con nueva cardinalidad que se encontraba implícita en la original.

cente consulta todas las clases equivalentes para las clases involucradas en cada asociación binaria. La figura 3.6 muestra una representación gráfica de las clases asertados a $\mathcal{O}$ durante el mapeo a OWL 2. La traducción Θ_{UML}^{OWL2} genera una clase (definida como un axioma de equivalencia) por cada cardinalidad ≥ 1 y ≤ 9 tanto mínima como máxima. Inicialmente y previo al razonamiento, los nuevas clases asertadas son subclases de las clases asociadas. La definición de cada clase está detalla en las tablas y respectivas.

Luego del razonamiento, el mapeo Ψ_{OWL2}^{UML} consulta todas las clases equivalentes, los cuales implican la cardinalidad actual de las propiedades de la ontología $\mathcal{O}'$. En caso que la cardinalidad actual no esté representada en el modelo visual original (i.e. la clase `MobileCall` se vuelve equivalente a `MobileCall_morigin_1`), se gráfica y se presenta como una consecuencia lógica del modelo. La figura 3.6 muestra una representación la ontología gráfica final y las clases equivalentes que derivan la nueva cardinalidad.

Interoperabilidad de Ontologías Gráficas

El siguiente ejemplo tiene como objetivo ilustrar el uso de *GOMS* para soportar interoperabilidad de modelos gráficos. En este contexto, la interoperabilidad está dada por la posibilidad de representar una ontología gráfica utilizando diferentes lenguajes visuales soportados por el sistema (siempre

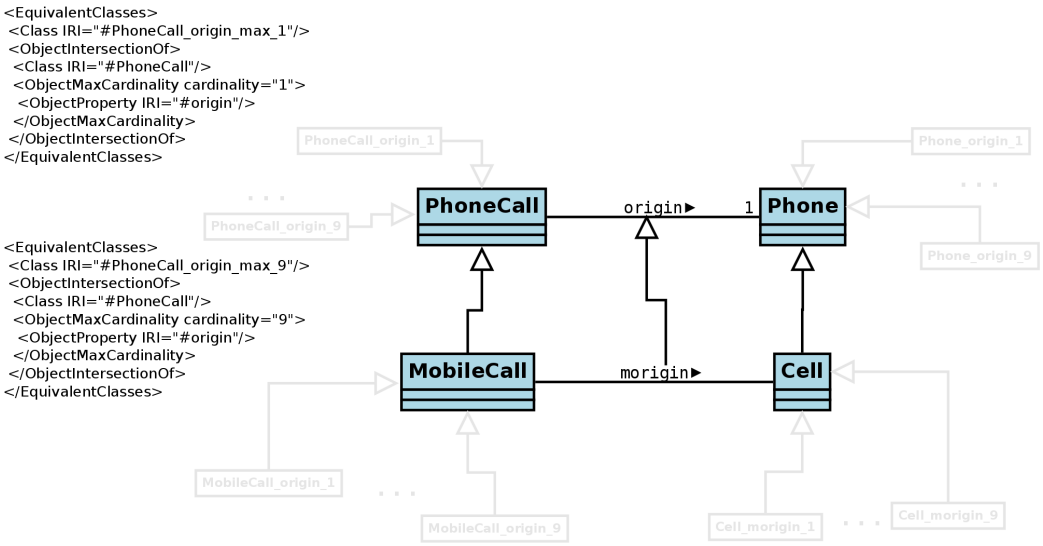


Figura 3.6: Representación UML de las clases asertados durante el mapeo Θ_{UML}^{OWL2}, para manipular las cardinalidades del modelos. Estas nuevas clases asertadas durante el mapeo de la ontología gráfica son transparentes para los modeladores.

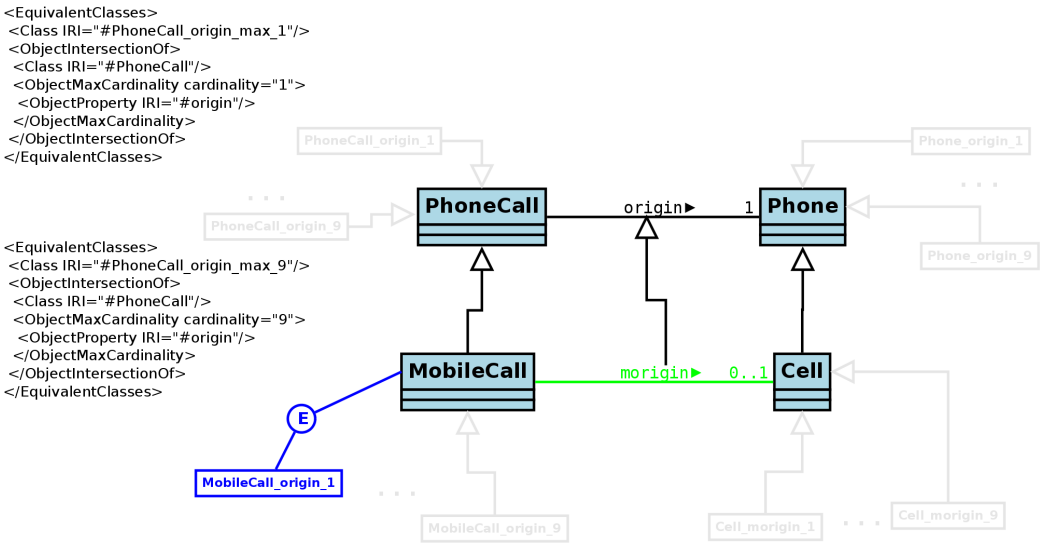


Figura 3.7: Representación UML resaltando el resultado razonamiento lógico. Notar que la clase `MobileCall` se vuelve equivalente a `MobileCall_morigin_1`, lo que implica que existe una cardinalidad máxima de 1 entre `MobileCall` y `Cell`.

que esto se posible) e interrelacionar diversas ontologías gráficas que están representadas en diferentes lenguajes visuales.

El concepto central dentro del *GOMS* que provee las operaciones necesarias para esta interoperabilidad es el metamodel KF [KF15], cuya estructura fue descripta previamente en la sección 3.2. Además de capturar las diferencias y similitudes de los lenguajes de modelado conceptual de datos, el KF define las siguientes reglas de mapeo: mapeo 1:1, transformaciones y aproximaciones. Los primeros son aquellos cuyos elementos de los lenguajes son los equivalentes desde un punto de vista ontológico. Las transformaciones involucran elementos los cuales son esencialmente los mismos, pero no sintácticamente. Las aproximaciones son tipos especiales de mapeos, los cuales requieren de un modelador para validar una conversión o transformación. Por último, debido a que los lenguajes del metamodelo no tiene la misma expresividad, existen primitivas que no pueden ser transformadas ni aproximadas en un lenguaje destino, como por ejemplo las cardinales compuestas de ORM 2 no puede representarse en UML.

Haciendo uso de estos mapeos entre lenguajes de modelado, para este ejemplo definimos una nueva instancia del *GOMS* que abarca subconjuntos de UML, EER y ORM 2 como lenguajes visuales, DL como lenguaje formal y utilizaremos los mismos mapeos gráfico y lógicos del ejemplo anterior. Entonces definimos los siguientes escenarios de interoperabilidad. En todos estos casos, cada ontología gráfica en un *GOMS* es una instancia del KF, de manera que ellas son convertidas a sus respectivas representaciones en el metamodelo para capturar las diferencias de expresividad entre lenguajes.

1. **Diagrama de Clases UML $\rightarrow$ DL $\rightarrow$ EER**. Este escenario ilustra la conversión de una ontología gráfica en UML, cómo ésta es mapeada a una base de conocimiento en DL y, luego de ejecutar razonamiento lógico sobre esta base, la ontología gráfica es reconstruida en EER. La primera y segunda fila de la tabla 3.4 muestra este escenario con ejemplos simples.

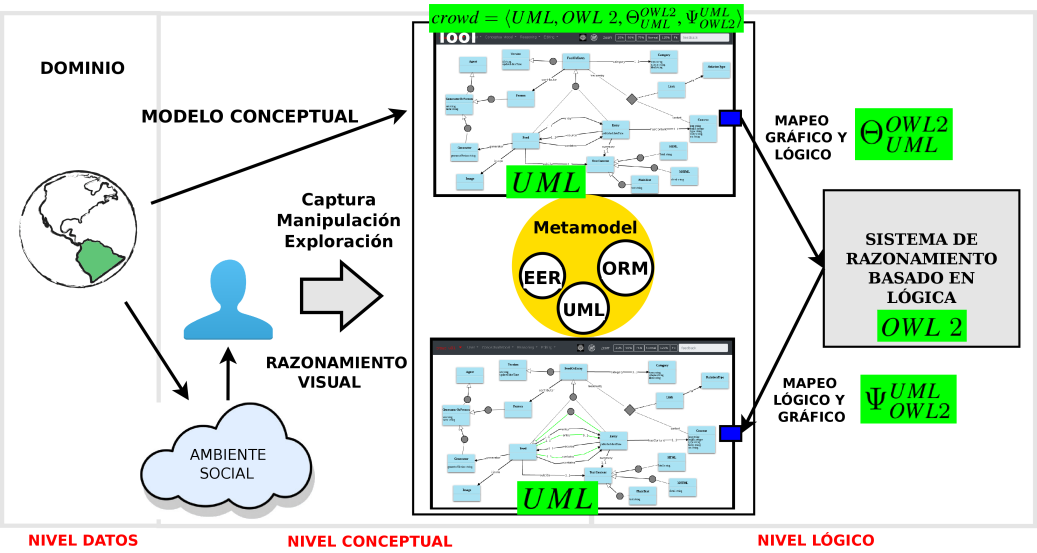
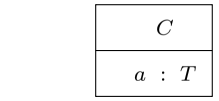
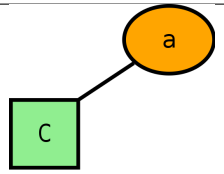
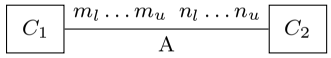


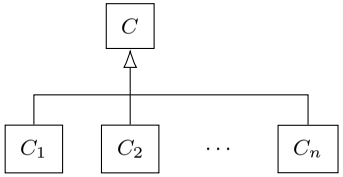
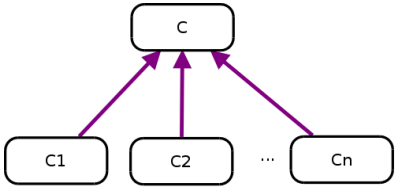
Figura 3.8: Vista Esquemática del Proceso de Visualización para Ontologías indicando los componentes de un GOMS.

2. **Diagrama de Clases UML → ORM 2**. El segundo escenario ilustra la transformación entre una ontología UML a una en ORM 2. La ontología resultante es nuevamente expresada en un respectivo perfil DL para razonamiento y luego es reconstruido en el mismo lenguaje visual con las posibles nuevas consecuencias derivadas del razonamiento. La tercer fila de la tabla 3.4 muestra una simple transformación para una generalización UML.

3.4. Conclusiones

En este capítulo se definió el concepto de proceso de visualización de conocimiento basado en ontologías, cuyo objetivo principal es proveer modelos visuales efectivos con una semántica bien definida para validarlos, mejorarlos y comunicarlos de manera precisa. La motivación está centrada en unificar criterios de usuarios finales y expertos de dominios junto su entendimiento sobre los dominios, además de las herramientas gráficas.

UML	DL	EER	Metamodel
	$\exists a.\top \sqsubseteq C$ $\exists a^{-}.\top \sqsubseteq T$ $C \sqsubseteq (\leq 1\, a.\top)$		Object type: [C], Subsumption:[], Relationship:[], Object type cardinality:[], Attribute: [C, a, T]
	$\exists A.\top \sqsubseteq C_1 \;\; \exists A^{-}.\top \sqsubseteq C_2$ $C_1 \sqsubseteq (\geq n_l\, A.\top) \sqcap (\leq n_u\, A.\top)$ $C_2 \sqsubseteq (\geq m_l\, A^{-}.\top) \sqcap (\leq m_u\, A^{-}.\top)$ $C_1\_A_{min_l} \equiv C_1 \sqcap (\geq n_l\, A)$ $C_1\_A_{max_u} \equiv C_1 \sqcap (\leq n_u\, A)$ $C_2\_A_{min_l} \equiv C_2 \sqcap (\geq m_l\, A^{-})$ $C_2\_A_{max_u} \equiv C_2 \sqcap (\leq m_u\, A^{-})$		Object type: [C1,C2], Subsumption:[], Relationship: [r1,[C1,C2]], Object type cardinality: [r1, [c1,nl,nu], [c2,ml,mu]], Attribute:[]

UML	ORM2	DL	Metamodel
		$C_1 \sqsubseteq C \;\; C_2 \sqsubseteq C \;\; \ldots \;\; C_n \sqsubseteq C$ $C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n \sqsubseteq C$	Object type: [C,C1,C2,Cn], Subsumption: [r1,C, [C1,C2,Cn]], Relationship:[], Object type cardinality:[], Attribute:[]

Cuadro 3.4: Escenario 1: primera y segunda fila. Ambos muestran primitivas UML visualizadas en EER a partir de una semántica precisa en DL. El tipo de dato para el atributo es mantenido en la representación del metamodelo. Escenario 2: la tercer fila muestra una conversión directa entre UML y ORM 2 manteniendo ambos la semántica en DL y la representación en el metamodelo.

Se describieron sus componentes principales del proceso y se introdujeron los conceptos de ontologías gráficas y cómo estas pueden ser derivadas desde un sistema para manipulación de ontologías gráficas (*GOMS*), a partir de transformaciones gráficas y lógicas. Asimismo, se definió un conjunto de servicios de razonamiento lógicos basados sobre estos mapeos, junto con su correspondencia en un lenguaje visual. Concluimos nuestra definición con un ejemplo concreto a partir de una ontología gráfica en UML, donde detallamos los mapeos utilizados para su manipulación lógica y su posterior reconstrucción. También, mostramos cómo los mapeos pueden ser extendidos para aumentar el poder de inferencia de un sistema, mediante la definición de nuevas clases y así utilizar el razonamiento para obtener cardinalidades implícitas en una ontología.

A partir de esta formalización es posible definir el poder expresivo (gráfico y lógico) de una herramienta visual con soporte de razonamiento automático. Asimismo, dotarlas de dos funcionalidades complementarias: la **visualización de ontologías**, para manipularlas, extenderlas y comprenderlas en el contexto de la ingeniería ontológica; y la **validación de modelos conceptuales** en procesos relacionados a la ingeniería de software. La figura 3.8 muestra el proceso de visualización final junto con las definiciones centrales del *GOMS*.

En el siguiente capítulo nos introducimos en los aspectos de diseño para tal ambiente, mediante una descripción detallada de una arquitectura de software de referencia.

Capítulo 4

Arquitectura de Referencia para Ambientes Web de Ingeniería Ontológica

Contents

La ingeniería ontológica es un conjunto de actividades, metodologías, lenguajes y herramientas relacionadas al desarrollo, mantenimiento, refinamiento, evaluación y uso de ontologías. El desarrollo de estos modelos es clave en este contexto y requiere ser soportada por metodologías apropiadas que guíen a los modeladores a través de este proceso. Actualmente, todas estas actividades aparecen fragmentadas en varias herramientas o sistemas adicionando un ingrediente más a la complejidad ya comentada, afectando también la calidad de los modelos. La literatura revela que si bien un gran número de ambientes han sido desarrollados, sólo algunos pocos están disponibles de manera web y además en casi todos los casos, ellos son meros visualizadores

con limitadas funcionalidades gráficas para edición e interacción con razonadores lógicos. Asimismo, algunos autores [VBJS14] concluyen taxativamente, que no existe ningún framework ampliamente aceptado para tareas básicas de ingeniería debido a la falta de entendimiento de la efectividad de los mismos.

En este sentido, nuevas arquitecturas para ambiente de ingeniería ontológica, deben ser diseñadas y documentadas para cubrir las falencias actuales, proveyendo funcionalidades básicas e interfaces que permitan su extensión. En el presente capítulo, se presenta y documenta un arquitectura de referencia para un ambiente de ingeniería ontológica web y gráfico.

La documentación del software es importante para definir y evaluar atributos de calidad, identificar partes donde el sistema puede ser extendido y comunicar lineamientos y decisión tomadas para que otros puedan usarla apropiadamente. Por lo tanto, durante el desarrollo de este capítulo, detallaremos tres vistas arquitectónicas: módulos, componentes y conectores, y despliegue, las cuales serán documentadas siguiendo los fundamentos presentados en [GBI+10, Gom11].

La arquitectura aquí descripta y documentada ha sido publicada en [BEF18] e incorpora los requerimientos hasta aquí identificados, junto con los siguientes, para favorecer el desarrollo modular de herramientas de manipulación de ontologías gráficas:

- ***(R3)*** *Accesible vía Web*: Los usuarios necesitan herramientas que puedan acceder en línea y a versiones actualizadas de dichos ambientes. Asimismo, los entornos Web favorecen el desarrollo de tareas colaborativas [FLGPJ97, TNNM13].
- ***(R5)*** *Framework Extensible*: Herramientas construidas utilizando Interfaces de Programación (APIs) y patrones de diseño establecidos permiten incorporar nuevas funcionalidades de manera flexible y rápida [KFNM04, GBI+10].

Finalmente, si bien la arquitectura es lo suficientemente genérica para la

implementación de otras instancias, nos referiremos a nuestra herramienta *crowd* para explicar sus detalles.

4.1. Vistas Arquitectónicas

En esta sección detallamos las vistas referidas previamente. Cada una de ellas resalta importantes aspectos de implementación y las decisiones de diseño tomadas. En casa vista, se explicará su motivación, su intención y se describirán sus principales elementos. En todos los casos, se utilizará notación del estándar UML. Para facilitar la comprensión del texto, asumiremos que *módulos* son unidades de implementación del software junto con un conjunto de responsabilidades. Asimismo, *componentes* representan elementos tales como objetos, clientes y servidores y, finalmente, *artefactos* denotan archivos y paquetes software para ser instalados y desplegados en un ambiente en ejecución.

4.1.1. Vista de Módulos

Motivación

El sistema es descompuesto en unidades de implementación o módulos, los cuales pueden ser relacionados como *es-parte-de*, *depende-de* y *es-un*. Cada módulo provee un conjunto de responsabilidades que definen su rol en cuando a lograr un determinada funcionalidad. Esta vista detalla cómo estas unidades son ensambladas para formar un estructura más compleja y su objetivo principal es proveer lineamientos iniciales para el código fuente.

Estilos

Descomposición, Usos y Capas.

Descripción

La vista modular de *crowd* es una combinación consistente de los estilos previamente mencionados. La arquitectura se divide en dos capas, donde cada un provee un conjuntos de servicios a la restante.

Como describe la Figura 4.1, la capa superior agrupa un conjunto de módulos relacionados a la interfaz de usuario. Al mismo tiempo, es descompuesta en unidades que describen una relación de inclusión y dependencia entre ellos. El módulo principal es el llamado `gui`, el cual depende de las `views` y los `models`. `views` implementa todos los *templates* y sus respectivos manejadores de eventos, mientras que `models` gestiona los modelos gráficos, sus primitivas y la interacción con la librería gráfica por intermedio de respectivas factorías UML, EER y ORM, provistas por un *Abstract Factory pattern* [GHJV95].

Dos módulos específicos: `model.uris` and `views.common` implementan características que son ortogonales a los modelos de un determinando lenguajes gráfico, como el tratamiento de espacios de nombres[1] y *templates* y eventos comunes de la interfaz de usuario, respectivamente. Finalmente, el manejo de eventos para toda la interfaz es implementado a través de un *State pattern*, quién captura los estados actuales de la interfaz y altera su conducta en base estados sucesores.

La capa inferior contiene los módulos requeridos para el procesamiento de modelos ontológicos: traductores y manipuladores de documentos, gestión de usuarios e interfaces con herramientas externas. El módulo principal de la capa es el `translator`, el cual tiene un rol doble de orquestar tanto la codificación de modelos gráficos en una base de conocimiento lógica y procesar las ontologías extraídas desde documentos OWL usando `strategies` y `builders` específicos.

El módulo `strategies` implementa un *Strategy pattern*, que define una familia de algoritmos que dependen de los lenguajes visuales y como ellos pueden reconstruidos usando lógica, mientras que el módulo `builders` da

[1] *namespaces*

la estructura de los documentos OWL generados a partir de la representación formal. El `translator` también incluye paquetes de QA para procesar las consultas a los razonadores. La interacción con herramientas externas es implementada a través de interfaces en el módulo `reasoner`.

La decodificación de ontologías está a cargo del paquete `decoder`, a cargo de procesar documentos OWL por medio de consultas SPARQL-DL. Un motor SPARQL-DL retorna la estructura de la actual ontología para ser importada en *crowd*. Este módulo implementa también funcionalidades para gestión de URIs y depende de una codificación particular provista por `strategies` y estructurada en `builders`. Conjuntamente, todos estos módulos son orquestados en un módulo `common`, el cual es responsable de implementar y ofrecer los servicios de inferencia. La Figura 4.2 ilustra la interacción entre objetos en un escenario de importación de OWL 2.

Figura 4.1: Vista de Módulos UML.

4.1.2. Vista de Componentes y Conectores

Motivación

Este tipo de vistas complementa la detallada anteriormente, mostrando elementos o componentes que tiene presencia en tiempos de ejecución. Dichos componentes representan objetos y como estos se comunican por medio de conectores (enlaces, protocolos). Conectores definen caminos de interacción entre los componentes, los cuales poseen un conjunto de puertos para conectarlos. El esquema resultante es un grafo que permite visualizar cómo funciona el sistema descripto.

Estilos

Llamada-Retorno.

Descripción

Desde su concepción, *crowd* fue pensado como una implementación cliente-servidor. En este contexto, un cliente provee una interfaz gráfica amigable para el usuario para modelado conceptual. Dicho cliente es interpretado por un navegador web y requiere de los servicios de un servidor, actualmente siendo ejecutado en un servidor web remoto. La herramienta sigue estos mismos principios, permitiendo a los clientes resolver peticiones usando el protocolo HTTP, el cual es una forma de invocación requerimiento/réplica *(request/-reply)*.

La figura 4.3 describe la vista de Componentes y Conectores (C&C) para el cliente, sus componentes y un esquema de comunicación entre ellos. *crowd* soporta *widgets* y eventos para múltiples lenguajes estándar. Ellos pueden ser accedidos por medio del componente `GUI` que gestiona instances de `GUIIMPL`. Cada una de estas instancias es la representación de una interfaz específica, sus eventos, *widgets* y diagramas.

Un componente `Adapter`, implementa el patrón estructural *Adapter* [GHJV95],

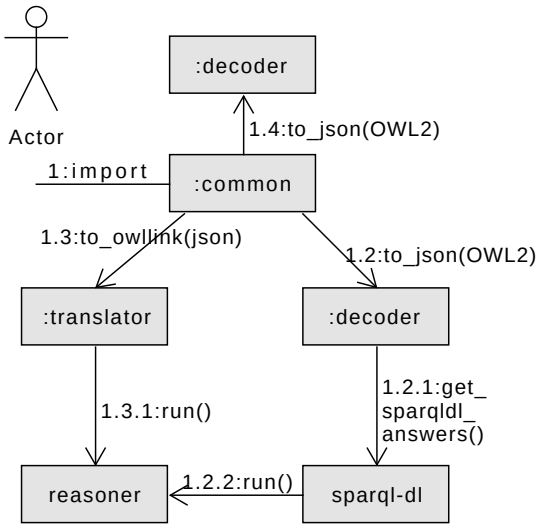


Figura 4.2: Diagrama de Comunicación UML para la importación de ontologías OWL 2. Las flechas indican nombre y dirección de la transmisión de mensajes entre objetos y los números representan la secuencias de mensajes.

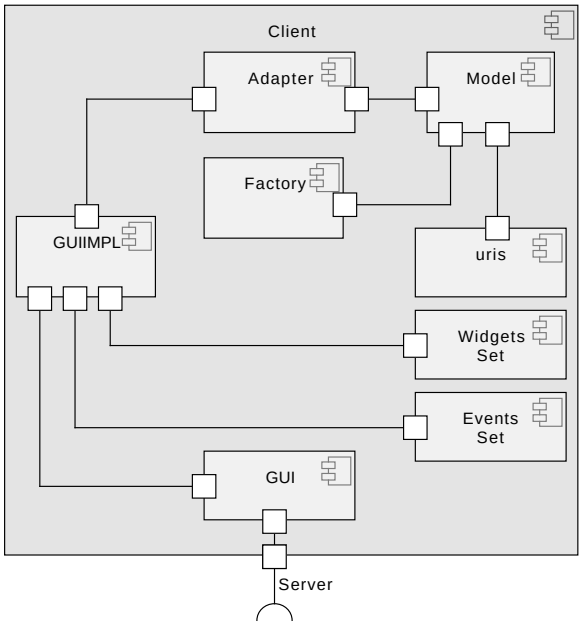


Figura 4.3: Vista de Componentes y Conectores del Cliente.

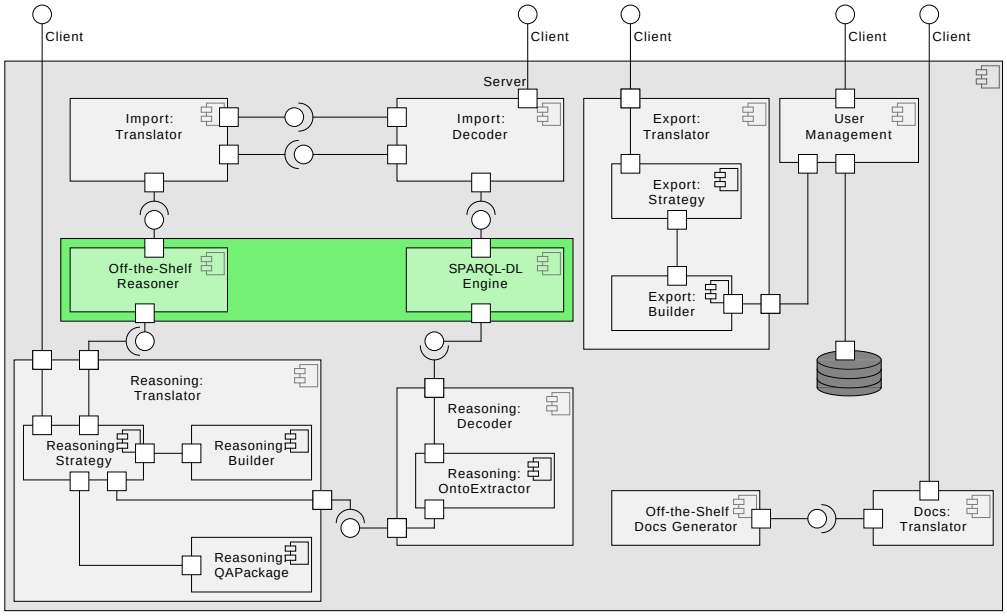


Figura 4.4: Vista de Componentes y Conectores del Servidor. Círculos superiores indican interfaces disponibles al cliente.

provee una comunicación entre un interfaz particular y un diagrama (o `Model`). Por ejemplo, para un diagrama UML y su interfaz gráfica. El componente `Widget Set` gestiona todos los componentes gráficos necesarios para componer un diagrama y también provee componentes para gestionar características comunes a todos los lenguajes soportados. Por último, un componente `Factory`, implementa un *Abstract Factory pattern* [GHJV95], define una interfaz para crear los elementos gráficos (o primitivas) de los lenguajes gráficos soportados por la herramienta. Es componente, delega esta tarea la librería gráfica JointJS [2].

Asimismo, los ontologías gráficas comparten componentes comunes tales como `uris`, para el tratamiento y validación de espacios de nombres. Una URI es un identificador único para cada primitiva de las ontologías. Por lo tanto, cada primitiva es registrada por medio de un nombre (gráfico), un prefijo y una URL. El tratamiento de URIs en ambientes de ingeniería ontológica

[2] `https://www.jointjs.com/opensource`

es esencial para la publicación y reuso, clave para la concreción de la Web Semántica [Miz04].

La figura 4.4 esquematiza la distribución de componentes en el servidor junto con sus sub-componentes y conectores. Para simplificar el diagrama, sólo los componentes `Reasoning` son expandidos completamente, mientras que detalles de otros componentes tales como `Import` son obviados ya que presentan vistas C&C similares.

La sección resaltada en el esquema (en verde) marca la importancia de los sistemas de razonamiento lógico en proveer soporte para los servicios de importación de ontologías OWL y de razonamiento sobre modelos visuales. En todos los casos, estas herramientas proveen interfaces de comunicación o protocolos tales como la OWL API [HB11] y el protocolo OWLlink [LLNW11]. En este sentido, los componentes `Traslator` and `Decoder` conectan a motores externos. En particular, el componente `OntoExtractor` representa objetos en ejecución implementando las responsabilidades definidas para los módulos `decoder` y `sparql-dl`. Además, está a cargo del procesamiento de los resultados de las consultas SPARQL-DL, el cual permite la extracción de axiomas intencionales (TBox) y extensionales (ABox) a partir de ontologías OWL 2. Tales consultas están definidas en la especificación del lenguaje SPARQL-DL language[3] (ver figura. 4.5), y son procesadas por una motor de consultas[4] construido a partir de razonadores lógicos.

[3] `http://derivo.de/en/resources/sparql-dl-api/sparql-dl-syntax/`
[4] `http://derivo.de/en/resources/sparql-dl-api/`

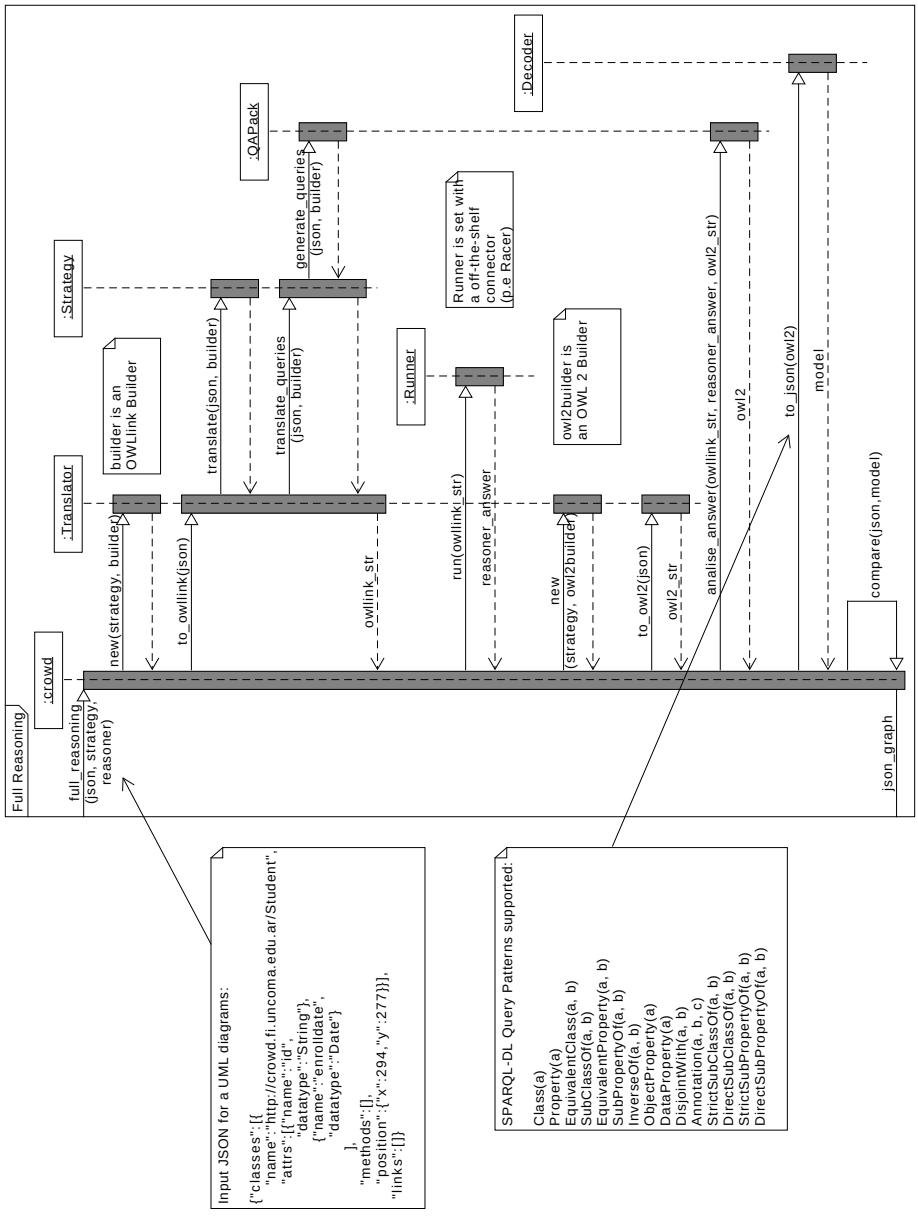


Figura 4.5: Diagrama de Secuencia UML para servicios de razonamiento.

Cuadro 4.1: Mapeo entre las vista C&C y de módulos para cliente y servidor. (*) denota módulos que son partes de más de un componente en tiempo de ejecución.

C&C View	**Module View**
Model, Factory, Uris	model
GUI, GUIIMPL, Adapter	gui
Widgets Set, Events Set	views
*:Translator	common, translator
*:Decoder	common, decoder
OntoExtractor	decoder
Strategy	strategies
*:Builder	builder
QAPackage	qapackages
Off-the-shelf Docs Generator	widoco
Off-the-shelf Reasoner, SPARQL-DL engine	reasoner
User Management	users

En el mismo sentido, los componentes `Strategy`, `Builder` y `QAPackage` se comunican entre sí e implementan las responsabilidades asociadas al módulo `translator`. `Translator` también conecta a sistemas externos a través del protocolo OWLlink, enviando una ontología OWL en un archivo para validar su consistencia. La salida del razonamiento es post-procesada por el componente `QAPackage`.

Finalmente, la nueva ontología OWL es enviada a una instancia del componente `Decoder` para luego ser comparado contra la ontología gráfica original. En este sentido, la decisión de integrar dos estrategias diferentes para consultar la ontología (OWLlink y SPARQL-DL) es una solución pragmática y es fundamentada en requerimientos de interoperabilidad: el uso de protocolos estándar como OWLlink para interconectar razonadores, y el soporte para la importación de ontologías OWL, la cual requiere de una mayor expresividad de los lenguajes de consulta para extraer más conocimiento desde estos documentos. La figura 4.5 presenta un diagrama de secuencia describiendo una funcionalidad completa, complementando la vista C&C y resaltando la

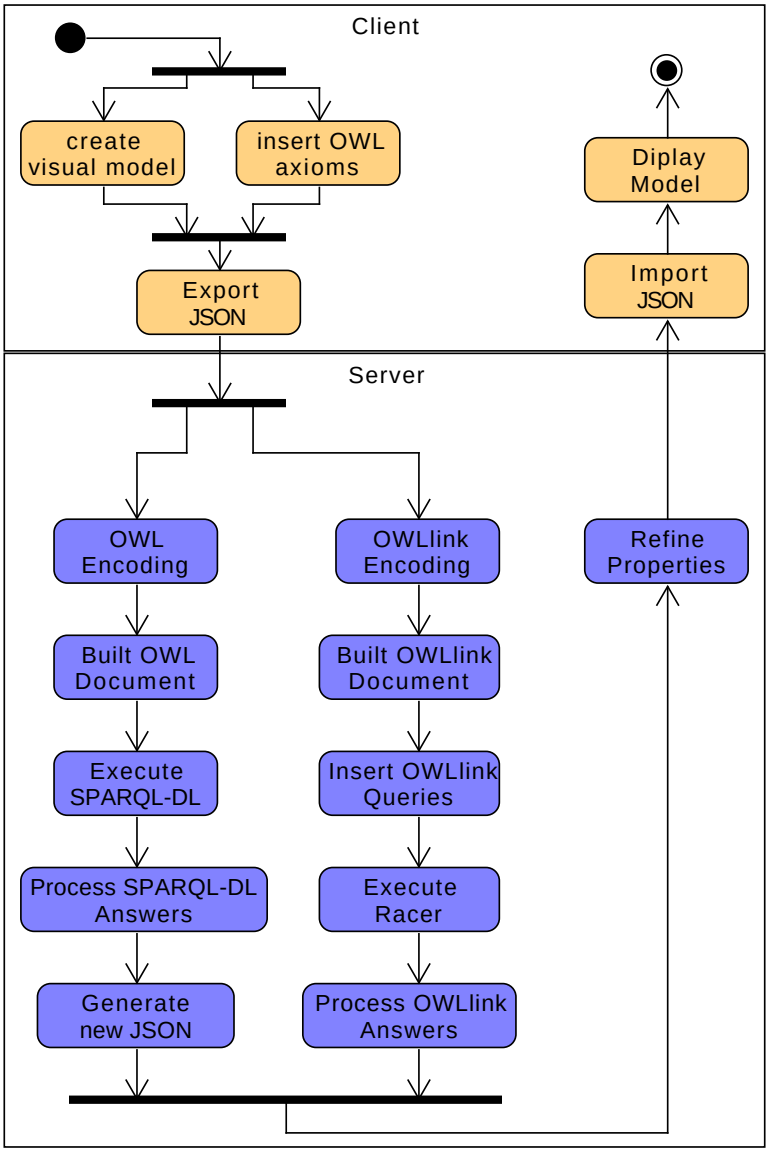


Figura 4.6: Diagrama de Actividades UML mostrando la concurrencia cuando los servicios de razonamiento son requeridos.

interacción en tiempos de ejecución. Asimismo, la figura 4.6 ilustra la misma interacción a través de un diagrama UML de actividades.

Para concluir la descripción de esta vista, la tabla 4.1 ilustra el mapeo entre componentes en C&C y la vista de Módulos. El mapeo muestra cómo los módulos en la vista de Módulos contribuyen a la implementación de los componentes en la vista C&C.

4.1.3. Vista de Asignación

Motivación

La vista de asignación presenta una correspondencia entre elementos software y hardware. En particular, la vista de despliegue describe la configuración física de la arquitectura y cómo los componentes de la misma son asignados a nodos físicos. Por otro lado, la vista de instalación mapea componentes de la arquitectura a estructuras en el sistema de archivos del ambiente de ejecución; mientras que el estilo de asignación de trabajo muestra los equipos responsables del desarrollo de cada uno de los módulos del software.

Estilos

Despliegue e Instalación.

Descripción

La arquitectura presentada es centralizada, por lo tanto, todos sus archivos PHP son desplegados en un sólo servidor HTTP, el cual ejecuta un servicio Apache[5] y un MySQL[6]. Herramientas externas y módulos, archivos ".*jar", son también instalados en el mismo servidor y corresponden al componente `Reasoner` de la vista detallada en la sección anterior. Actualmente, el servidor es un x86 Intel 3GHz, ejecutando Apache v2.4.10, MySQL v5.6.4 en un sistema Debian GNU/Linux 8 jessie. Además, PHP v7.0 y Java RE

[5] `https://httpd.apache.org/`
[6] `https://www.mysql.com/`

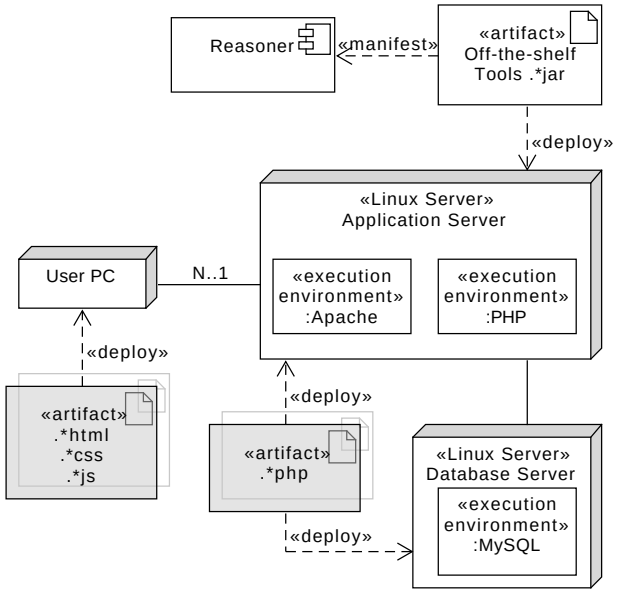


Figura 4.7: Diagrama de Despliegue UML y Esquema de Instalación para la arquitectura.

v1.8.0 son también desplegados en dicho nodo. Finalmente, la multiplicidad $N.,1$ indica el número de instancias de un cliente en cada comunicación. Tales clientes representan navegadores web, los cuales interpretan los archivos ".*html", ".*css" and ".*js" de la interfaz de usuario. La figura 4.7 muestra un diagrama de despliegue e instalación detallado.

4.2. Conclusiones

Durante el desarrollo del presente capítulo, presentamos la documentación de una arquitectura de referencia para un ambiente web de ingeniería ontológica. Para esto, describimos tres vistas complementarias y, a su vez, combinando estilos para cada una: vista de módulos (descomposición, capaz y usos); componentes y conectores (llamada-retorno) y asignación (despliegue e instalación).

La arquitectura referida cumple con las motivaciones de nuestro trabajo, en particular con los requerimientos identificados, y da las bases para el desarrollo de novedosas arquitecturas colaborativas. Los primeros pasos de esta implementación fueron reportados en [GBCF16, BGFC17].

También fue utilizada para los desarrollos presentados en [GBCF18, OBCF18] demostrando su flexibilidad y extensibilidad, no solo para el manejo de ontologías gráficas sino también para la definición de un sistema de consultas visuales para el lenguaje de consultas SPARQL-DL y de un sistema para modelado y verificación de patrones en modelos de variabilidad (OVM) para lineas de productos software.

Por último, el código de la arquitectura es código abierto y está disponible en `https://bitbucket.org/gilia/wicom-qdod/src`.

Capítulo 5

crowd v1.0 y Evaluación

Contents

En este capítulo, se presenta la primer versión de *crowd* que se encuentra en linea a partir de Noviembre de 2018 en `http://crowd.fi.uncoma.edu.ar` y que fue implementada a partir de un sistema para manipulación de ontologías gráficas y la arquitectura de referencia presentada previamente. Para esto, se detalla su alcance funcional y se describe el editor UML que la herramienta provee.

Por otro lado, hemos conducido estudios para medir la usabilidad por medio de experimentos con un grupo de alumnos. Asimismo, formalizamos a *crowd* como una instancia de un modelo *WYSIWYM* (*What You See Is*

What You Mean) [KA15]. Esta evaluación nos permite concluir sobre la usabilidad actual de la herramienta y también sobre la relación entre los modelos semánticos generados y los respectivos elementos gráficos para tareas de visualización, exploración y desarrollo de ontologías. Finalmente, se detallan sendos estudios y se concluye sobre los resultados de los mismos.

5.1. *crowd* v1.0

Presentamos la herramienta *crowd* como una instanciación de un *GOMS*, como definido en el capítulo 3. En todos los casos, mediante un abuso de notación denotamos a la representación gráfica en un *GOMS* por el lenguaje mismo (UML, EER, ORM2), aunque estos lenguajes provean primitivas que no es posible representar en una herramienta, tal el caso de las asociaciones n-arias de UML. Por lo tanto, la representación de diagramas de clases UML será denotada como UML. De manera análoga, utilizamos EER y $ORM2$, así como también la notación OWL 2 para referirnos al lenguaje formal subyacente de un *crowd*.

Definition 1 `crowd v1.0` *es una instancia de un* GOMS*:*

$$crowd = \langle UML, OWL\ 2, \Theta_{UML}^{OWL2}, \Psi_{OWL2}^{UML} \rangle$$

donde:

- $\Theta_{UML}^{OWL2} \ : \ UML \rightarrow OWL\ 2$
- $\Psi_{OWL2}^{UML} \ : \ OWL\ 2 \rightarrow \ UML$

En *crowd*, los modelos visuales son lógicamente reconstruidos en $\mathcal{ALCQI}$ por medio de un conjunto de mapeos, basados en [BCD05] y escrito en OWL 2. Como consecuencia, el razonamiento sobre las ontologías gráficas UML en

crowd es EXPTIME-complete. Por otro lado, una ontologías OWL 2 procesada por los razonadores es mapeada a un modelo visual, modificando el original. Esta caracterización de *crowd* como un *GOMS* trasparenta la semántica total y precisa de nuestra herramienta a través de los mapeos definidos presentados en el apéndice A.

Bajo este concepto, los siguientes nuevos mapeos pueden ser incorporados a *crowd* para EER [ACK+07] y ORM 2 [FMS12], para los cuales nos encontramos desarrollando las respectivas interfaces de usuario (en tiempos de presentación de este trabajo).

$$\Theta_{EER}^{OWL2} \ : \ EER \rightarrow OWL\ 2$$

$$\Psi_{OWL2}^{EER} \ : \ OWL\ 2 \rightarrow \ EER$$

$$\Theta_{ORM2}^{OWL2} \ : \ ORM\ 2 \rightarrow OWL\ 2$$

$$\Psi_{OWL2}^{ORM2} \ : \ OWL\ 2 \rightarrow \ ORM\ 2$$

Habiendo precisado la semántica y el poder expresivo de *crowd* como un sistema para la manipulación de ontologías gráficas, estamos en condiciones de presentar su descripción funcional y su editor UML.

5.2. Descripción Funcional

Un proyecto en *crowd* consiste de diagramas o modelos representando ontologías OWL 2. Para modelado gráfico, la herramienta utiliza los lenguajes de modelado conceptual estándar: EER, UML y ORM 2. Sin embargo, el modelo gráfico subyacente está basado sobre un núcleo común de estos lenguajes a través del metamodel KF. De esta manera, cada modelo gráfico en

la herramienta es un instancia de dicho metamodelo. Esta implementación permitirá a *crowd* la capacidad de enlazar, integrar y/o convertir modelos representados en diferentes lenguajes. Durante esta especificación funcional utilizaremos la terminología propuesta en el KF.

Las primitivas básicas de *crowd* son los tipos de objetos (clases, entidades), *object types*, los cuales denotan conjuntos de objetos llamados instancias, que comparten propiedades comunes; y relaciones (asociaciones, relaciones), *relationships*, que denotan un conjunto de tuplas partir de instancias de las diferentes entidades involucradas en dicha relación. Debido a que la misma entidad puede estar involucrada en la misma relación más de una vez, la participación de esas entidades es representada por medio de roles a los cuales se les asigna un único nombre. Ambas primitivas pueden tener atributos, es decir, propiedades cuyos valores pertenecen a un dominio predefinido como por ejemplo cadenas (*string*) o enteros *integer*, entre otros.

Los roles denotan la conexión de un tipo de objeto a una relación y son usados para expresar también las restricciones de cardinalidad de tales objetos y relación. Un rol puede tener dos restricciones: total, o mínima cardinalidad y unicidad, representando la máxima cardinalidad. En la versión actual de *crowd*, los valores para expresar estas cardinalidades están restringidos entre 0 a 9, y la máxima es indicada con "*". Una mínima cardinalidad de 1 indica que todas las instancias de una clase deben participar en la relación al menos una vez (restricción mandatoria). Una máxima cardinalidad de 1 indica que todas las instancias de una clase pueden sólo participar una sola vez en la relación (restricción funcional).

Subsumciones (generalizaciones, isa), *subsumptions*, entre tipos de objetos es un tipo especial de relación. Ellas establecen aserciones de inclusión entre tipos de objetos y entre relaciones, con la posibilidad de especificar restricciones totales y disjuntas. La primera restricción expresa el hecho que un tipo de objeto es equivalente a la unión de los descendientes; mientras que la segunda establece que tipos de objetos son mutuamente disjuntos. En ausencia de una restricción específica, tipos de objetos pueden estar solapados.

Cada modelo puede tener espacios de nombres asociados, incluyendo los estándares `rdf, rdfs, xsd, owl, xml` y los definidos por el usuario, y que son considerados para la definición de cada componente visual. Esta funcionalidad es clave para el desarrollo de la Web Semántica y de Linked Data debido a que habilita la gestión de URIs para la integración de diversos vocabularios en un mismo modelo visual, cumpliendo con las buenas prácticas para publicar y documentar ontologías.

Los modelos visuales de *crowd* son codificados como bases de conocimiento lógico y enviados a un razonador para revisar y mejorar su calidad. El razonamiento completo sobre lenguajes de ontologías da soporte a los ingenieros de ontologías para tareas de creación y mantenimiento de modelos, visibilizando posibles restricciones implícitas durante el proceso.

Cuando los servicios de razonamiento son invocados desde la interfaz gráfica, el modelo visual es serializado en una representación OWLlink, i.e. una versión XML de la codificación DL del modelo, y enviado a un razonador externo conectado a *crowd*. Un conjunto de consultas sobre los conceptos, propiedades y el estado de la base de conocimiento son adicionados a dicha representación OWLlink. De manera concurrente, una representación OWL 2 del mismo modelo es también generado y enviado a un motor de consultas SPARQL-DL para un reconstrucción gráfica de una ontología.

Cada tipo de objeto, atributo y relación en el modelo es validado por satisfacibilidad. En particular, para tipos de objetos y relaciones, se determinan sus equivalentes, subclases y superclases. Para la validación de restricciones de cardinalidad, el sistema consulta cada rol por consistencia de sus clases y relaciones y luego sobre la subsumción de nuevos conceptos que representan las restricciones mandatorias y funcionales.

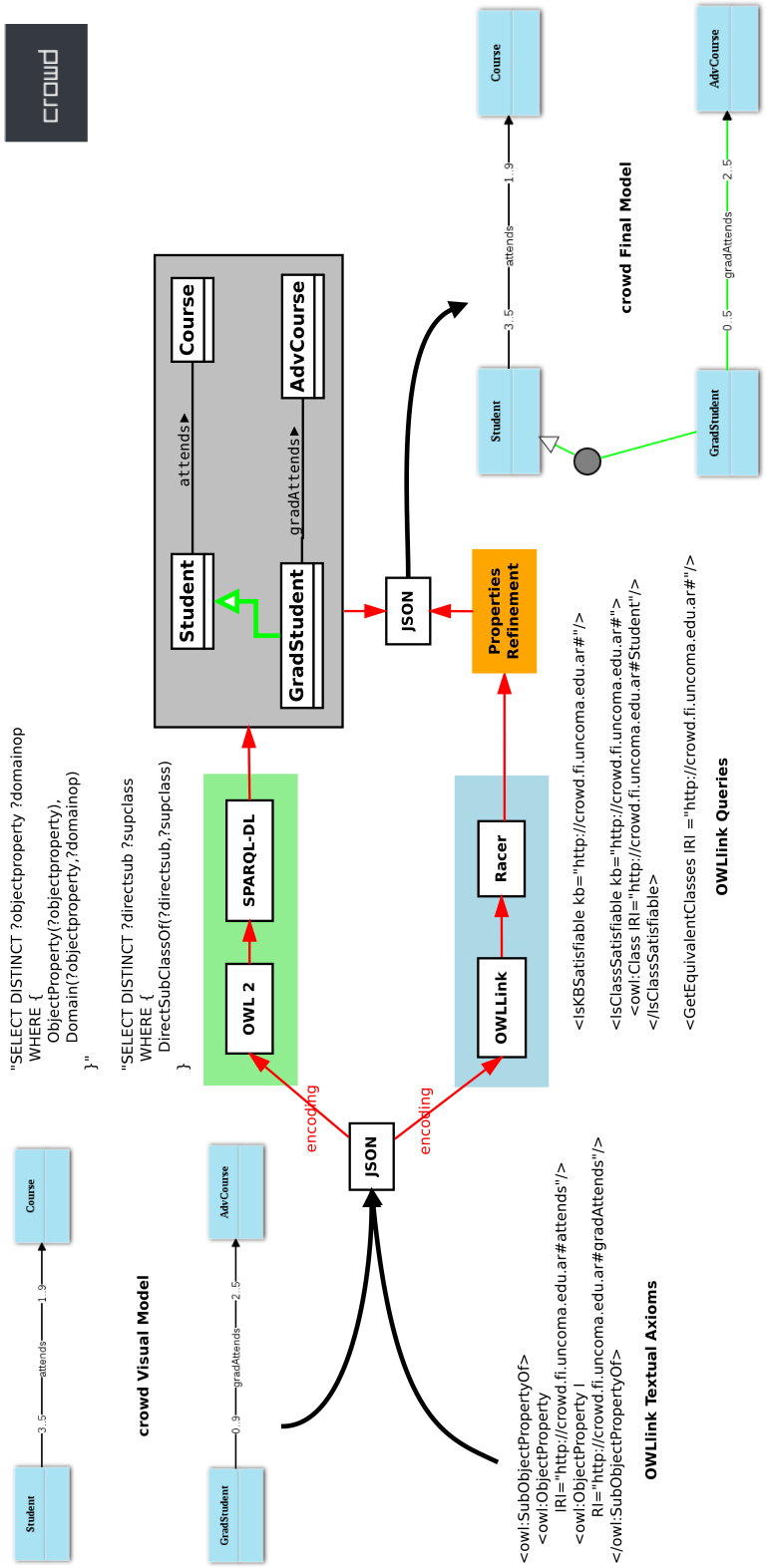


Figura 5.1: Esquema funcional de *crowd* respecto a la arquitectura de referencia definida en el capítulo anterior.

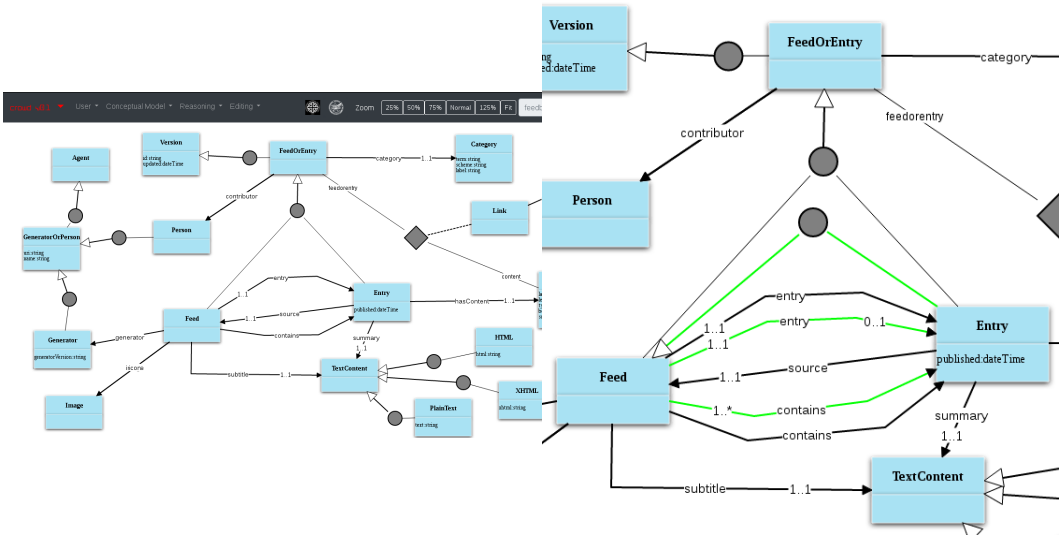


Figura 5.2: (izq.) Modelo inicial en *crowd* para ontología `AtomOwl`. (der.) Zoom sobre nuevas conclusiones luego del razonamiento.

Luego de la finalización del razonamiento automático, *crowd* provee al usuario un nuevo diagrama con las posibles deducciones gráficas, modificando el modelo original. Cada tipo de objeto insatisfacible, describiendo un conjunto vacío de instancias, es resaltado en color rojo. Si todo objeto es satisfacible entonces el diagrama será consistente. Deducciones no explícitas serán visualizadas en color verde para ser distinguidas por sobre los elementos originales del diagrama. En particular, la herramienta mostrará gráficamente: relaciones de inclusión inferidas y restricciones de cardinalidad más restrictivas que las originales. Finalmente, aunque las nuevas deducciones son incluidas en el mismo diagrama, es decisión de los usuarios mantenerlas como parte del modelo o descartarlas. Deducciones que son descartadas serán nuevamente mostradas luego del proceso de verificación si continúan implícitas.

Una vista esquemática de la funcionalidad e interacción dentro de *crowd* es mostrada en la figura 5.1. Este esquema describe cómo es la interacción de los componentes principales de la arquitectura y cómo se genera una nueva visualización para luego ser refinada y comparada con la versión original del gráfico. Notar que, por limitaciones tecnológicas en algunos motores de

consulta y para una mejor performance, *crowd* genera dos representaciones de una misma ontología. La primera en sintaxis OWL/XML, la cual es enviada a un motor SPARQL-DL para luego reconstruir el modelo visual (dicho motor también razona sobre la ontología de entrada) con posibles nuevas relaciones de subclasificación, y la segunda, en sintaxis OWLlink, la cual es procesada por un razonador que soporta el protocolo OWLlink y que realiza consultas sobre las clases involucradas en las cardinalidades del modelo. Finalmente, el resultado de ambos es integrado a un modelo visual final.

Por último, ilustramos esta descripción funcional con un ejemplo para mostrar las tareas de verificación de una ontología gráfica en *crowd*, a partir de un dominio real. *AtomOwl*[1] es una ontología para capturar la semántica del formato RFC4287, para gestión de contenido en linea tales como weblogs, podcasts y videocasts. La gestión de contenido en linea es útil para alertar a lectores interesados sobre cambios en sitios web. Un vista de este modelo es mostrados en la figura 5.2 (izq), y ha sido representada en *crowd* desde la versión original reportada en el sitio de *AtomOwl*. Debido a que *crowd* no soportan aún jerarquías gráficas de roles, los siguientes axiomas OWL 2 fueron incluidos en modo textual (`entry` es una sub propiedad de `source` y `contains`):

```
<owl:SubObjectPropertyOf>
 <owl:ObjectProperty
 IRI="http://bblfish.net/work/atom-owl/2006-06-06/#entry"/>
 <owl:ObjectProperty
 IRI="http://bblfish.net/work/atom-owl/2006-06-06/#source"/>
</owl:SubObjectPropertyOf>

<owl:SubObjectPropertyOf>
 <owl:ObjectProperty
 IRI="http://bblfish.net/work/atom-owl/2006-06-06/#entry"/>
 <owl:ObjectProperty
 IRI="http://bblfish.net/work/atom-owl/2006-06-06/#contains"/>
</owl:SubObjectPropertyOf>
```

[1] `https://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.html`

Este modelos puede ser reproducido en *crowd* siguiente los pasos en `https://sites.google.com/a/fi.uncoma.edu.ar/german-braun/crowd` y cargando el diagrama adjunto en la misma locación.

Algunas de las clases más importantes de *AtomOwl* son: `Feed`, `Entry` y `FeedOrContent`, las cuales están relacionas por medio de las relaciones `entry`, `contains` y `source`, juntos con sus respectivas restricciones de cardinalidad.

Una vista rápida nos permite visualizar y comprender preliminarmente el modelo, sin embargo, esto no es suficientemente preciso, ya que esconde importantes aspectos que podrían mejorar su calidad. Dichos aspectos están implícitos en el diagrama. La figure 5.2 (der.) revela cómo *crowd* muestra tales deducciones, dando una relación exacta entre `Entry` y `Feed`, además de una nueva cardinalidad en las asociaciones `entry` y `contains`.

El razonamiento de *crowd* permite concluir sobre un generalización inesperada entre `Entry` y `Feed` bajo la actual definición del modelo visual. Para entender por qué estos sucede, debemos considerar el hecho de que `entry` es una subpropiedad de `source`. Entonces, necesariamente `Entry` es una subclase de `Feed` debido a que cada individuo en `Entry` es requerido para estar relacionado a sólo un `Feed`, sin embargo, también es requerido formar parte del rango de `Feed`.

En conclusión, la complejidad inherente al modelado de dominios reales, como resalta el ejemplo anterior, y el limitado soporte que los ambientes actuales a los modeladores para realizar su tareas, es un signo visible de que ambientes para desarrollar ontologías deben estar dotados con tareas para creación, mantenimiento, evaluación y documentación de las ontologías.

5.3. Un Editor para UML

El editor UML implementado en *crowd* es sólo el primero de un grupo de ambientes visuales (junto a EER y ORM 2) previstos para implementar

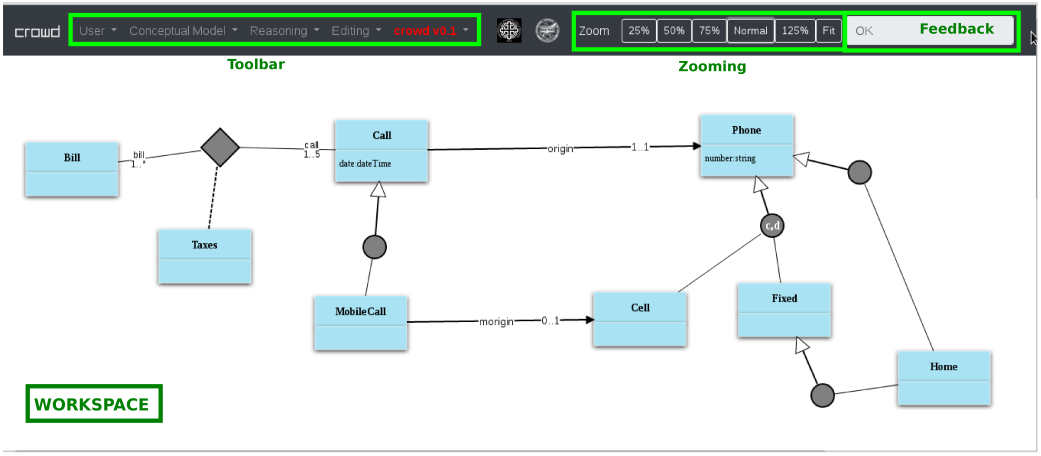


Figura 5.3: interfaz de Usuario de *crowd*

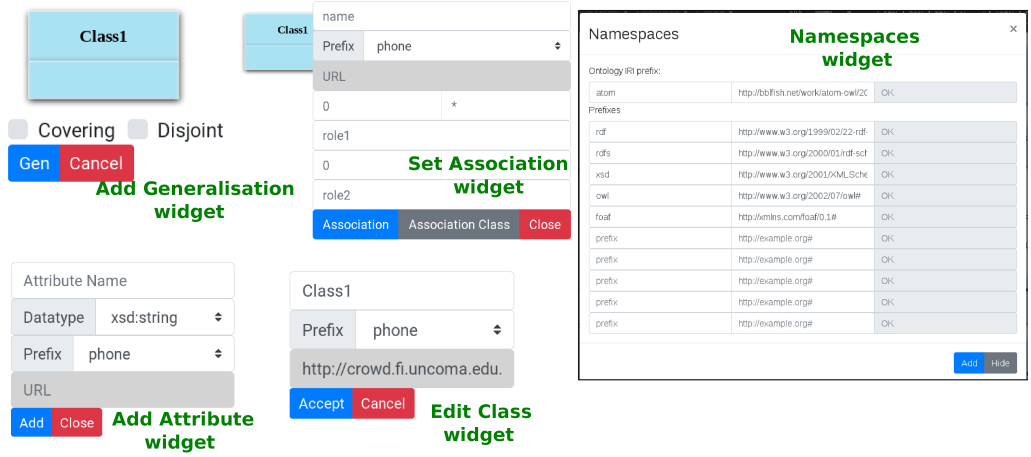


Figura 5.4: Widget de la interfaz de Usuario de *crowd*

en la herramienta. Al momento de presentación de este trabajo, ya tenemos disponible pero no publicado un prototipo que soporta edición de modelos EER y comenzamos con la implementación de un editor para diagramas ORM 2. A continuación, describimos el editor UML provisto por *crowd*. Los restantes seguirán los mismos lineamientos.

El editor UML de *crowd* presenta un espacio de trabajo para el lenguaje visual. Los usuarios podrán introducir sus credenciales, en caso de tenerlas, para almacenar sus diagramas. Sin embargo, la herramienta puede ser usada sin necesidad de registrarse, aunque ningún estado será almacenado en el servidor. La figura 5.3 muestra el espacio de trabajo actual, dónde se identifican: una barra de tareas, incluyendo opciones para gestión de usuarios (`User Management`), opciones para el modelado (`Conceptual Model`), configuraciones del razonamiento (`Reasoning`) y opciones de edición para el gráfico actual (`Editing`). La interfaz de usuario también dispone de un paleta para incrementar o decrementar el tamaño de la visualización (*zoom*) y una ventana de *feedback*, para informar a los usuarios sobre el estado actual de la interfaz.

Por otro lado, la figura 5.4 muestra un conjunto de componentes gráficos (*widgets*) disponibles para la construcción de modelos visuales y para la definición de espacios de nombres del diagrama. Las opciones de modelado y los *widgets* son actualmente accedidos mediante el botón izquierdo del mouse y estando sobre la primitiva gráfica. Cada primitiva provee su propio menú.

Tipos de objetos son creados seleccionando la opción de edición desde la barra de tareas. Una vez creada, el sistema asigna un nombre nuevo, el cual puede ser editado definiendo un nuevo nombre y un prefijo seleccionando la opción `Edit` desde el menú de dicho objeto. Para gestionar la identificación interna de las primitivas, los nombres de los elementos del diagrama son URIs únicas. URIs puede ser cargadas previamente y validadas en el espacio de trabajo accediendo a la opción `Namespaces` en la barra de tareas. Cada espacio de trabajo en *crowd* tiene un conjunto de URIs estándar predefinidas. Los atributos pueden ser adicionados a los objetos por medio de la opción

`Attributes` desde el menú del objeto. Para cada atributo se debe indicar un rango definido en el espacio de nombres de XMLSchema[2], para los tipos de datos soportados por la definición de OWL 2.

Con el objetivo de crear estructuras gráficas más complejas, las relaciones requieren de más parámetros y por lo tanto, los usuarios deben seleccionar una clase origen en el diagrama, luego ingresar los parámetros requeridos en el `Association`. Finalmente, seleccionar la clase destino. *crowd* presenta al usuario un formulario para definir nombres, prefijos, roles y sus cardinalidades. En particular, las asociaciones binarias sin clases se crean con nombres y cardinalidades por defecto, es decir que toman el nombre de sus clases y $0..*$, a menos que se indique lo contrario. Estás asociaciones pueden ser removidas del diagrama utilizando una cruz roja que resalta del link cuando se invoca al evento *mouseover*. En las asociaciones binarias con clases, los roles se visualizan de manera explicita y pueden ser editados, junto con sus cardinalidades, utilizando el icono diamante de la asociación.

Durante la creación de generalizaciones, las propiedades total y disjunto pueden definirse cuando el respectivo *widget* es presentado. Generalización se agregan al diagrama seleccionando, en primer lugar, la clase padre y luego seleccionando cada una de las subclases intervinientes. La edición es también provista por la interfaz, mediante un menú que puede ser accedido utilizando el mouse sobre el círculo que define la generalización. En tal menú pueden también incluirse más subclases para estar en la primitiva.

Otras funcionalidades generales tales como la exportación de OWL 2, exportación e importación de los diagramas en un objeto JSON, y la documentación de las ontologías, pueden ser invocadas desde el menú `Conceptual Model` de la barra de tareas.

Por último, el editor no implementa actualmente técnicas de layout automático para el manejo de ontologías de gran tamaño, sin embargo, algunos trabajos preliminares fueron publicados en [MBCF18], basadas en la reduc-

[2] `http://www.w3.org/2001/XMLSchema#`

ción del número de cruces en grafos [KS14].

Un versión interactiva mostrando las funcionalidades principales de *crowd* puede reproducirse en `https://sites.google.com/a/fi.uncoma.edu.ar/german-braun/crowd-survey`.

5.4. Evaluación basada en Usuarios

Para la obtención de resultados sobre como los usuarios perciben *crowd*, hemos llevado a cabo un proceso de evaluación, recolectando *feedback* por medio a una sesión experimental y posterior encuesta a un grupo de estudiantes. Es importante aclarar que ninguno de los participantes del experimento recibió entrenamiento o guía alguna previo a la sesión sobre como modelar con la herramienta. Actualmente, la encuesta diseñada y utilizada para el presente reporte, permanece en línea[3] para generar un proceso de retroalimentación permanente. Una versión *offline* de la encuesta es incluida en el apéndice B.

La usabilidad es uno de los factores principales que definen la calidad de un producto software e implica que el sistema realice las tareas necesarias para las que fue desarrollado y que, además, sea fácil de usar [Lau05]. Para diseñar interfaces adecuadas es necesario estudiar a los potenciales usuarios y sus tareas, y luego desarrollar un prototipo y revisarlo en conjunto con estos usuarios. Para poder seguir estas premisas, se diseño una evaluación de usabilidad cuyos objetivos principales son los siguientes:

- Obtener un valor concreto de referencia de la usabilidad actual de *crowd*, relativa a un grupo de usuarios del ambiente académico. Dicho valor será calculado en base a la Escala de Usabilidad de Sistemas [LS09]. Si bien, como considera la definición de SUS [Bro96], el valor obtenido a partir de esta evaluación estará supeditados al contexto particular en el cual se desarrolló el experimento, esta medida es válida

[3] `https://sites.google.com/a/fi.uncoma.edu.ar/german-braun/crowd-survey`

para ponderar la interfaz de un software y puede también ser utilizada para comparar la usabilidad a través de diversos contextos.

- Falencias y sugerencias funcionales para mejorar la interacción de los usuarios con el software, a partir de ejercicios con la herramienta y de un cuestionario basada en Factores de Usabilidad [Lau05]. Los resultados obtenidos a partir de este análisis serán considerados para mejorar las versiones posteriores de la herramienta y en consecuencia, re evaluar el factor SUS.

Todos los experimentos fueron extraídos de trabajos relacionados, mientras que otras evaluaciones, como por ejemplo una comparación exhaustiva con herramientas de ingeniería ontológica, también fueron consideradas, sin embargo, al momento de la validación de este trabajo no contamos con una comunidad de usuarios que trabaje con ellas.

5.4.1. Diseño del Experimento

Se reclutaron 14 (catorce) estudiantes de grado en Ciencias de la Computación y en Sistemas de Información, con diversos conocimientos sobre los tópicos relacionados a este trabajo: ontologías, lenguajes de modelado conceptual y Lógicas Descriptivas. La mitad de ellos ya había tenido una mínima interacción con otras herramientas tales como Protégé. La elección de este grupo de estudiantes nos garantiza la posibilidad de identificar requerimientos que no se presentan intuitivos para quienes nunca han trabajado con ontologías (o solamente tiene conceptos teóricos al respecto).

Cada participante fue asignado a una estación de trabajo en el mismo laboratorio, con los requisitos mínimos para usar *crowd*, un navegador Web tal como Mozilla or Chrome. El objetivo de generar una sesión experimental de este tipo fue importante para estimular la discusión entre los participantes mientras interactuaban con la herramienta.

Una versión de *crowd* fue instalada en la siguiente dirección: `http://`

`crowd.fi.uncoma.edu.ar`, el cual también contiene toda la documentación del proyecto. Luego, se realizó una introducción a las funcionalidades principales de la herramienta junto con un video demostrativo, de aproximadamente 10 minutos.

Se diseñó una encuesta en linea, con dos secciones claramente definidas. La primer sección incluye ejercicios prácticos de interacción con la interfaz gráfica junto con algunas consultas relacionadas a estos ejercicios. Asimismo, esta sección incluye un conjunto de tareas de modelado simples y aleatorias donde cada participante debe responder sobre la cantidad de intentos para realizarla correctamente.

- (Tarea 1) *Agregar una nueva URI junto con su prefijo al espacio de nombres del modelo, luego crear una clase UML usando tal prefijo*,
- (Tarea 2) *Modelar una Asociación Binaria con un clases asociada. Definir sus roles y cardinalidades*,
- (Tarea 3) *Editar la Asociación modelada previamente, modificando alguno de sus roles*,
- (Tarea 4) *Definir una Generalización y agregar una nueva subclase*,
- (Tarea 5) *Exportar el diagrama resultando a OWL 2*,
- (Tarea 6) *Exportar el diagrama resultante como un objeto JSON*, y
- (Tarea 7) *Remover el diagrama del espacio de trabajo e importar nuevamente el objeto JSON previo.*

La segunda sección de la encuesta consiste de dos cuestionarios complementarios: Factores de Usabilidad[4] [Lau05] y Escala de Usabilidad de Sistemas (en inglés, SUS)[5] [LS09] Además, se incluyeron algunas consultas sobre

[4] *Factors of Usability*
[5] *System Usability Scale (SUS)*

el conocimiento de los participantes de los temas relacionados, con el objetivo de dotar al experimento con otras dimensiones.

Los siguientes son los factores de usabilidad incluidos, donde cada uno puede ser ponderado desde "Pobre" a "Excelente":

- (FU1) *Apto para uso*,
- (FU2) *Facilidad de aprendizaje*,
- (FU3) *Eficiente para realizar tareas*,
- (FU4) *Fácil de recordar*,
- (FU5) *Satisfacción subjetiva*, y
- (FU6) *Comprensibilidad*

Por otro lado, para utilizar el SUS, se presentan las siguientes preguntas, las cuales pueden ser ponderadas en una escala de 5 puntos: 1 para "Muy en desacuerdo" a 5 para "Completamente de acuerdo". Asimismo, preguntas impares son consideradas positivas por la escala mientras que las pares son las negativas.

- (Q1) *Utilizaría el sistema con frecuencia*,
- (Q2) *Encuentro al sistema complejo de manera innecesaria*,
- (Q3) *El sistema es fácil de usar*,
- (Q4) *Necesitaría el soporte de un técnico para poder usar la herramienta*,
- (Q5) *Varias funciones no fueron integradas apropiadamente*,
- (Q6) *Se encontraron demasiadas inconsistencias en el sistema*,
- (Q7) *Otras personas aprenderían a usar el sistema rápidamente*,

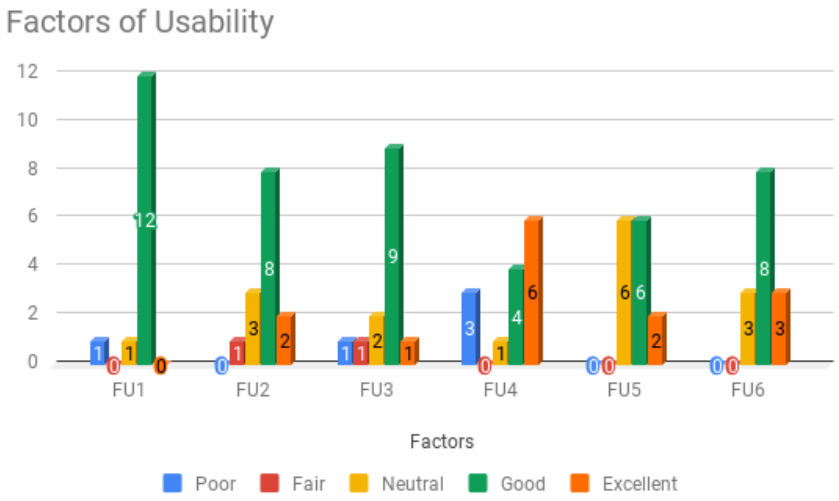


Figura 5.5: Factors of Usability

- (Q8) *El sistema es muy incómodo de usar*,
- (Q9) *Sentí seguridad al utilizar el sistema*,
- (Q10) *Necesito aprender muchos nuevos conceptos para utilizar el sistema*

5.4.2. Resultados

Los resultados reportados por el experimento mostraron que los participantes no tuvieron inconvenientes en comprender las funcionalidades de *crowd* y en distinguir clases, relaciones y atributos, y en modelar los ejercicios prácticos. Considerando aspectos negativos, muchos alumnos resaltaron la necesidad de una edición más dinámica para nombrar las clases (doble-click). Asimismo, resaltaron la necesidad de incorporar explicaciones para las conclusiones derivadas del razonamiento lógico.

Entre los aspectos positivos, los participantes concluyeron que la herramienta implementa funcionalidades interesantes, lo cual permite agilizar el

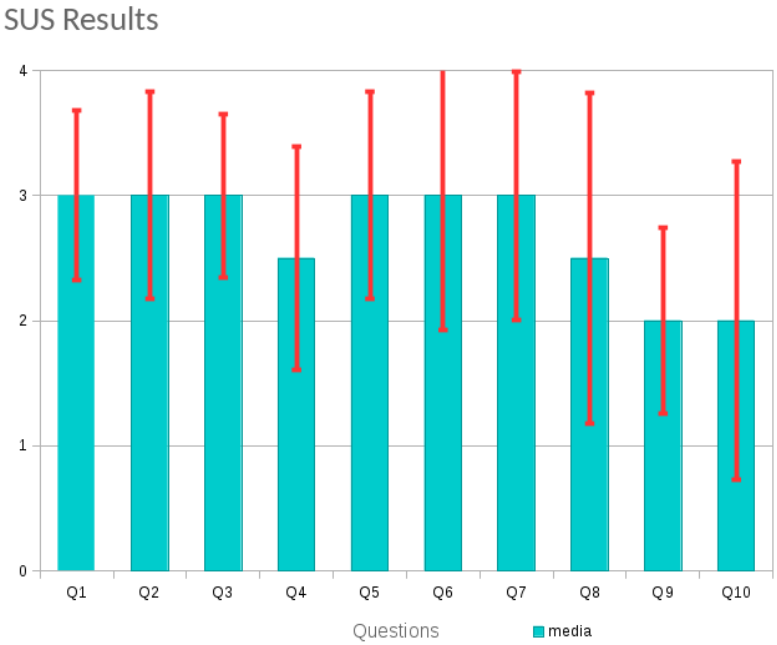


Figura 5.6: SUS Results

modelado, además del acceso vía web sin necesidad de realizar ningún tipo de instalación. Los aspectos más remarcados fueron las características web, visual y soporte lógico para validación. A continuación detallados estos resultados.

Considerando el primer experimento (Tarea 1-7), sobre un total de 98 posibles respuestas (7 tareas por 14 participantes), 97 fueron completadas exitosamente. Los resultados mostraron que 82,5 % (80 de 97) fueron realizadas en el primer intento; 12,4 % (12 de 97) en el segundo y sólo 5 % (5 de 97) en más de dos. Sólo tres participantes completaron todas las tareas requeridas en un sólo primer intento.

Por otro lado, a pesar del poco tiempo que los participantes tuvieron para familiarizarse con la herramienta, los factores de usabilidad mostraron resultados por demás positivos e indicando un promedio aceptable de usabilidad, "Bueno" en términos de su escala. La mayoría de los participantes sugirieron que *crowd* es apto para usar, fácil de aprender, eficiente para usuarios

frecuentes y fácil de comprender. Estos mismos resultados también sugirieron que no es una herramienta posiblemente fácil de recordar para usuarios ocasionales. Sin embargo, este punto débil es compensado con la facilidad de aprendizaje para usuarios nuevos. La figure B.6 gráfica cada factor junto con su evaluación.

El último experimento reportó un factor de usabilidad de 65,5 %, indicando un valor aceptable de usabilidad, y cercano al valor promedio del 70 %. En este contexto, sólo uno de los participantes considero la herramienta innecesariamente compleja, mientras que sólo dos sugirieron que necesitaría soporte técnico para utilizarla. Los resultados del SUS son descriptos en la figura B.1, donde se representa un valor medio de la usabilidad (de 0 a 4, de acuerdo a la contribución de cada pregunta al valor total de la escala) junto con su desviación estándar.

5.5. *crowd* es una instancia del modelo *WYSIWYM*

El concepto *WYSIWYM* [KA15] formaliza la relación entre los modelos semánticos para representar conocimiento y los elementos de un interfaz de usuario para visualización, exploración y creación de dichos modelos. Entre sus beneficios, puede ser usado para evaluar interfaces, proveer terminologías para comunicación entre involucrados en los modelos y para los ingenieros de software.

Dada la versatilidad del concepto, planteamos, en esta sección, una instanciación de un *WYSIWYM* para validar la interacción que nuestra herramienta posee entre sus elementos gráficos y la representación de ontologías OWL 2. Primeramente, detallamos los principales conceptos propuestos por los autores en [KA15] y, finalmente, caracterizamos a *crowd* siguiendo esta formalización.

Una interfaz *WYSIWYM* se define en términos de un modelo semántico

para representación de información; técnicas de visualización, exploración y creación; y componentes llamados "helpers". Los primeros permiten expresar el significado de la información a través de entidades y sus relaciones. Por ejemplo, modelos basados en grafos son adecuados para representar esquemas tales como los diagramas Entidad-Relación.

Las técnicas de visualización tiene como objetivo presentar esta información a través de representaciones visuales, facilitando el acceso a humanos. Asimismo, las técnicas de exploración tratan con técnicas para la navegación de dichas representaciones visuales. Finalmente, las técnicas de creación permiten definir y agregar más semántica a los modelos por medio de funcionalidades de edición. Los componentes denominados "helpers", son también utilizados para mejorar la calidad de los modelos y actual como una extensión de las funcionalidades principales del *WYSIWYM*. Ejemplos de estos componentes son los de recomendación, mediante sugerencias a los usuarios; colaboración, habilitando capacidades de edición compartida, entre otros.

Formalmente, el modelo es una quíntupla (D, V, X, T, H), dónde D son los modelos de representación semántica; V es un conjunto de tuplas (v, C_v) asociando a técnicas de visualización v y un conjunto de configuraciones posibles C_v para v; X es un conjunto de tuplas (x, C_x) asociando una técnica de exploración x y un conjunto de configuraciones posibles C_x para x; y T es un conjunto de tuplas (t, C_t), dónde t es una técnica de creación y C_t, un conjunto de posibles configuraciones. H es un conjunto de componentes "helpers". Una instancia de un *WYSIWYM* agrega un nuevo elemento para la formalización, llamado *binding*, el cual indica cómo los elementos de los modelos semánticos son mapeados a una técnica de visualización, exploración y creación.

En este punto, estamos en condiciones de dar una formalización de *crowd* como un *WYSIWYM*. El objetivo es mostrar que las ontologías generadas en la herramienta pueden mapearse a elementos de su interfaz gráfica para tareas de ingeniería ontológica. Los servicios de razonamiento que permiten explicitar conocimiento posiblemente implícito, actúan como "helper" . A

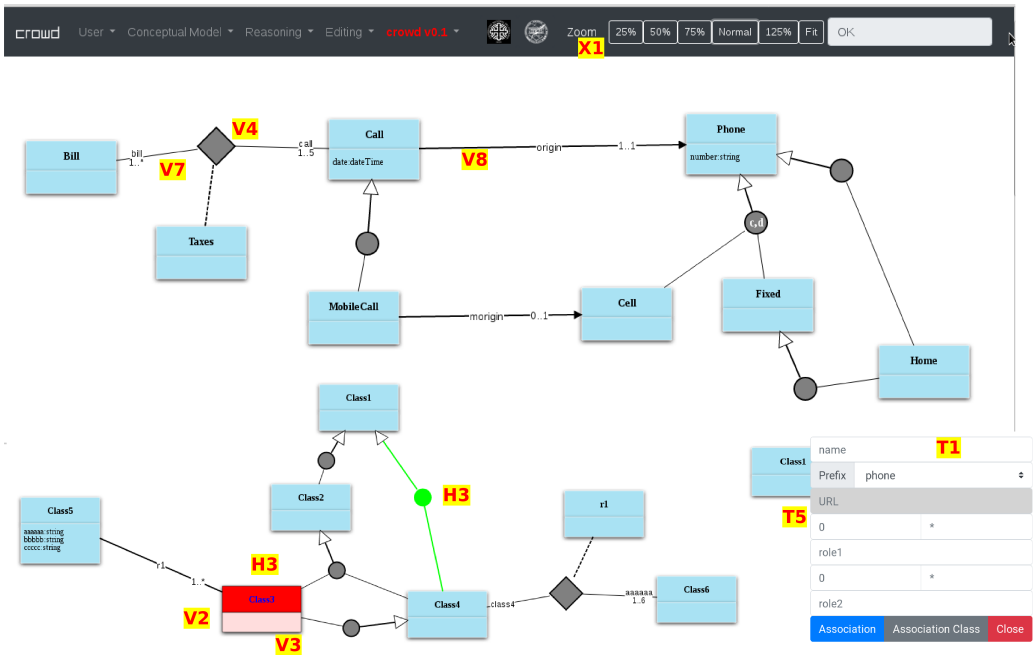


Figura 5.7: Interface *WYSIWYM* de *crowd*.

continuación damos detalles de esta evaluación.

crowd es una interfaz *WYSIWYM* para la creación de ontologías gráficas, descripta como:

- **D**: OWL 2,
- **V**: asociación de primitivas gráficas de los lenguajes gráficos (iconos para clases y relaciones); templates textuales para axiomas OWL 2; clases inconsistentes y axiomas implícitas son resaltadas para distinguirlas del modelo original,
- **X**: zooming,
- **T**: edición a través de formularios, gráficos y menús contextuales,
- **H**: recomendaciones gráficas y textuales basados en razonamiento automático basado en lógica.

Cuadro 5.1: *Bindings* posibles para la instanciación de *crowd*. V = visualización, E = exploración, C = creación. T = técnica, I = instancias, CL = classes, R = relaciones, Tg = tags, VL = valores, TD = tipos de datos, N/A = no aplica.

	Estructura	UI	T	I	CL	R	Tg	VL	TD
V	Texto	highlighting	text formatting	N/A	●	❍		N/A	
	Imágenes	highlighting	images color effects	N/A	❍	❍		N/A	❍
			Icons	N/A	●			N/A	
		associating	line connectors	N/A		❍		N/A	
			arrow connectors	N/A		●		N/A	
E	Texto	zooming		N/A				N/A	●
	Imágenes	zooming		N/A	●	●		N/A	
C	Texto/Imágenes	Edición de Forms		N/A	●	●		N/A	●
		Íconos		N/A	●	●		N/A	
		Menú Textual		N/A	●	❍		N/A	❍

La tabla 5.1 presenta también los *bindings* para *crowd*. Círculos negros (●) representan situaciones cuando una técnica específica soporta completamente un elemento del modelo semántico tales como iconos para clases, flechas para generalizaciones, entre otras. *crowd* resalta conocimiento implícito e inconsistencias por medio de colores, pero sólo para axiomas que pueden ser gráficamente representados. Este ejemplo es un soporte parcial y es representado por círculos blancos (❍). Otro ejemplo de soporte parcial está relacionado a los menús contextuales y a la falta de opciones de edición para relaciones binarias y atributos.

La asociación de clases usando lineas sin flechas para indicar otros axiomas OWL2, tales como por ejemplo disyunción, requieren de una primitiva gráfica, sin embargo, esto no es soportado por la herramienta e indicado con celdas blancas en la tabla. Por último, "N/A" difiere de anterior ya indica que no se provee ningún soporte para un determinada técnico o primitiva gráfica y, además, está fuera del alcance de *crowd*. Esta tabla junto con su correspondencia en la interfaz de *crowd* es visualmente representada en la figura 5.7.

La evaluación permite concluir, positivamente, que *crowd* es una herramienta capaz relacionar coherentemente un modelo semántico con los respectivos elementos de una interfaz gráfica. Sin embargo, las versiones posteriores de *crowd* deben mejorar el soporte parcial identificado, principalmente proveer técnicas de visualización y edición de relaciones. Aquí resaltamos la consistencia de esta evaluación con la basada en usuarios, donde los participantes resaltaron la necesidad de mejorar el acceso a menús para incrementar la flexibilidad de la herramienta.

5.6. Conclusiones

crowd v1.0 es la primer herramienta web desarrollada como un proceso de visualización de conocimiento basado en ontologías y un sistema de

manipulación de ontologías gráficas (*GOMS*). Ontologías gráficas puede ser definida a partir de un lenguaje gráfico y conjuntos de mapeos que permiten dar un significado preciso al modelo visual, pero también extraerlo desde una formalización lógica. Esta característica es complementada con otras funcionalidades necesarias para la ingeniería ontológica: definición de espacios de nombres, documentación e interoperabilidad (importación y exportación OWL 2 y JSON).

Las evaluaciones basadas en usuarios realizadas sobre *crowd v1.0* arrojan resultados positivos, con una usabilidad del 65,5 %. Dicho valor es obtenido a partir del cálculo de un factor de usabilidad (SUS). Ampliamente utilizado y aceptado, el SUS produce un único número representando una medida compuesta de la usabilidad completa de un sistema en estudio. Si bien, algunos autores indican que un factor aceptable de usabilidad debe superar el 70 %, consideramos como beneficioso nuestro análisis, dada una primera versión oficial de *crowd*. Finalmente, concluimos que la validación de *crowd* requiere de una evaluación con mayor profundidad, para la cual nos encontramos trabajando al momento de la presentación de esta Tesis.

Durante la parte final del capítulo, también hemos detallado una instanciación de *crowd* como un modelo para visualizar contenido semántico, definido como *WYSIWYM*. Esta evaluación es por demás positiva para nuestro aporte, ya que concluye sobre cómo un formalismo para representar conocimiento (como las ontologías), deben y pueden ser ligadas a un conjunto de elementos de una interfaz gráfica para visualización, exploración y creación de modelos.

A continuación, damos un análisis general de las herramientas relevadas en la literatura y que fueron evaluadas para esta investigación, junto con una descripción de cada una considerando los requerimientos propuestos para *crowd*.

Capítulo 6

Comparación con otras Herramientas

Contents

Durante el transcurso de este trabajo hemos relevado y estudiado herramientas existentes que posibilitan la creación, edición y visualización de

ontologías. Este análisis ha sido realizado siguiendo los lineamientos del Proceso de Visualización detallado en el capítulo 3 y considerando los siguientes requerimientos que fueron planteados *crowd* y justificados a lo largo de esta investigación:

(R1) *Soporte para Ingeniería Ontológica*

(R2) *Edición Visual*

(R3) *Accesible vía Web*

(R4) *Compatibilidad con la Web Semántica*

(R5) *Framework Extensible*

(R6) *Servicios de Razonamiento Integrados*

(R7) *Independencia de los Lenguajes Visuales*

(R8) *Manejo Simultaneo de Modelos Visuales*

Una vez revisados los requerimientos que rigen nuestra propuesta, estamos en condiciones de conducir un análisis detallado y comprensivo sobre un conjunto de herramientas reportadas por la bibliografía.

6.1. Análisis General de la Comparación

Como resultado de revisiones de la literatura y búsqueda web, se han identificado las siguientes herramientas para desarrollo de ontologías que, en algún grado, están activamente mantenidas: WebProtégé [TNNM13], Protégé [KFNM04] OWLViz[1], OntoGraf[2], SOVA[3], NORMA [CH10], ICOM [FFT12], TopBraid Composer [Top11], Graphol [CLSS14], OWLGrEd [COLS12], Menthor [MSG+16], NeOn Toolkit [HLSE08], VOWL [LNB14], GrOWL [KWV07],

[1] `http://protegewiki.stanford.edu/wiki/OWLViz` Nov 2018
[2] `http://protegewiki.stanford.edu/wiki/OntoGraf` Nov 2018
[3] `http://protegewiki.stanford.edu/wiki/SOVA` Nov 2018

Cuadro 6.1: Comparación de Herramientas Relevadas. e = edición, m = mantenimiento, ev = evaluación. ✓= la herramienta está alineada con el respectivo requerimiento de *crowd*, ✪= soporte parcial y, ✗= no hay soporte para el requerimiento.

	(R1)	(R2)	(R3)	(R4)	(R5)	(R6)	(R7)	(R8)
WebProtégé	✓	✗	✓	✓	✓	✗	✗	✗
Protégé OWLViz	✓	Nodos-enlaces (subclasificación)	✗	✓	✓	✪ *solo subclasificación	✗	✗
Protégé OntoGraf	✓	Nodos-enlaces (subclasificación)	✗	✓	✓	✗	✗	✗
Protégé SOVA	✓	Nodos-enlaces (OWL 1)	✗	✓	✓	✪ *solo subclasificación	✗	✗
NORMA	e/m/ev	ORM 2 ($\mathcal{ALCQI}$)	✗	✗	✓	✓	✗	✗
ICOM	e/m/ev	EER ($\mathcal{ALCI}$)	✗	✗	✗	✓	✗	✓
TopBraid Composer	e/m/ev	Nodos-enlaces/ UML (subcjto OWL 2)	✗	✓	✓	✪ *solo subclasificación	✗	✗
OWLGrEd	e/m	UML Extendido (subcjto OWL 2)	✓	✓	✗	✗	✗	✗
eddy - Graphol	e/m	Graphol	✗	✗	✗	✗	✗	✗
Menthor	✓	OntoUML	✗	✗	✓	✗	✗	✗
NeOn Toolkit	e/m/ev	Nodos-enlaces(OWL 2)	✗	✓	✓	✪ *solo subclasificación	✗	✗
VOWL	✗	VOWL	✓	✗	✗	✗	✗	✗
Graffoo	e	Nodos-enlaces (OWL 2)	✗	✓	✗	✗	✗	✗
crowd	✓	UML	✓	✓	✓	✓	✓	✗

OntoTrack [Lie03], SWOOP [KPS+05], Hozo [KKIM02], y Graffo [FGP+14]. Cinco de ellos (VOWL, Graphol, Hozo, GrOWL and Graffo) son meros visualizadores de ontologías, mientras que las restantes exhiben algún grado de integración iteractiva de soporte lógico con modelos gráficos. OntoTrack, GrOWL, SWOOP y Hozo no se encuentran disponibles públicamente para ser descargadas y realizar una evaluación detallada.

Sólo OWLGrEd, VOWL and WebProtégé ofrecen algún tipo de compatibilidad web: OWLGrEd y VOWL son visualizadores web para UML-like y el lenguaje VOWL, respectivamente. WebProtégé provee soporte para edición y definición de espacios de nombres, aunque carece de notación visual. Otros aspectos débiles de las herramientas relevadas es la falta de independencia de lenguajes visuales. Esta característica en el contexto de herramientas integradas con razonamiento lógico requiere de múltiples transformaciones para codificar los modelos visuales, sin embargo, tampoco esto es ofrecido por herramientas relacionadas.

Integración de modelos visuales y razonamiento es provista en algún grado por ICOM y NORMA, para subsumciones, restricciones funcionales y opcionales, disyunción y equivalencias, y ausente en los ambientes restantes. La ausencia de esta integración está vinculada a nuestro requerimiento (R6), el cual establece que los resultados del razonamiento deben reflejarse no solo en el modelo formal, sino también en el modelo visual. Esta es una diferencia importante con respecto a otras herramientas que interactúan con razonadores, debido a que muchas de ellas lo hacen (Protégé, Menthor), pero no lo reflejan en el modelo gráfico.

En relación al soporte para ingeniería ontológica, sólo Protégé (y WebProtégé), provee soporte a una gran cantidad de tareas de ingeniería gracias a su arquitectura altamente flexible, que permite su extensión mediante sofisticados “plug-ins”. Herramientas más actuales tales como Menthor, presentan soporte en la misma linea que nuestros requerimientos.

Diferencias y similitudes entre ontologías y modelos han sido abordadas

por diversas propuestas [AGK06]. Estos criterios están basados principalmente en la necesidad de formalizar modelos conceptuales (p.e. UML, ER, ORM) para expresar ontologías y por otro lado, en las capacidades de razonamiento lógico que son provistas por los lenguajes de ontologías pero no capturas por los modelos. Más allá de obvias diferencias entre la expresividad de los CMLs y los lenguajes de ontologías, muchas herramientas han validado de forma parcial este hecho. De esta manera, la efectividad de usar una sintaxis gráfica basada en estos lenguajes conceptuales para expresar ontologías, está siendo aún considerada para ambientes de ingeniería ontológica [ABAG17, VBJS14]. Evidencias de esto indican que un conjunto común de UML, EER, y ORM 2 pueden ser formalizados en $\mathcal{ALNI}$ [BCM+03a], garantizando razonamiento tratable sobre ellos [FK15]. Por otro lado, diversas notaciones visuales para OWL 2 han sido también propuestas, las cuales están mayormente basadas en grafos [NHL13, CLSS14], con la intención principal de proveer a los modeladores una notación visual unificada.

En conclusión, este análisis da una real motivación para ir hacia la integración de notaciones visuales y sus características comunes con actividades de ingeniería ontológica, en un único ambiente con soporte de razonamiento lógico. La tabla 6.1 resume este análisis, indicando con ✓si la herramienta está alineada con el respectivo requerimiento de *crowd*, ✪si el soporte es parcial y, en otro caso, ✗. Finalmente, en la sección subsiguiente damos detalles de cada una de las herramientas analizadas.

6.2. Revisión de Herramientas Relevadas

En esta sección presentamos un revisión detallada de cada una de las herramientas consideradas para la comparación, y que sustentan los requerimientos definidos para la implementación de *crowd*.

6.2.1. Protégé y WebProtégé

Protégé [KFNM04] es una plataforma Java libre y de código abierto desarrollada por la Universidad de Stanford. Protégé provee funcionalidades básicas para creación, edición y mantenimiento de ontologías, pero requiere de componentes adicionales, *plug-ins*, para soporte de visualización y otras tareas de ingeniería.

La edición en Protégé es principalmente textual y un escaso soporte gráfico es provisto por *plug-ins* como OWLViz y SOVA (Simple Ontology Visualization API). De manera similar el razonamiento es completamente soportado de manera textual, pero las deducciones obtenidas por razonadores externos son limitadas a las de tipo Is-A de manera gráfica. Entre sus aspectos favorables, OWLViz permite navegar grande ontologías de manera incremental y mostrando clases inferidas. También provee algoritmos de layout automático y zoom. SOVA implementa respectivos elementos gráficos para todos los constructores OWL. Estos elementos son diagramados a través de un grafo de fuerzas, el cual resulta en un modelo animado que posiciona los nodos del grafos de manera dinámica. Asimismo, el componente presenta una vista de árbol jerárquica para visualizar subclasificaciones inferidas.

Otras extensiones gráficas, como por ejemplo OntoGraf dan soporte interactivo para navegar sólo las relaciones de subclasificación, individuos, dominios y rangos de las propiedades y equivalencias en OWL. En cuanto a la integración con el razonador, es limitada y levemente soportada. Asimismo, ninguna nueva deducción a nivel textual es reflejada gráficamente usando OntoGraf. OWLViz visualiza la jerarquía de clases de ontologías OWL, las cuales pueden ser además navegadas de manera incremental, mientras que los conceptos inconsistentes y las jerarquías inferidas pueden ser visualizadas luego de razonar sobre el modelo. Si bien OntoGraf ofrece técnicas de layout, la visualización de grande ontologías es un punto débil. Sin embargo, soporta algunas formas complementarias de exploración tales como etiquetas pop-up.

Esta importante limitación también se presenta en la versión Web de

la herramienta denominada WebProtégé [TNNM13], en la cual la interacción con razonadores es nula y tampoco se provee un soporte gráfico, siendo principalmente un repositorio de ontologías con capacidades de edición colaborativa.

6.2.2. TopBraid Composer y NeOn Toolkit

TopBraid Composer [Top11] y NeOn Toolkit [HLSE08] son herramientas desarrolladas sobre la plataforma Eclipse para la ingeniería de ontologías. Permiten la carga y edición textual de ontologías desde archivos en formato OWL. Similar a Protégé, también muestran inferencias gráficamente, pero limitadas a relaciones Is-A. El primero, sin embargo, también provee otro formato de visualización por medio de gráficos RDF, en tanto que el soporte de visualización en NeOn Toolkit es mediante el *plug-in* "OWL Ontology Visualization".

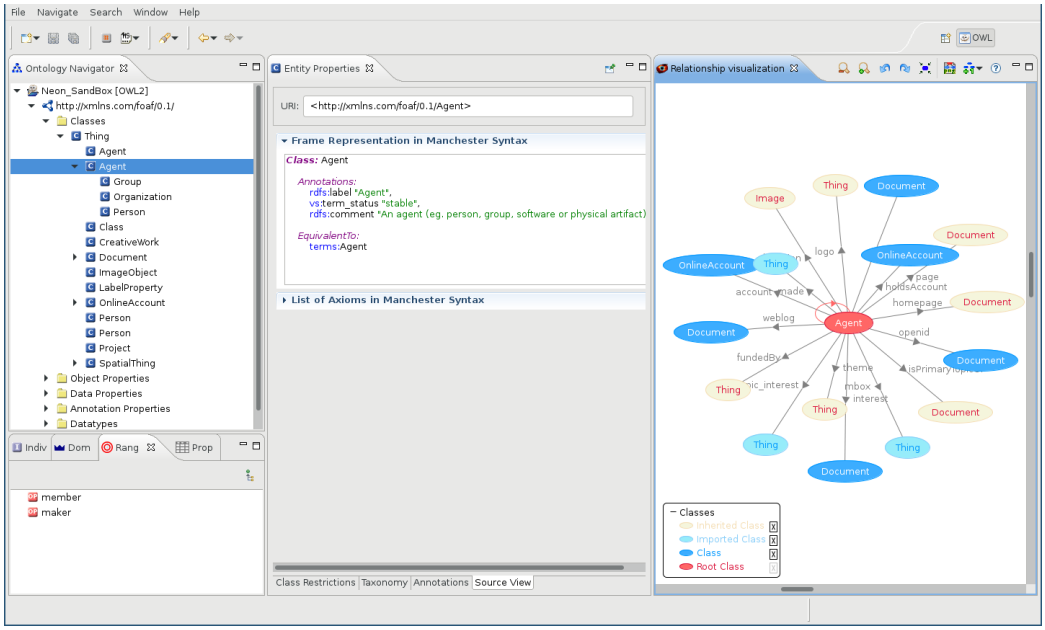


Figura 6.1: Captura de Pantalla de NeOn Toolkit mostrando un Agente de la ontología FOAF.

Las tareas de edición de ontologías en TopBraid Composer son provistas

sobre una representación jerárquica de árbol. En relación a su interfaz de usuario, la herramienta no provee técnicas de layout automático, pero grafos y diagramas puede ser alineadas invocando una opción de reordenamiento. Finalmente, otras funcionalidades visuales permiten la selección de partes específicas de un diagrama a partir de una vista global del modelo. Con respecto a NeOn Toolkit, su visualizar sí implementa ténicas de layout automático, el cual permite iniciar y terminar el proceso para ajustar la ontología en la pantalla. Es es previsto mediante *plug-ins* tales como KC-Viz.

En la Figura 6.1 se muestra la herramienta NeOn Toolkit con la ontología FOAF (Friend of a Friend) cargada desde la URI `http://xmlns.com/foaf/0.1` y con el *plug-in* de visualización centrado en la clase "Agent" mostrando sus distintas subclases.

A diferencia de *crowd*, estas herramientas no proveen de servicios de razonamiento *out-of-the-box*. Sin embargo, algunas se apoyan de *plug-ins* y de la posibilidad de exportar a OWL para poder interactuar con razonadores y realizar consultas, incluso utilizan la sintaxis RDF y soportan SPARQL para hacer consultas específicas solamente en cuanto a sus individuos. En cuanto a la visualización de ontologías, *crowd* permite la edición en el mismo lenguaje gráfico y no requiere que el usuario posea conocimiento acerca de DL ni de la sintaxis OWL, en contraposición con TopBraid Composer y NeOn Toolkit que se enfoca en una edición textual en archivos OWL.

6.2.3. NORMA

NORMA (Natural ORM Architect) [CH10] es una herramienta desarrollada por la ORM Foundation[4] para modelado en ORM 2 (fact-oriented modelling language), y es implementado como un componente de código abierto para Microsoft Visual Studio. NORMA provee un gran poder expresivo para propósitos de modelado conceptual gráfico, y complementado con soporte para la verbalización automática y validación de errores para notificar a los

[4] `https://www.ormfoundation.org/`

modeladores sobre fallas sintácticas. Este tipo de validación es diferente a la validación semántica propuesta a lo largo de este trabajo, ya que en NORMA, el conocimiento implícito permanece oculto. Tempranas versiones de NORMA no soportaban interacción con razonadores lógicos, sin embargo, algunos trabajos recientes [Spo16] incorporan este soporte.

Desde el punto de vista de la manipulación y navegación, NORMA soporta ventanas concurrentes con un mismo modelos para poder copiar elementos entre diversos diagramas, e hiperlinks desde su navegador de verbalización. Por último, implementa ventanas contextuales que permiten visualizar los vecinos más cercados de un elemento ORM seleccionado.

6.2.4. ICOM

La herramienta más relacionada a nuestro trabajo es ICOM [FFT12], considerándola como la herramienta fundacional de *crowd*. ICOM adopta un lenguaje gráfico neutral para representar ontologías por medio de diagramas similares al UML o ER, aprovechando la ventaja de que sea claro aún para personas que no están familiarizados con los lenguajes ontológicos más clásicos.

ICOM posee una interfaz que brinda apoyo a la ingeniería ontológica para crear modelos claros y de criterios de calidad medibles. Esto se logra por medio de la formalización interna, traduciendo los diagramas y restricciones entre modelos a una lógica basada en clases. Su editor provee funcionalidades para manipular el diagrama y ajustar los modelos al espacio de trabajo y zoom. Posteriores versiones de ICOM fueron dotadas con algoritmos estándar de layout.

En la Figura 6.2 se presenta una captura de pantalla de la herramienta con una ontología de ejemplo. ICOM ofrece una opción para conectarse a un razonador, chequear la consistencia del modelo e inferir restricciones posiblemente ocultas en el modelo gráfico. Dichas deducciones pueden ser incluidas

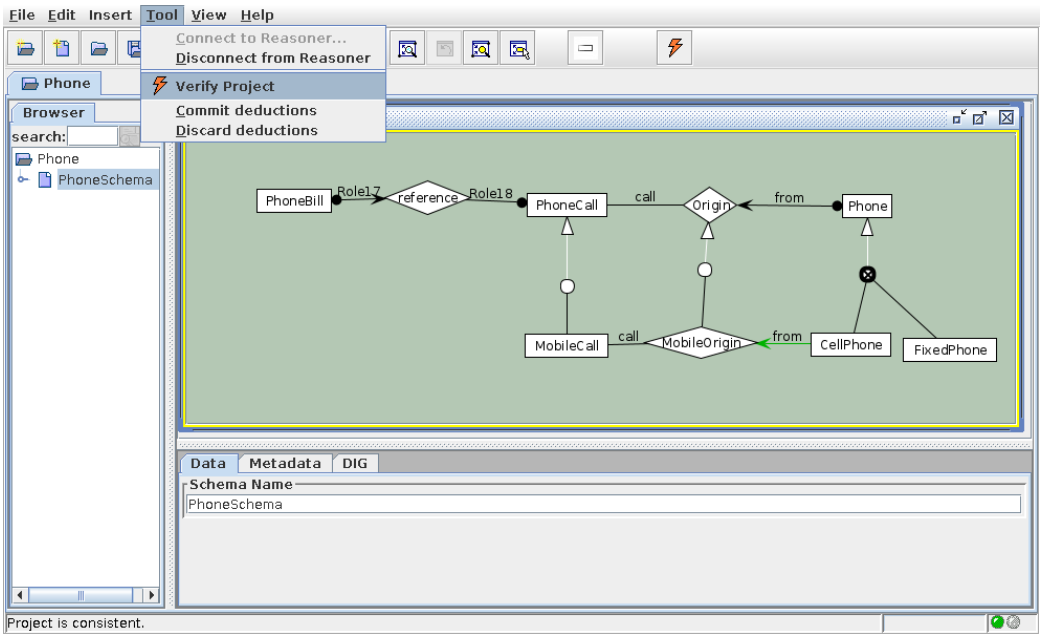


Figura 6.2: ICOM con una ontología de ejemplo.

al diagrama por el usuario o descartadas. Finalmente, ICOM soporta la definición de restricciones inter-modelos, es decir, entre dos o más modelos desde sus respectivos esquemas. Entre sus debilidades, la herramienta no exporta OWL 2 ni tampoco posee manejo de espacios de nombres para potenciar su uso en ingeniería ontológica.

A pesar de tener la posibilidad de utilizar servicios de razonamientos, el protocolo DIG, junto con sus limitaciones para modelar ontologías más expresivas, y las bibliotecas gráficas que utiliza, ya se encuentran discontinuados haciendo que estos aspectos críticos de la herramienta dificulten su continuidad en el desarrollo. *crowd* está desarrollada con tecnología web para poder ser utilizada por varias personas y promover así el uso colaborativo de la misma descartando las dificultades de instalación y puesta en marcha, diferencia sustancial con ICOM la cual está escrita en Java con la intención de que sea solamente una aplicación de escritorio.

6.2.5. eddy - Graphol

Graphol [CLSS14] es un lenguaje visual que permite representar ontologías por medio de una representación completamente gráfica evitando la escritura que requiere de sintaxis complejas. Actualmente, la expresividad Graphol $\mathcal{SROIQ}(\mathcal{D})$.

La herramienta ofrecida en su sitio web[5] se denomina Eddy y permite importar y exportar OWL 2 para ser graficado, y luego visualizar y editar este archivo por medio del lenguaje visual. La Figura 6.3 muestra al editor Eddy con una ontología de ejemplo.

Cabe destacar que se requiere la comprensión de un nuevo lenguaje puesto que no se provee soporte para los lenguajes estándares en la industria del desarrollo de software (i.e. UML, EER, etc.). Además, no posee ningún tipo de servicio de razonamiento para determinar la consistencia o resolver consultas. A pesar de que permite la incorporación de *plug-ins*, no se han encontrado ninguno que permita conectar la herramienta a tal servicio.

La posibilidad de utilizar OWL 2 como lenguaje permite la utilización de un razonador *a posteriori* si la ontología exportada es previamente encapsulada en un texto OWLlink junto con las consultas. Sin embargo, esto debe realizarse por separado de forma manual, escribiendo dicho texto y ejecutando el razonador.

6.2.6. OWLGrEd

El editor de ontologías OWLGrEd [COLS12] provee un entorno gráfico para OWL 2 utilizando una sintaxis gráfica de estilo similar al de los diagramas de clases UML. Dicha sintaxis consiste en un mapeo entre conceptos OWL que son similares a los de UML, por ejemplo: de clases de OWL a clases UML, de subclases OWL (`subclassOf`) a herencia UML, etc. Sin embargo,

[5] `http://www.dis.uniroma1.it/~graphol/`

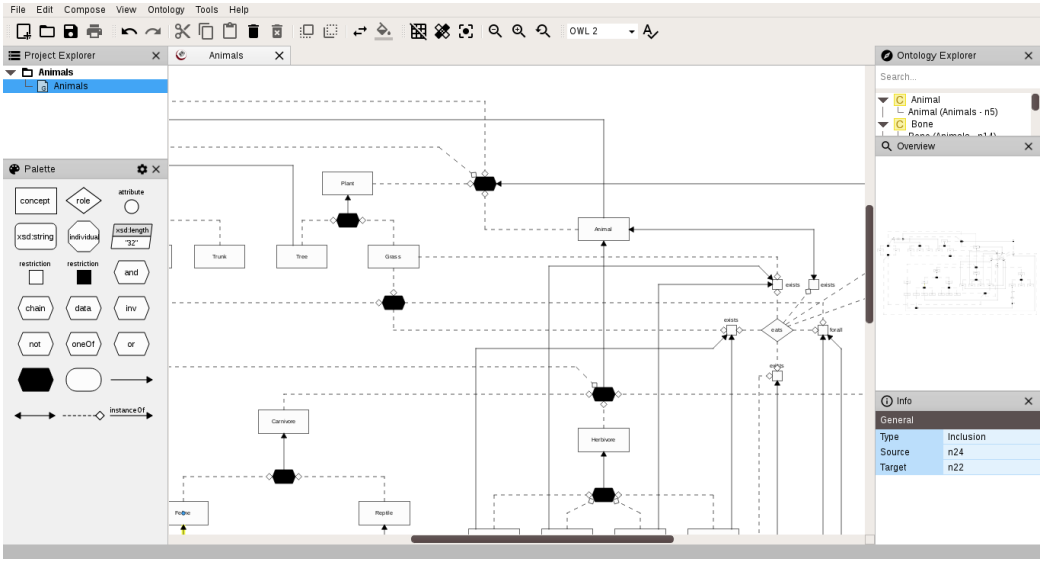


Figura 6.3: Editor gráfico Eddy mostrando una ontología en Graphol.

existen conceptos únicos en OWL por lo que también define una extensión de la notación UML con símbolos adicionales y expresiones textuales para abarcarlos. El entorno completo es capaz de mostrar ontologías OWL y/o RDF en el lenguaje descripto, además de editarlas bajo el mismo lenguaje y guardarlas en el formato con el que se cargó. Asimismo, otras ontologías pueden ser importadas como paquetes UML a un único proyecto.

El *plug-in* OWLGrEd permite interoperar entre Protégé y el entorno completo habilitando la capacidad de exportar el gráfico a esta última y viceversa. Sin embargo, dicho *plug-in* no soporta las últimas versiones de Protégé, limitándose a versiones previas a la 5.0.0 beta. En su versión online, disponible en `http://owlgred.lumii.lv/online_visualization`, sólo permite visualizar una ontología a partir de un archivo OWL, RDF, etc, aunque existen limitaciones de tamaño y sólo se observa la ontología directa y no las que se hacen referencia.

Finalmente, si bien OWLGrEd, ya sea en su versión de escritorio o Web, soportan un lenguaje basado en el UML estándar, no poseen ningún servicio de razonamiento integrado más allá de los provistos por Protégé, cuyas

posibles nuevas inferencias no son inmediatamente graficadas. Sin embargo, la utilización de un modelado gráfico similar al utilizado por la ingeniería de software hace a la herramienta más accesible, a pesar de considerar los aspectos propios de OWL que hacen a la extensión del lenguaje.

6.2.7. Menthor y OntoUML

OntoUML es una versión formal basada en patrones del lenguaje UML. Su metamodelo ha sido diseñado en conformidad con las distinciones ontológicas de una teoría bien fundada denominada Unified Foundational Ontology (UFO) propuesta en [Gui05].

Por su parte, Menthor [MSG+16] es un editor gráfico para OntoUML, que tiene la posibilidad de incluir reglas OCL (Object Constraint Language) [Gro14] al modelo y también de verificar la consistencia de los diagramas por medio de un razonador denominado Alloy Analizer. También, permite exportar el modelo a una ontología en varios formato, incluyendo OWL, y utilizando varias codificaciones denominadas "aproximaciones". Desde una perspectiva visual, Menthor permite generar los modelos de manera gráfica aunque el conjunto de funcionalidades para su manipulación es escueto. Sólo es dotado por zoom, alineación de modelos y algunos configuraciones particulares como por ejemplo para mostrar/esconder los atributos de sus clases.

Tanto *crowd* como Menthor utilizan lenguajes visuales y pueden generar una representación en DL de los modelos provistos por el usuario. Sin embargo, *crowd* se basa en lenguajes de modelado conceptual ampliamente conocidos por la ingeniería del software, como el UML, para el desarrollo de ontologías, evitando la necesidad de aprender un lenguaje nuevo. Asimismo, para la verificación de satisfacibilidad del modelo del usuario, el servicio de razonamiento de *crowd* utiliza el programa razonador RACER cuya expresividad se corresponde con la DL $\mathcal{SHIQ}$. Sin embargo, el razonador utilizado por Menthor, Alloy Analizer, está basado en un algoritmo capaz de resolver

SAT (Problema Booleano de Satisfacibilidad). Alloy Analizer recibe como entrada modelos escritos en Alloy [Jac12], un lenguaje para describir estructuras, basado en lógica e inspirado por el lenguaje para especificaciones de software Z [ASM80], requiriendo así, una transformación del modelo para su posterior verificación de satisfacibilidad.

6.2.8. VOWL y WebVOWL

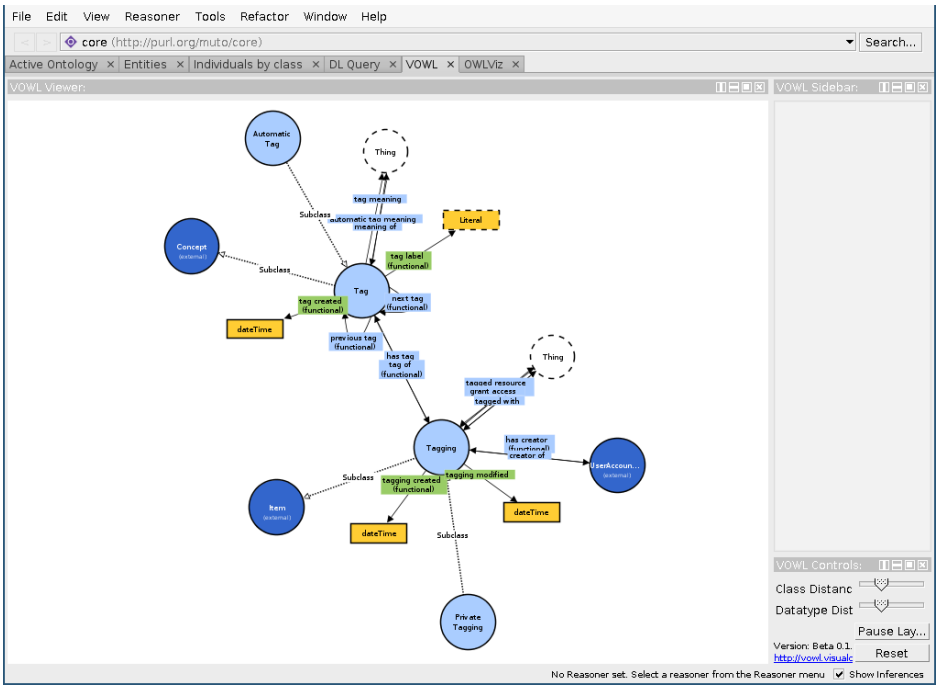


Figura 6.4: VOWL Protégé mostrando una ontología OWL.

VOWL ("Notación Visual para Ontologías OWL") [LNB14] es básicamente un lenguaje visual para la representación de ontologías orientado al usuario. Define símbolos gráficos para los elementos de OWL que son combinados con un grafo dirigido visualmente distribuido. Se enfoca en la visualización de los esquemas de ontología (TBox), asimismo incluye recomendaciones en cómo dibujar individuos y valores de datos correspondientes al ABox de la ontología. Este lenguaje fue implementado en dos herramientas, un *plug-in* para Protégé denominado ProtégéVOWL y una aplicación Web llamada

WebVOWL. Ambos realizan una representación visual, que no es editable en ese mismo lenguaje de la ontología que el usuario provee por medio de un archivo en formato OWL. La Figura 6.4 muestra la interfaz de ProtégéVOWL donde, de forma automática y animada, presenta las clases de la ontología MUTO[6] (Modular Unified Tagging Ontology).

A pesar de su facilidad de uso y de que distribuye las primitivas automáticamente en pantalla, existen grandes diferencias con *crowd*: Primeramente, es requisito estar familiarizado con el lenguaje OWL y las demás tecnologías de la Web Semántica para poder utilizarlo, puesto que utiliza un lenguaje visual que representa la DL y sus operadores, en vez de utilizar aquellos propios del dominio de la ingeniería de software. Segundo, tanto el *plug-in* como WebVOWL, no permiten la edición de las ontologías en el mismo lenguaje gráfico, sólo la visualización de las mismas. Tercero, no se provee ningún tipo de servicios de razonamiento en la interfaz web, y en cuanto al *plug-in*, sólo lo que posibilita la herramienta Protégé.

6.2.9. yEd y Graffo

Graffo [FGP+14] es una notación visual para ontologías OWL 2[7], dotada de elementos gráficos, distinguidos a través de colores, para facilitar el modelado y reducir el esfuerzo cognitivo de los modeladores.

Debido a que el esfuerzo se sus desarrolladores fue direccionado solamente a definir una notación y utilizar editores externos, los elementos de Graffo han sido implementados como una paleta de un editor llamado *yEd*[8]. Este hecho implica que ningún soporte de razonamiento lógico integrado es provisto. La herramienta también soporta la gestión de espacios de nombres, como una lista de pares (prefijo, URI) y, por lo tanto, cada entidad del modelos puede ser definida por su prefijo o URI.

[6] `http://purl.org/muto/core#`
[7] `http://www.essepuntato.it/static/graffoo/specification/current.html`
[8] `https://www.yworks.com/products/yed`

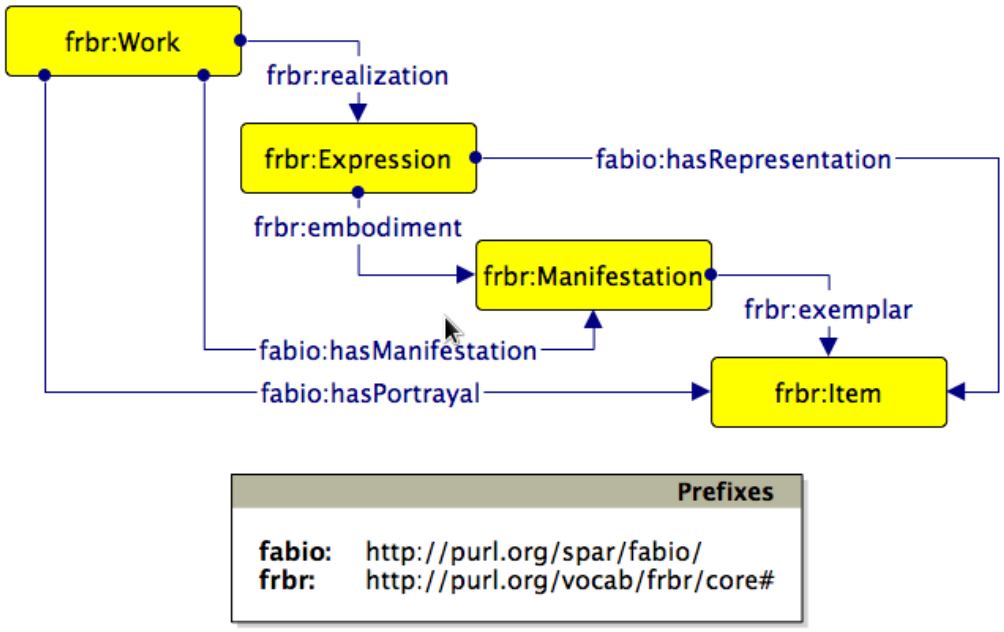


Figura 6.5: Modelo Graffo de FaBiO para descripción de entidades bibliográficas.

Finalmente, un aspecto positivo de este enfoque es facilitado por el editor *yEd*, ya que es una poderosa aplicación de escritorio usada para generar diagramas de manera efectiva y rápida. Para esto, provee algoritmos avanzados de layout automático, además de otras complementarias tales como zoom, selección, búsqueda y selección de partes específicas de un diagrama a partir de una vista global del modelo. La Figura 6.5 ilustra un modelo Graffo para la ontología FaBiO[9].

6.3. Conclusiones

Diversas herramientas relacionadas a la Web Semántica y en particular, a la ingeniería ontológica pueden encontrarse en la literatura actualizada, donde cada una cubre requerimientos y tareas específicas. Durante el desarrollo de este capítulo, realizamos una revisión y una descripción de las caracterísi-

[9] `http://www.sparontologies.net/ontologies/fabio`

cas principales de tales herramientas. Para esto, tomamos como referencia el proceso de visualización presentados en este trabajo y los requerimientos definidos para *crowd*. Los resultados de este análisis y evaluación son contundentes: ninguna de estas herramientas ofrece soporte interactivo y web, e integración con servicios de razonamiento lógico en el mismo ambiente de ingeniería.

Entre las herramientas recopiladas, Protégé y WebProtégé son las más referencias por diferentes autores, aunque su amplia aceptación no justifica la falta de un soporte visual más expresivo, dificultando la legibilidad y comprensión de grandes ontologías. En otra dirección, herramientas más recientes tales como eddy, VOWL y OWLGrEd, presentan interfaces visuales basadas en grafos o UML, pero la interacción con razonadores y aún limitada o ausente.

Nuevas herramientas están en desarrollo y/o serán propuestas para dar soporte a metodologías de ingeniería ontológica, sin embargo, es importe remarcar la necesidad de mantener la retroalimentación cruzada entre los sistemas visuales para representar conocimiento y los formalismos lógicos. Tal retroalimentación es por demás productiva y ha ido en aumento desde el momento en el que las DLs emergieron como una manera estructurada y bien definida para proveer caracterizaciones semánticas de *frames* y redes semánticas [NB03].

Capítulo 7

Conclusiones y Trabajos Futuros

Las preguntas de investigación estaban centradas en la integración de los sistemas de representación de conocimiento basados en lógica y los visuales en el contexto de herramientas software concretas. En particular, la integración de ontologías, equivalentes a una base de conocimiento lógica, y lenguajes visuales de modelado para dar soporte a la ingeniería ontológica.

Este capítulo examina nuevamente las preguntas de investigación y discute cómo ellas han sido abordadas. Asimismo, también presenta las contribuciones principales logradas durante el análisis de cada una de ellas. Finalmente, se provee una lista de publicaciones en congresos y revistas científicas, y un detalle de futuras direcciones de esta investigación.

7.1. Discusión de Preguntas de Investigación

La discusión principal de esta investigación abordó dos enfoques claramente definidos al momento de representar conocimiento de un dominio de aplicación puntual: los sistemas basados en interfaces visuales (redes semánticas) y los sistemas basados en lógica. Los primeros son considerados más

efectivos, sin embargo, la ausencia de una semántica bien definida introduce ambigüedades en la caracterización de dominios reales de aplicación.

En lo que respecta a sistemas lógicos, las DLs han realizado un significativo aporte a las Ciencias de la Computación y, en particular a la representación de conocimiento, para ofrecer una semántica precisa a las redes semánticas gráficas. El poder expresivo de las DLs puede aún ser más explotado, para lo cual es fundamental ofrecer ambientes intuitivos para manipular las bases de conocimiento, las cuales llamamos de manera equivalente *ontologías*. Sin embargo, estos avances no fueron acompañados de manera coherente por sistemas visuales que permitan esta explotación. Por lo tanto, el objetivo de esta investigación fue responder la siguiente pregunta principal.

- **(PI)** ¿Cómo la formalización de ontologías y los lenguajes visuales pueden ser integrados en una herramienta para ingeniería ontológica?

A su vez, está pregunta fue subdividida en cuatro preguntas relacionadas, las cuales detallaremos y direccionaremos mediante las contribuciones de esta Tesis.

- **(PI1)** ¿Cómo integrar efectivamente la formalización de ontologías en las herramientas de ingeniería ontológica?
- **(PI2)** ¿Cómo puede aprovecharse el razonamiento automático en los procesos de ingeniería ontológica?
- **(PI3)** ¿Cómo influyen los lenguajes visuales de ontologías en herramientas de la ingeniería ontológica?
- **(PI4)** ¿Cuáles son los requerimientos principales que una herramienta visual, para tareas de ingeniería ontológica, debe satisfacer?

Las preguntas **(PI1)** y **(PI2)** fueron abordadas, en el capítulo 3, a partir de la definición de dos conceptos teóricos complementarios: un *Proceso de*

Visualización de Conocimiento basado en Ontología y un *Sistema de Manipulación de Ontologías Gráficas* (*GOMS*). Ambos presentan lineamientos base para la definición de herramientas concretas que incorporen aspectos visuales y lógicos en un mismo contexto, aprovechando el potencial de las DLs, y en particular el razonamiento automático, para mejorar las tareas de ingeniería ontológica.

Un ejemplo concreto del aprovechamiento del razonamiento automático fue presentado en el ejemplo 3.3.3 para la manipulación de cardinalidades en asociaciones binarias. Este algoritmo se basa en la generación de axiomas lógicos equivalentes a las clases involucradas en una asociación binaria, para luego consultar la ontología con un razonador lógico y revisar las restricciones de cardinalidad para cada nuevo axioma participando en la asociación. En este sentido, restricciones de cardinalidad más estrictas pueden ser inferidas y visualizadas para valores máximos y mínimos $1 \leq n \leq 9$.

El proceso de visualización es una adaptación de un proceso genérico presentado en [War04], en el cual se combinan los estados de recolección de datos, su preprocesamiento, la producción de una visualización y finalmente, de percepción humana. En particular, ambos estados de procesamiento y de producción de visualizaciones están definidos por medio de nuestro *GOMS*.

Un *GOMS* está basado en la definición de mapeos que habilitan la transformación de modelos gráficos a representaciones lógicas, así como también de formalismos lógicos a modelos visuales. Los modelos generados a partir de un *GOMS* y que cumplen con los requerimientos impuestos por este sistema, son lo que llamamos *ontologías gráficas*. Diferentes formas de procesar estos datos pueden ser definidas a través de un *GOMS*, con un objetivo común: explorar y extraer el conocimiento a partir del razonamiento lógico sobre modelos conceptuales. Resultados de estos trabajos han sido presentados en [BCF15, BGCF16, BGCF17].

Las contribuciones teóricas inducen el desarrollo de nuevas tecnologías para ser usados en herramientas más efectivas. Esto lleva a una interacción

cercana entre los aportes básicos y los aplicados. En este sentido, la siguiente contribución está relacionada a la concreción de una arquitectura para ambientes de ingeniería ontológica. Esto requirió de la identificación de requerimientos específicos derivados, principalmente, del *GOMS*, y que fueron introducidos durante el desarrollo de cada uno de los capítulos. Cada requerimiento debidamente justificado, nos permitió arribar a la conclusión de la pregunta **(PI4)**.

A partir de las formalización presentada, se planteó y se documentó una arquitectura de referencia para ambientes web de ingeniería de ontologías gráficas, donde se estudiaron diversas vistas complementarias de módulos (descomposición, usos y capas), componentes y conectores (llamada-retorno), y asignación (despliegue e instalación). El reporte de este diseño fue presentado en [BEF18] y ha sido validado también en trabajos relacionados, demostrando su flexibilidad y extensibilidad para la definición de consultas gráficas [GBCF18] y validación de modelos de variabilidad OVM [OBCF18].

Finalmente, para poder direccionar la pregunta **(PI3)** y dar un implementación concreta, se presentó una herramienta web para la manipulación de ontologías gráficas, llamada *crowd*, junto con un editor visual UML e interfaces para interacción con razonadores lógicos. El ambiente desarrollado también incluye la gestión de espacios de nombres, interoperabilidad por medio de la exportación OWL 2 y la generación automática de documentación de los modelos. La primer versión oficial de *crowd* está disponible en `http://crowd.fi.uncoma.edu.ar/`.

Para su validación, se realizaron dos evaluaciones. La primera basada en usuarios, por medio de una encuesta sobre el uso de la herramienta para tareas de modelado. A partir de este estudio, se obtuvo un factor de usabilidad del 65 % (en la escala SUS). Si bien, mediante esta evaluación, podemos concluir sobre un nivel de usabilidad aceptable en cuanto a la manipulación visual de ontologías, es necesario continuar con la profundización de dichas evaluaciones sobre *crowd*.

Entre los aspectos positivos, los participantes concluyeron sobre los beneficios de tener una herramienta web y visual que permita validar modelos semánticos, mientras que también reportaron la necesidad de complementar las funcionalidades de la interfaz gráfica para hacer aún más amigable su uso: menús contextuales usando doble-click y un mejor reporte de los resultados del razonamiento. La encuesta utilizada para este trabajo se encuentra aún en linea `https://sites.google.com/a/fi.uncoma.edu.ar/german-braun/crowd-survey`, para continuar con un proceso de evaluación permanente.

La segunda evaluación es teórica y en base a la instanciación de *crowd* como un sistema *WYSIWYM* para la visualización y gestión de contenido semántico. Para esta última evaluación se identificaron y mapearon técnicas de exploración, visualización y creación con los respectivos elementos gráficos de *crowd* y se detectaron aquellos mapeos parciales o ausentes.

Por último, se realizó una comparación con herramientas relevadas de la literatura y que reforzaron nuestras conclusiones. En particular, ninguna de las herramientas relevadas definen una integración coherente de aspectos visuales y formales para mejorar las tareas de ingeniería ontológica. Asimismo, la influencia de los lenguajes visuales en la ingeniería ontológica es aún un aspecto a profundizar, debido a que no se han alcanzado estándares o consensos sobre notaciones visuales para ontológicas.

7.2. Trabajos Publicados

1. G. Braun, E. Estevez y P. Fillottrani. *A Reference Architecture for Ontology Engineering Web Environments*. Journal of Computer Science & Technology, 2018. In Press.

2. C. Gimenez, G. Braun, L. Cecchi y P. Fillottrani. *Towards a Visual SPARQL-DL Query Builder*. In Proc. of the Congreso Argentino de Ciencias de la Computación (CACIC), 2018

3. A. Oyarzún, G. Braun, L. Cecchi y P. Fillottrani. *A Graphical Web Tool with DL-based Reasoning*. In Proc. of the Congreso Argentino de Ciencias de la Computación (CACIC), 2018.

4. G. Braun y L. Cecchi. *El Valor de los Datos, entre ventajas y responsabilidades*. Revista COMAHUE Nuestra Región N 5. Edición FUNYDER. Universidad Nacional del Comahue. ISSN 2591-300X

5. M. Zárate, G. Braun and P. Fillottrani. *Adding Biodiversity Datasets from Argentinian Patagonia to the Web of Data*. In Proceeding of the 16th International Semantic Web Conference (ISWC 2017) - Poster&Demos Track

6. M. Zárate, G. Braun and P. Fillottrani. *Adding Biodiversity Datasets from Argentinian Patagonia to the Web of Data*. In Proceeding of the 2nd International Workshop on Semantics for Biodiversity (S4BioDiv) co-located with 16th International Semantic Web Conference (ISWC 2017)

7. G. Braun, C. Gimenez, L. Cecchi and P. Fillottrani. *Towards a Visualisation Process for Ontology-Based Conceptual Modelling (Abstract)*. In the International Workshop on Description Logics 2017. Montpellier, Francia

8. G. Braun, M. Pol'la, A. Buccella, L. Cecchi, P. Fillottrani and A. Cechich. *A DL Semantics for Reasoning over OVM-based Variability Models*. In Proceeding of the International Workshop on Description Logics 2017. Montpellier, Francia

9. G. Braun, C. Gimenez, L. Cecchi and P. Fillottrani. *Towards Conceptual Modelling Interoperability in a Web Tool for Ontology Engineering*. In Proceeding of the III Simposio Argentino de Ontologías y sus Aplicaciones (SAOA 2017), JAIIO 2017

10. C. Gimenez, G. Braun, L. Cecchi and P. Fillottrani. *crowd: A Tool for Conceptual Modelling assisted by Automated Reasoning - Preliminary*

Report. In Proceedings of the II Simposio Argentino de Ontologías y sus Aplicaciones (SAOA 2016), JAIIO 2016

11. C. Gimenez, G. Braun, L. Cecchi and P. Fillottrani. *crowd: A Tool for Conceptual Modelling assisted by Automated Reasoning - Preliminary Report*. In Proceedings of the II Simposio Argentino de Ontologías y sus Aplicaciones (SAOA 2016), JAIIO 2016

12. G. Michelan, G. Braun, L. Cecchi and P. Fillottrani. *Integration of Scientific Information through Linked Data - Preliminary Report*. In Proceedings of the II Simposio Argentino de Ontologías y sus Aplicaciones (SAOA 2016), JAIIO 2016

13. G. Braun. *Methodologies and Tools for Conceptual Modelling with Reasoning Support*. In Doctoral Consortium at 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016)

14. G. Braun, L. Cecchi and P. Fillottrani. *Integrating Graphical Support with Reasoning in a Methodology for Ontology Evolution*. In Proceedings of the Ninth International Workshop on Modular Ontologies (WoMO 2015), IJCAI 2015

15. G. Braun and L. Cecchi. *Extension Rules for Ontology Evolution with a Conceptual Modelling Tool*. In Proceedings of the I Simposio Argentino de Ontologías y sus Aplicaciones (SAOA 2015), JAIIO 2015

7.3. Trabajo Futuro

A partir de la investigación presentada en esta tesis se abren las siguientes líneas de investigación.

En primer lugar, el desafío de enriquecer el *GOMS*, mediante la definición de múltiples mapeos, fortaleciendo la interacción gráfica y lógica para extraer más conocimiento de los modelos ontológicos. En este sentido, algunos trabajos iniciales han sido reportados de manera teórica [BC15], utilizando el

concepto de patrones ontológicos en el contexto de nuestra herramienta, mediante la aplicación de reglas de extensión. Para esto, se diseñó un catalogo de reglas y se validaron de manera teórica. Si bien, las sugerencias aportadas por estas reglas no son consecuencias lógicas del modelos, ellas pueden determinar ciertos patrones presentes usando servicios de razonamiento lógicos.

Por otro lado, resultados preliminares del uso de la arquitectura de referencia para manejo de múltiples lenguajes de modelado gráfico, fueron presentados en [BGFC17]. Avanzar sobre la concreción de esta funcionalidad y en el soporte de otros lenguajes como ORM 2, junto con sus mapeos lógicos, completaría el desarrollo de *crowd*.

Otros dos últimos tópicos interesantes para futuras investigaciones están referidos al uso de este ambiente en los procesos de extracción, revisión e integración de Grandes Datos [Ce18, CZ14] y en el acceso de datos basado en ontologías [XCK+18].

En el primero, se propone la generación de nuevos principios y herramientas formales para la gestión semántica de datos en las etapas iniciales del ciclo de Big Data, aplicando razonamiento eficiente en ontologías generadas a partir de modelos de datos incompletos y parciales. En este contexto, las ontologías facilitan la integración semántica de los datos y la extracción de conocimiento desde ellos, ocultando en ocasiones, las formas de almacenamiento de los datos crudos.

Por último, el acceso de datos basado en ontologías (OBDA) es un paradigma semántico para acceso a datos almacenados en datos relacionales. En este sentido, OBDA ha sido ampliamente desarrollado y aplicado en la industria [GGH+13], sin embargo, es necesario proveer interfaces gráficas para poder crear y mantener los modelos ontológicos y los mapeos que requiere este framework.

Apéndice A

Transformaciones Gráfico - Lógicas en *crowd v1.0*

En este apéndice, adjuntamos las transformaciones Ψ_{OWL2}^{UML} y Θ_{UML}^{OWL2} implementadas en la actual versión de *crowd*.

A.1. $\Psi_{OWL2}^{UML} : OWL\ 2 \rightarrow UML$

```
// Consultas SPARQL-DL para extraer axiomas desde un documento OWL 2

// Obtener todas las clases
SELECT ?class
WHERE {
    Class(?class)
}

// Obtener todas las OWL 2 ObjectProperties
SELECT ?objectproperty
WHERE {
    ObjectProperty(?objectproperty)
}

// Obtener Dominios para cada ObjectProperty
SELECT DISTINCT ?objectproperty ?domainop
WHERE {
```

```
    ObjectProperty(?objectproperty),
    Domain(?objectproperty,?domainop)
},

// Obtener Rangos para cada ObjectProperty
SELECT DISTINCT ?objectproperty ?rangeop
WHERE {
    ObjectProperty(?objectproperty),
    Range(?objectproperty,?rangeop)
},

// Obtener Subclases directas y estrictas para cada Clase
SELECT DISTINCT ?strictsub ?strictsupclass
WHERE {
    DirectSubClassOf(?strictsub,?strictsupclass),
    StrictSubClassOf(?strictsub,?strictsupclass)
},

// Obtener Subclases directas para cada Clase
SELECT DISTINCT ?directsub ?supclass
WHERE {
    DirectSubClassOf(?directsub,?supclass)
},

// Obtener Sub ObjectProperties directas y
   estrictas para cada ObjectProperty
SELECT DISTINCT ?subobjectproperty ?strictsupobjectproperty
WHERE {
    DirectSubPropertyOf(?subobjectproperty,?strictsupobjectproperty),
    StrictSubPropertyOf(?subobjectproperty,?strictsupobjectproperty)
},

// Obtener Sub ObjectProperties directas para cada ObjectProperty
SELECT DISTINCT ?subobjectproperty ?supobjectproperty
WHERE {
    DirectSubPropertyOf(?subobjectproperty,?supobjectproperty)
},

// Obtener clases equivalentes
SELECT DISTINCT ?classeq1 ?classeq
WHERE {
    EquivalentClass(?classeq1,?classeq)
},
```

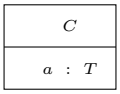
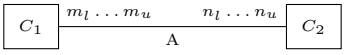
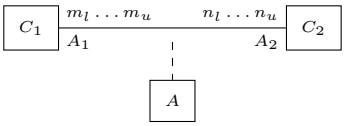
```
// Obtener Sub ObjectProperties equivalentes
SELECT DISTINCT ?objectpropertyeq1 ?objectpropertyeq
WHERE {
    EquivalentProperty(?objectpropertyeq1,?objectpropertyeq)
},

// Obtener todas las DataProperties
SELECT DISTINCT ?dataproperty
WHERE {
    DataProperty(?dataproperty)
},

// Obtener Dominios para cada DataProperty
SELECT DISTINCT ?dataproperty ?domaindp
WHERE {
    DataProperty(?dataproperty),
    Domain(?dataproperty,?domaindp)
},

// Obtener Rangos para cada DataProperty
SELECT DISTINCT ?dataproperty ?rangedp
WHERE {
    DataProperty(?dataproperty),
    Range(?dataproperty,?rangedp)
},
```

A.2. Θ_{UML}^{OWL2} : $UML \rightarrow OWL\ 2$

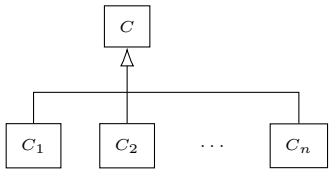
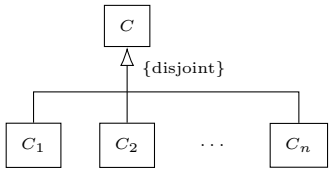
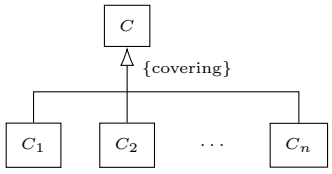
A.3. Ontología para el ejemplo 3.3.3

En esta sección mostramos la ontología $\mathcal{O}$ para el ejemplo 3.3.3 en sintaxis OWLlink, donde se detalla cada uno de los conceptos asertados para la manipulación de cardinalidades. Este mapeo es extraido de la versión actual de *crowd*.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Primitiva	UML	$\mathcal{ALCQI}$
Tipo de Objeto		C
Atributo		$\exists a.\top \sqsubseteq C$ $\exists a^-.\top \sqsubseteq T$ $C \sqsubseteq (\leq 1\, a.\top)$
Relación binaria sin clase		$\exists A.\top \sqsubseteq C_1$ $\exists A^-.\top \sqsubseteq C_2$ $C_1 \sqsubseteq (\geq n_l\, A.\top) \sqcap (\leq n_u\, A.\top)$ $C_2 \sqsubseteq (\geq m_l\, A^-.\top) \sqcap (\leq m_u\, A^-.\top)$ $C_1\_A_{min_1} \equiv C_1 \sqcap (\geq n_1\, A)$ $C_1\_A_{min_l} \equiv C_1 \sqcap (\geq n_l\, A)$ $C_1\_A_{max_1} \equiv C_1 \sqcap (\leq n_1\, A)$ $C_1\_A_{max_u} \equiv C_1 \sqcap (\leq n_u\, A)$ $C_2\_A_{min_1} \equiv C_2 \sqcap (\geq m_1\, A^-)$ $C_2\_A_{min_l} \equiv C_2 \sqcap (\geq m_l\, A^-)$ $C_2\_A_{max_1} \equiv C_2 \sqcap (\leq m_1\, A^-)$ $C_2\_A_{max_u} \equiv C_2 \sqcap (\leq m_u\, A^-)$
Relación binaria con clase		$\exists A_1.\top \sqsubseteq A$ $\exists A_1^-.\top \sqsubseteq C_1$ $\exists A_2.\top \sqsubseteq A$ $\exists A_2^-.\top \sqsubseteq C_2$ $A \sqsubseteq \exists A_1 \sqcap (\leq 1\, A_1) \sqcap \exists A_2 \sqcap (\leq 1\, A_2)$ $C_1 \sqsubseteq (\geq n_l A_1^-) \sqcap (\leq n_u A_1^-)$ $C_2 \sqsubseteq (\geq m_l A_2^-) \sqcap (\leq m_u A_2^-)$ $C_1\_A_{1\_min_1} \equiv C_1 \sqcap (\geq n_1\, A_1^-)$ $C_1\_A_{1\_min_l} \equiv C_1 \sqcap (\geq n_l\, A_1^-)$ $C_1\_A_{1\_max_1} \equiv C_1 \sqcap (\leq n_1\, A_1^-)$ $C_1\_A_{1\_max_u} \equiv C_1 \sqcap (\leq n_u\, A_1^-)$ $C_2\_A_{2\_min_1} \equiv C_2 \sqcap (\geq m_1\, A_2^-)$ $C_2\_A_{2\_min_l} \equiv C_2 \sqcap (\geq m_l\, A_2^-)$ $C_2\_A_{2\_max_1} \equiv C_2 \sqcap (\leq m_1\, A_2^-)$ $C_2\_A_{2\_max_u} \equiv C_2 \sqcap (\leq m_u\, A_2^-)$

Cuadro A.1: Mapeos Gráficos-Lógicos de primitivas UML en $\mathcal{ALCQI}$ y transformación para cardinalidades $n, m \geq 1$ y $n, m \leq 9$, basados en [BCD05].

Primitiva	UML	$\mathcal{ALCQI}$
Subclasificación		$C_1 \sqsubseteq C$ $C_2 \sqsubseteq C$ ... $C_n \sqsubseteq C$ $C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n \sqsubseteq C$
Subclasificación disjunta		$C_i \sqsubseteq (\sqcap_{j=i+1}^{n} \neg C_j)\ \ 1 \leq i \leq n-1$
Subclasificación disjunta		$C \sqsubseteq (\sqcup_{i=1}^{n} C_i)$

Cuadro A.2: Mapeos Gráficos-Lógicos de primitivas UML en $\mathcal{ALCQI}$ para relaciones de subclasificación basados en [BCD05].

```
<RequestMessage xmlns="http://www.owllink.org/owllink#" xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.owllink.org
     /owllink# http://www.owllink.org/owllink-20091116.xsd" xml:base="http://crowd.fi.uncoma.edu.ar#"
     >
   <CreateKB kb="http://crowd.fi.uncoma.edu.ar#">
      <Prefix name="rdf" fullIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
      <Prefix name="rdfs" fullIRI="http://www.w3.org/2000/01/rdf-schema#" />
      <Prefix name="xsd" fullIRI="http://www.w3.org/2001/XMLSchema#" />
      <Prefix name="owl" fullIRI="http://www.w3.org/2002/07/owl#" />
      <Prefix name="crowd" fullIRI="http://crowd.fi.uncoma.edu.ar#" />
   </CreateKB>
   <Set kb="http://crowd.fi.uncoma.edu.ar#" key="abbreviatesIRIs">
      <Literal>false</Literal>
   </Set>
   <Tell kb="http://crowd.fi.uncoma.edu.ar#">
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
         <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
         <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
      </owl:SubClassOf>
      <owl:SubObjectPropertyOf>
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         <owl:ObjectProperty IRI="http://www.w3.org/2002/07/owl#topObjectProperty" />
      </owl:SubObjectPropertyOf>
      <owl:SubObjectPropertyOf>
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         <owl:ObjectProperty IRI="http://www.w3.org/2002/07/owl#topObjectProperty" />
      </owl:SubObjectPropertyOf>
      <owl:SubClassOf>
         <owl:ObjectSomeValuesFrom>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
            <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
         </owl:ObjectSomeValuesFrom>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:ObjectSomeValuesFrom>
            <owl:ObjectInverseOf>
               <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
            </owl:ObjectInverseOf>
            <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
         </owl:ObjectSomeValuesFrom>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMinCardinality cardinality="1">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMinCardinality>
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="1">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:SubClassOf>
      <owl:EquivalentClasses>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
              origin_min_1" />
         <owl:ObjectIntersectionOf>
            <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
            <owl:ObjectMinCardinality cardinality="1">
               <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
            </owl:ObjectMinCardinality>
         </owl:ObjectIntersectionOf>
      </owl:EquivalentClasses>
      <owl:EquivalentClasses>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
              origin_max_1" />
         <owl:ObjectIntersectionOf>
            <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
            <owl:ObjectMaxCardinality cardinality="1">
```

```
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_2" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="2">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_3" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="3">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_4" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="4">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_5" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="5">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_6" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="6">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_7" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="7">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_8" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="8">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/PhoneCall_http://crowd.fi.uncoma.edu.ar/
           origin_max_9" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
         <owl:ObjectMaxCardinality cardinality="9">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
```

```
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_1" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="1">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_2" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="2">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_3" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="3">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_4" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="4">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_5" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="5">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_6" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="6">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_7" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="7">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
```

```
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_8" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="8">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_min_9" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMinCardinality cardinality="9">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_1" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="1">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_2" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="2">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_3" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="3">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_4" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="4">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_5" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="5">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
```

```
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_6" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="6">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_7" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="7">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_8" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="8">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Phone_http://crowd.fi.uncoma.edu.ar/
        origin_max_9" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      <owl:ObjectMaxCardinality cardinality="9">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:SubClassOf>
   <owl:ObjectSomeValuesFrom>
      <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
   </owl:ObjectSomeValuesFrom>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
</owl:SubClassOf>
<owl:SubClassOf>
   <owl:ObjectSomeValuesFrom>
      <owl:ObjectInverseOf>
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectInverseOf>
      <owl:Class IRI="http://www.w3.org/2002/07/owl#Thing" />
   </owl:ObjectSomeValuesFrom>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
</owl:SubClassOf>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_min_1" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMinCardinality cardinality="1">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_min_2" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMinCardinality cardinality="2">
```

```
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_3" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="3">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_4" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="4">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_5" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="5">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_6" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="6">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_7" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="7">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_8" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="8">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_min_9" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMinCardinality cardinality="9">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMinCardinality>
      </owl:ObjectIntersectionOf>
   </owl:EquivalentClasses>
   <owl:EquivalentClasses>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
           morigin_max_1" />
      <owl:ObjectIntersectionOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:ObjectMaxCardinality cardinality="1">
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectMaxCardinality>
      </owl:ObjectIntersectionOf>
```

```
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_2" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="2">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_3" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="3">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_4" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="4">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_5" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="5">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_6" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="6">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_7" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="7">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_8" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="8">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/MobileCall_http://crowd.fi.uncoma.edu.ar/
        morigin_max_9" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
      <owl:ObjectMaxCardinality cardinality="9">
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
```

```
        morigin_min_1" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="1">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_2" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="2">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_3" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="3">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_4" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="4">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_5" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="5">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_6" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="6">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_7" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="7">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
```

```
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_8" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="8">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_min_9" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMinCardinality cardinality="9">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMinCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_max_1" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMaxCardinality cardinality="1">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_max_2" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMaxCardinality cardinality="2">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_max_3" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMaxCardinality cardinality="3">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_max_4" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMaxCardinality cardinality="4">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
<owl:EquivalentClasses>
   <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
        morigin_max_5" />
   <owl:ObjectIntersectionOf>
      <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
      <owl:ObjectMaxCardinality cardinality="5">
         <owl:ObjectInverseOf>
            <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         </owl:ObjectInverseOf>
      </owl:ObjectMaxCardinality>
   </owl:ObjectIntersectionOf>
</owl:EquivalentClasses>
```

```
      <owl:EquivalentClasses>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
              morigin_max_6" />
         <owl:ObjectIntersectionOf>
            <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
            <owl:ObjectMaxCardinality cardinality="6">
               <owl:ObjectInverseOf>
                  <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
               </owl:ObjectInverseOf>
            </owl:ObjectMaxCardinality>
         </owl:ObjectIntersectionOf>
      </owl:EquivalentClasses>
      <owl:EquivalentClasses>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
              morigin_max_7" />
         <owl:ObjectIntersectionOf>
            <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
            <owl:ObjectMaxCardinality cardinality="7">
               <owl:ObjectInverseOf>
                  <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
               </owl:ObjectInverseOf>
            </owl:ObjectMaxCardinality>
         </owl:ObjectIntersectionOf>
      </owl:EquivalentClasses>
      <owl:EquivalentClasses>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
              morigin_max_8" />
         <owl:ObjectIntersectionOf>
            <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
            <owl:ObjectMaxCardinality cardinality="8">
               <owl:ObjectInverseOf>
                  <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
               </owl:ObjectInverseOf>
            </owl:ObjectMaxCardinality>
         </owl:ObjectIntersectionOf>
      </owl:EquivalentClasses>
      <owl:EquivalentClasses>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar/Cell_http://crowd.fi.uncoma.edu.ar/
              morigin_max_9" />
         <owl:ObjectIntersectionOf>
            <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
            <owl:ObjectMaxCardinality cardinality="9">
               <owl:ObjectInverseOf>
                  <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
               </owl:ObjectInverseOf>
            </owl:ObjectMaxCardinality>
         </owl:ObjectIntersectionOf>
      </owl:EquivalentClasses>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#MobileCall" />
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#PhoneCall" />
      </owl:SubClassOf>
      <owl:SubClassOf>
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Cell" />
         <owl:Class IRI="http://crowd.fi.uncoma.edu.ar#Phone" />
      </owl:SubClassOf>
      <owl:SubObjectPropertyOf>
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#morigin" />
         <owl:ObjectProperty IRI="http://crowd.fi.uncoma.edu.ar#origin" />
      </owl:SubObjectPropertyOf>
   </Tell>
</RequestMessage>
```

Apéndice B

Materiales para Evaluación

En este apéndice, adjuntamos el cálculo del índice SUS y la encuesta realizada para la evaluación de *crowd*.

B.1. Cálculo SUS

La tabla B.1 muestra las estadísticas finales y el cálculo de índice SUS para *crowd*. Cada fila de dicha tabla corresponde a las respuestas dadas por un usuario participante del experimento.

Las columnas Q1..Q10 representan las preguntas incluidas en el cuestionario, mientras que las columnas notadas con V1..V10 corresponde al valor calculado para dicha pregunta según la ponderación (1) "Muy en desacuerdo", (2) "En desacuerdo", (3) "Neutral", (4) "De acuerdo", (5) "Completamente de acuerdo".

Cada celda V_i se calcula restando 1 al valor escalar de cada item positivo (impares Q1, Q3, Q5, Q7, Q9). Por ejemplo, para Q1 y respuesta "De acuerdo", el valor será 3. Por otro lado, la contribución de los items negativos (pares Q2, Q4, Q6, Q8, Q10), se calcula como 5 menos el valor escalar. Por ejemplo, para Q2 y respuesta "Muy en desacuerdo", el valor será 4.

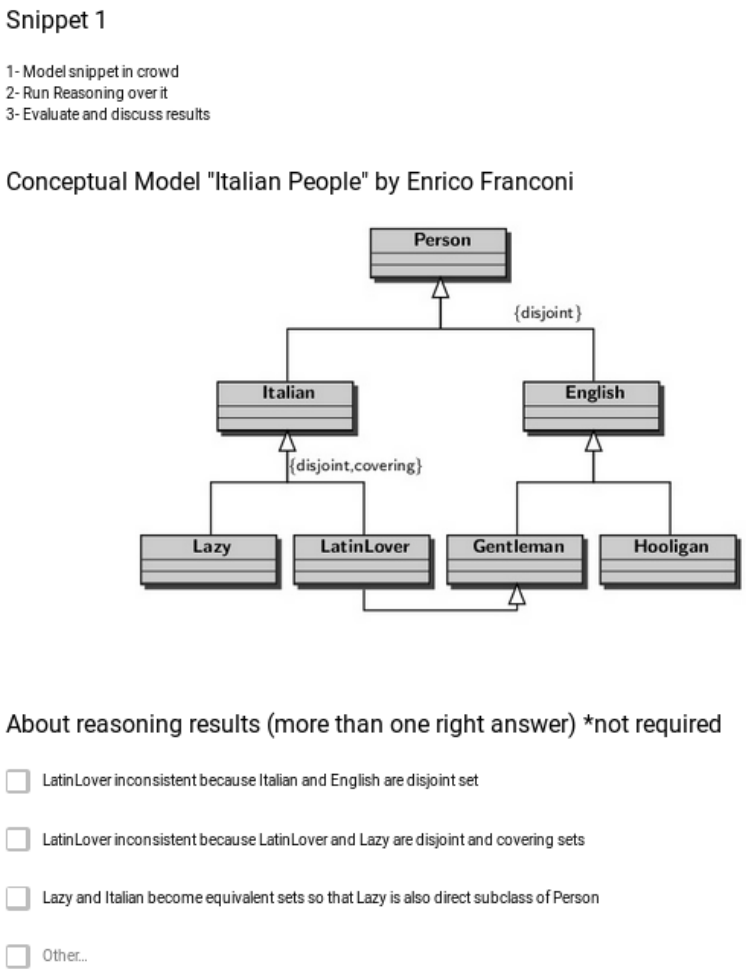


Figura B.1: Ejercicio de Modelado en *crowd*.

El valor I de cada fila corresponde a la suma de los valores V_i y el valor global de cada respuesta G se calcula multiplicando I por 2,5 (definido en la escala). Finalmente, el factor SUS para la evaluación es calculado como el promedio de los factores parciales.

B.2. Encuesta a Usuarios

Cuadro B.1: Valores SUS para cada pregunta y los respectivos cálculos para la contribución al factor final. SD = "Muy en desacuerdo". D = "En desacuerdo". N = "Neutral". A = "De acuerdo". SA = "Completamente de acuerdo".

Q1	V1	Q2	V2	Q3	V3	Q4	V4	Q5	V5	Q6	V6	Q7	V7	Q8	V8	Q9	V9	Q10	V10	I	G
A	3	SD	4	N	2	D	3	A	3	A	1	A	3	A	1	D	1	D	3	24	60 %
SD	4	SD	4	A	3	A	1	SA	4	SA	4	SA	4	SA	0	SA	4	SD	4	32	80 %
A	3	D	3	N	2	A	1	A	3	D	3	A	3	D	3	N	2	N	2	25	62,5 %
A	3	D	3	A	3	D	3	N	2	D	3	A	3	A	1	N	2	D	3	26	65 %
N	2	A	1	D	1	N	2	A	3	A	1	A	3	A	1	N	2	SA	0	16	40 %
A	3	D	3	N	2	A	1	A	3	D	3	N	2	N	2	N	2	A	1	22	55 %
A	3	D	3	A	3	D	3	N	2	SD	4	SA	4	D	3	N	2	SD	4	31	77,5 %
SA	4	D	3	A	3	D	3	A	3	SD	4	D	1	D	3	N	2	N	2	28	70 %
A	3	N	2	N	2	A	1	D	1	N	2	N	2	A	1	A	3	N	2	19	47,5 %
A	3	SD	4	A	3	D	3	A	3	D	3	A	3	SD	4	A	3	D	3	32	80 %
N	2	D	3	A	3	D	3	A	3	SD	4	D	1	N	2	N	2	SA	0	23	57,5 %
N	2	D	3	N	2	N	2	SA	4	SD	4	N	2	D	3	N	2	A	1	25	62,5 %
A	3	SD	4	A	3	N	2	SA	4	SD	4	SA	4	SD	4	A	3	N	2	33	82,5 %
SA	4	D	3	A	3	D	3	A	3	D	3	A	3	SD	4	A	3	N	2	31	77,5 %
																					65,5 %

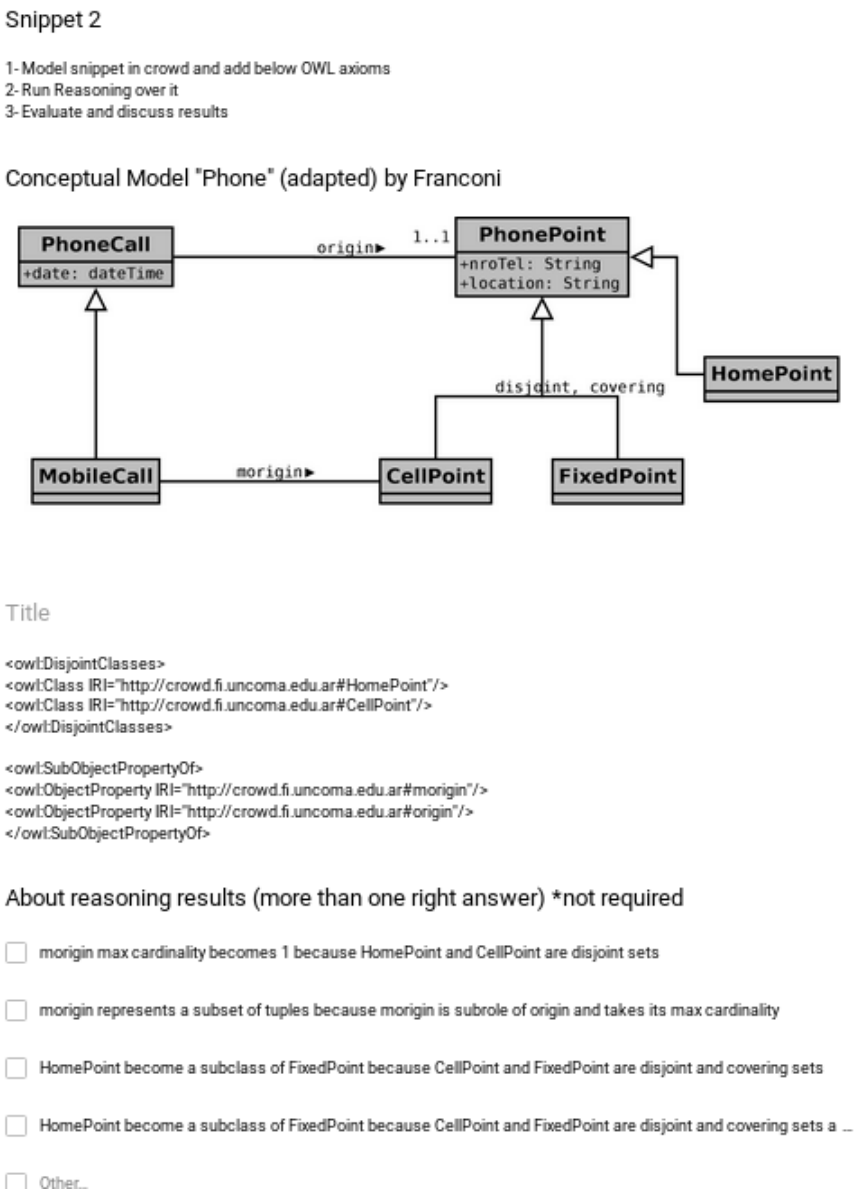


Figura B.2: Ejercicio de modelado en *crowd*, con axiomas textuales para modelar subclasificación de roles.

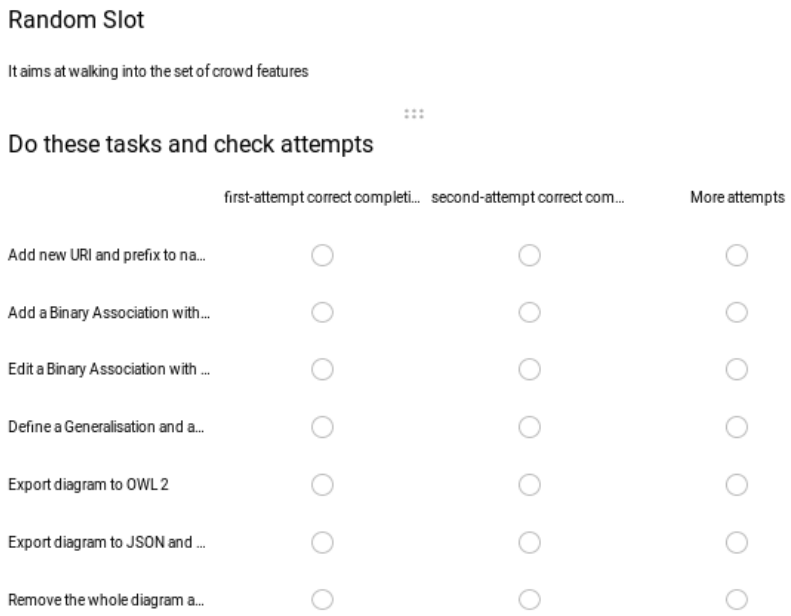


Figura B.3: Ejercicios de modelado para explorar las funcionalidades de la herramienta y medición basada en cantidad de intentos.

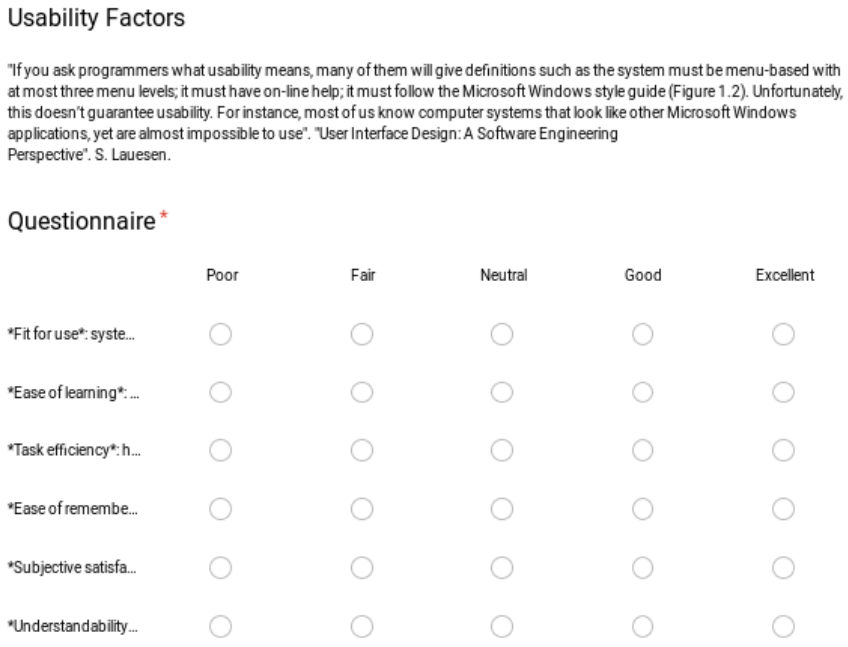


Figura B.4: Cuestionario para calcular el Factor de Usabilidad.

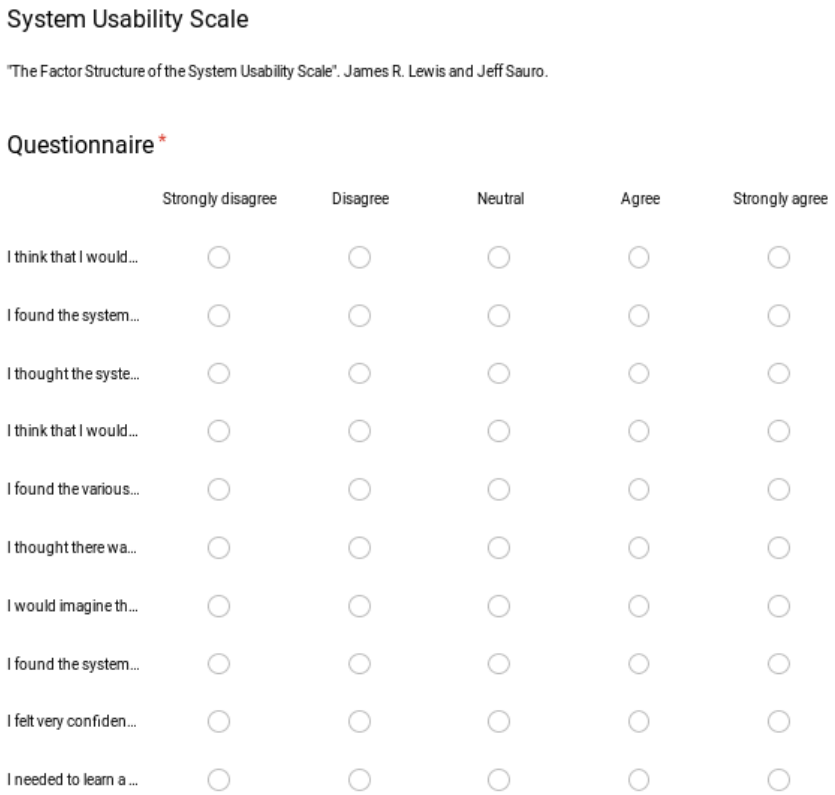


Figura B.5: Cuestionario para medir la Usabilidad del Sistema (SUS)

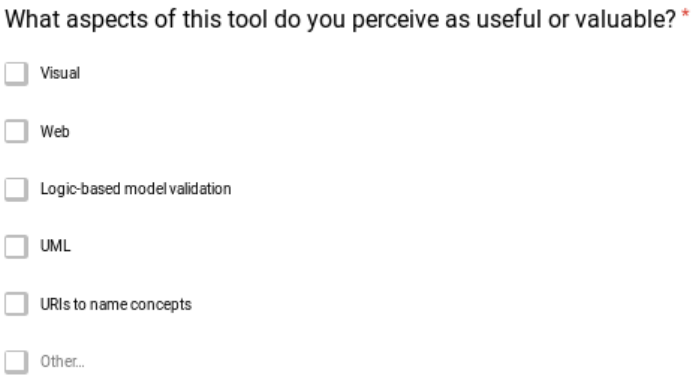


Figura B.6: Aspectos que los usuarios consideran valorable de la nueva herramienta.

Bibliografía

[ABAG17] N. Achich, B. Bouaziz, A. Algergawy, and F. Gargouri. Ontology Visualization: An Overview. In *ISDA*. Springer, 2017.

[ACFLGP01] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE: A Scalable Workbench for Ontological Engineering. In *K-CAP*. ACM, 2001.

[ACK+07] A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Complexity of Reasoning in Entity Relationship Models. In *Description Logics*, 2007.

[AGK06] C. Atkinson, M. Gutheil, and K. Kiko. On the Relationship of Ontologies and Models. In *WoMM*, 2006.

[AH11] D. Allemang and J.A. Hendler. *Semantic Web for the working ontologist effective modeling in RDFS and OWL*. Morgan Kaufmann/Elsevier, 2011.

[ASM80] J. Abrial, S. A. Schuman, and B. Meyer. Specification Language. In *On the Construction of Programs*. 1980.

[bao12] *OWL 2 Web Ontology Language Quick Reference Guide (Second Edition)*. W3C Recommendation, 2012. Disponible en `https://www.w3.org/TR/owl-quick-reference/`.

[BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the EL Envelope. In *IJCAI*. Morgan Kaufmann Publishers Inc., 2005.

[BBLPC14] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, and G. Carothers. *RDF 1.1 Turtle*. World Wide Web Consortium, 2014. `https://www.w3.org/TR/turtle/`.

[BC15] G. Braun and L. Cecchi. Extension Rules for Ontology Evolution within a Conceptual Modelling Tool. In *SAOA@JAIIO*, 2015.

[BCD05] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 2005.

[BCF15] G. Braun, L. Cecchi, and P. Fillottrani. Integrating Graphical Support with Reasoning in a Methodology for Ontology Evolution. In *JOWO@IJCAI*, 2015.

[BCLW12] A. Burton-Jones, R. Clarke, K. Lazarenko, and R. Weber. Is Use of Optional Attributes and Associations in Conceptual Modeling Always Problematic? Theory and Empirical Tests. In *ICIS*, 2012.

[BCM+03a] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[BCM+03b] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 1. Cambridge University Press, New York, NY, USA, 2003.

[BCM+03c] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 2. Cambridge University Press, New York, NY, USA, 2003.

[bec04] *RDF/XML Syntax Specification (Revised).* W3C Recommendation, 2004. Disponible en `https://www.w3.org/TR/rdf-syntax-grammar/`.

[BEF18] G. Braun, E. Estevez, and P. Fillottrani. A Reference Architecture for Ontology Engineering Web Environments. *Journal of Computer Science & Technology*, 2018.

[BGCF16] G. Braun, C. Gimenez, L. Cecchi, and P. Fillottrani. Towards a Visualisation Process for Ontology-Based Conceptual Modelling. In *ONTOBRAS*. CEUR-WS.org, 2016.

[BGCF17] G. Braun, C. Gimenez, L. Cecchi, and P. Fillottrani. Towards a Visualisation Process for Ontology-Based Conceptual Modelling - Extended Abstract. In *Description Logics*. CEUR-WS.org, 2017.

[BGFC17] G. Braun, C. Gimenez, P. Fillottrani, and L. Cecchi. Towards Conceptual Modelling Interoperability in a Web Tool for Ontology Engineering. In *SAOA@JAIIO*, 2017.

[BHGS01] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a Reasonable Ontology Editor for the Semantic Web. In *KI*. Springer-Verlag, 2001.

[BHLS17] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[BHS08] F. Baader, I. Horrocks, and U. Sattler. Description Logics. In *Handbook of Knowledge Representation*, chapter 3. Elsevier, 2008.

[BL85] J. R. Brachman and H. Levesque. *Readings in Knowledge Representation*. Morgan Kaufmann Publishers Inc., 1985.

[BLB08] F. Baader, C. Lutz, and S. Brandt. Pushing the EL Envelope Further. In *OWLED*, 2008.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001.

[BLW16] L. Bühmann, J. Lehmann, and P. Westphal. DL-Learner - A framework for inductive learning on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2016.

[BMC03] S. Bechhofer, R. Moller, and P. Crowther. The DIG Description Logic Interface. In *Description Logics*. CEUR-WS.org, 2003.

[BPC+17] G. Braun, M. Pol'la, L. Cecchi, A. Buccella, P. Fillottrani, and A. Cechich. A DL semantics for reasoning over ovm-based variability models. In *Description Logics*. CEUR-WS.org, 2017.

[BRJ05] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide*. Addison-Wesley Professional, 2005.

[Bro96] J. Brooke. *SUS-A quick and dirty usability scale. Usability evaluation in industry*. CRC Press, 1996.

[Bur05] R. Burkhard. *Towards a Framework and a Model for Knowledge Visualization: Synergies Between Information and Knowledge Visualization*. 2005.

[CdK08] R. Cornet and N. de Keizer. Forty years of SNOMED: a literature review. *BMC medical informatics and decision making*, 2008.

[Ce18] Paolo Ceravolo and et.al. Big data semantics. *J. Data Semantics*, 2018.

[CGL+98] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information Integration: Conceptual Modeling and Reasoning Support. In *CoopIS*. IEEE Computer Society, 1998.

[CGL+07] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning*, 2007.

[CH10] M. Curland and T. Halpin. The NORMA Software Tool for ORM 2. In *CAiSE Forum*, Lecture Notes in Business Information Processing, 2010.

[CKA+18] M. Cheatham, A. Krisnadhi, R. Amini, P. Hitzler, K. Janowicz, A. Shepherd, T. Narock, M. Jones, and P. Ji. The GeoLink knowledge graph. *Big Earth Data*, 2018.

[CLSS14] M. Console, D. Lembo, V. Santarelli, and D. Savo. Graphol: Ontology Representation through Diagrams. In *Description Logics*, 2014.

[CMS99] S. Card, J. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., 1999.

[CNS11] S. Cao, L. Nguyen, and A. Szalas. On the Web Ontology Rule Language OWL 2 RL. In *ICCCI*, 2011.

[COLS12] K. Cerans, J. Ovcinnikova, R. Liepins, and A. Sprogis. Advanced OWL 2.0 Ontology Visualization in OWLGrEd. In *DB&IS*, Frontiers in Artificial Intelligence and Applications, 2012.

[Con04] World Wide Web Consortium. OWL Web Ontology Language Overview, Febrero 2004. `https://www.w3.org/TR/owl-features/`, accedido en Junio del 2017.

[CZ14] C.L. Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275(0):314 – 347, 2014.

[dAF14] R. de Almeida Falbo. SABiO: Systematic Approach for Building Ontologies. In *ONTO.COM@FOIS*, 2014.

[DLSP18] M. Dudáš, S. Lohmann, V. Svátek, and D. Pavlov. Ontology visualization methods and tools: a survey of the state of the art. *The Knowledge Engineering Review*, 33, 2018.

[FFT12] P. Fillottrani, E. Franconi, and S. Tessaris. The ICOM 3.0 intelligent conceptual modelling tool and methodology. *Semantic Web Journal*, 2012.

[FGP+14] R. Falco, A. Gangemi, S. Peroni, D. Shotton, and F. Vitali. Modelling OWL Ontologies with Graffoo. In *The Semantic Web – ESWC*, 2014.

[FK15] P. Fillottrani and C. M. Keet. Evidence-based languages for conceptual data modelling profiles. In *Advances in Databases and Information Systems*. Springer International Publishing, 2015.

[FLGPJ97] M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: from Ontological Art towards Ontological Engineering. In *AAAI97 Spring Symposium*, 1997.

[FMS12] E. Franconi., A. Mosca, and D. Solomakhin. ORM2: Formalisation and Encoding in OWL2. In *On the Move to Meaningful Internet Systems Workshops*, 2012.

[Gar17] D. Garijo. Widoco: A wizard for documenting ontologies. In *The Semantic Web – ISWC 2017*. Springer International Publishing, 2017.

[GBCF16] C. Gimenez, G. Braun, L. Cecchi, and P. Fillottrani. crowd: A Tool for Conceptual Modelling assisted by Automated Reasoning - Preliminary Report. In *SAOA@JAIIO*, 2016.

[GBCF18] C. Gimenez, G. Braun, L. Cecchi, and P. Fillottrani. Towards a Visual SPARQL-DL Query Builder. In *Congreso Argentino de Ciencias de la Computación (CACIC)*, 2018.

[GBI+10] D. Garlan, F. Bachmann, J. Ivers, J. Stafford, L. Bass, P. Clements, and P. Merson. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2010.

[GGH+13] B. Cuenca Grau, M. Giese, I. Horrocks, T. Hubauer, E. Jiménez-Ruiz, E. Kharlamov, M. Schmidt, A. Soylu, and D. Zheleznyakov. Towards Query Formulation, Query-Driven Ontology Extensions in OBDA Systems. In *OWLED*. CEUR-WS.org, 2013.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[GHM+08] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The Next Step for OWL. *Journal of Web Semantics*, 2008.

[Gog94] M. Gogolla. *Extended Entity-Relationship Model: Fundamentals and Pragmatics*. Springer-Verlag, 1994.

[Gom11] H. Gomaa. *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge University Press, 2011.

[Gro14] Oject Management Group. *Object Constraint Language*. Object Management Group, February 2014.

[Gui05] G. Guizzardi. *Ontological foundations for structural conceptual models*. PhD thesis, University of Twente, 2005.

[GW10] G. Guizzardi and G. Wagner. *Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages*. Springer Netherlands, 2010.

[HB11] M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal*, 2011.

[HJ12] R. Hodrob and M. Jarrar. On using a graphical notation in ontology engineering, 2012.

[HKP+09] P. Hitzler, M. Krötzsch, B. Parsia, P. Patel-Schneider, and S. Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[HKS06] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *KR*. AAAI Press, 2006.

[Hla04] J. Hladik. A tableau system for the Description Logic $\mathcal{SHIO}$. In *IJCAR*. CEUR-WS.org, 2004.

[HLSE08] P. Hasse, H. Lewen, R. Studer, and M. Erdmann. The NeOn Ontology Engineering Toolkit. 2008.

[HM01] V. Haarslev and R. Möller. RACER System Description. In *IJCAR*. Springer-Verlag, 2001.

[HM08] T. Halpin and T. Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers Inc., 2008.

[Hor08] I. Horrocks. Ontologies and the Semantic Web. *Commun. ACM*, 51(12), 2008.

[Hor11] I. Horrocks. Tool Support for Ontology Engineering. In *Foundations for the Web of Information and Services - A Review of 20 Years of Semantic Web Research.*, 2011.

[HPS12] M. Horridge and P. Patel-Schneider. *OWL 2 Web Ontology Language Manchester Syntax (Second Edition)*. W3C Recommendation, 2012. Disponible en `https://www.w3.org/TR/owl2-manchester-syntax/`.

[HPSvH03a] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Journal of Web Semantics*, 2003.

[HPSvH03b] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Journal of Web Semantics*, 2003.

[HR04] D. Harel and B. Rumpe. Meaningful modeling: what's the semantics of "semantics¿ *Computer*, 2004.

[HS04] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 2004.

[Jac12] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. Books24x7.com, 2012.

[KA15] A. Khalili and S. Auer. WYSIWYM - integrated visualization, exploration and authoring of semantically enriched un-structured content. *Semantic Web Journal*, 2015.

[Kee18] C.M. Keet. *An Introduction to Ontology Engineering*. University of Cape Town, 2018.

[KF15] C. M. Keet and P. Fillottrani. An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2. *Data Knowledge Engineering Journal*, 2015.

[KFNM04] H. Knublauch, R. Fergerson, N. Noy, and M. Musen. The Protégé OWL plugin: An Open Development Environment for Semantic Web Applications. In *The Semantic Web – ISWC 2004*, 2004.

[KGH11] I. Kollia, B. Glimm, and I. Horrocks. SPARQL Query Answering over OWL Ontologies. In *The Semantic Web: Research and Applications*, 2011.

[KKIM02] K. Kozaki, Y. Kitamura, M. Ikeda, and R. Mizoguchi. Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship". In *EKAW*. Springer, 2002.

[KPS+05] A. Kalyanput, B. Parsia, E. Sirin, B.C. Grau, and J. Hendler. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*, 2005.

[Krö12] M. Krötzsch. OWL 2 Profiles: An Introduction to Lightweight Ontology Languages. In *Reasoning Web*, 2012.

[KS02] J. Krogstie and A. Solvberg. *Information Systems Engineering: Conceptual Modeling in a Quality Perspective*. 2002.

[KS14] S.G. Kobourov and S. Pupyrev B. Saket. Are crossings important for drawing large graphs? *Lecture Notes in Computer Science, Graph Drawing. GD 2014.*, 2014.

[KT05] T. Keller and S-O. Tergan. Knowledge and Information Visualization. chapter Visualizing Knowledge and Information: An Introduction. Springer-Verlag, 2005.

[KWV07] S. Krivov, R. Williams, and F. Villa. GrOWL: A tool for visualization and editing of OWL ontologies. *Journal of Web Semantics*, 2007.

[Lau05] S. Lauesen. *User Interface Design: A Software Engineering Perspective*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[Lie03] T. Liebig. OntoTrack: Fast browsing and easy editing of large ontologies. In *2nd EON@ISWC*, 2003.

[LLNW11] T. Liebig, M. Luther, O. Noppens, and M. Wessel. Owllink. *Semantic Web Journal*, 2011.

[LLR+08] T. Liebig, M. Luther, M. Rodriguez, D. Calvanese, M. Wessel, R. Möller, M. Horridge, S. Bechhofer, D. Tsarkov, and E. Sirin. OWLlink: DIG for OWL 2. In *OWLED*. CEUR-WS.org, 2008.

[LNB14] S. Lohmann, S. Negru, and D. Bold. The ProtégéVOWL Plugin: Ontology Visualization for Everyone. In *The Semantic Web: ESWC 2014 Satellite Events*. 2014.

[LS09] J. Lewis and J. Sauro. The Factor Structure of the System Usability Scale. In *Human Centered Design*. Springer Berlin Heidelberg, 2009.

[LV14] J. Lehmann and J. Voelker. An Introduction to Ontology Learning. In *Perspectives on Ontology Learning*. AKA / IOS Press, 2014.

[MBCF18] G. Marinelli, G. Braun, L. Cecchi, and P. Fillottrani. ArcGen: Hacia un Algoritmo Genético de Layout Automático para Visualización de Modelos Conceptuales. In *Congreso Nacional de Ingeniería Informática – Sistemas de Información (CoNaIISI)*, 2018.

[MGH+12] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, editors. *OWL 2 Web Ontology Language Profiles*. World Wide Web Consortium, second edition edition, December 2012. `https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/`.

[Min74] M. Minsky. A Framework for Representing Knowledge. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[Miz04] R. Mizoguchi. *Ontology Engineering Environments*. Springer, 2004.

[MLH+15] N. Matentzoglu, J. Leo, V. Hudhra, U. Sattler, and B. Parsia. A Survey of Current, Stand-alone OWL Reasoners. In *ORE@DL*, 2015.

[Moo06] D. Moody. What Makes a Good Diagram? Improving the Cognitive Effectiveness of Diagrams in IS Development. *Advances in Information System Development*, 2006.

[Moo09] D. Moody. The "Physics.ºf Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 2009.

[mot12] *OWL 2 Web Ontology Language XML Serialization (Second Edition)*. W3C Recommendation, 2012. Disponible en `https://www.w3.org/TR/owl2-xml-serialization/`.

[MS05] B. Motik and R. Studer. KAON2–A Scalable Reasoning Tool for the Semantic Web. In *ESWC*, 2005.

[MSG+16] J. Moreira, T. Sales, J. Guerson, B. Braga, F. Brasileiro, and V. Sobral. Menthor Editor: An Ontology-Driven Conceptual Modeling Platform. In *JOWO@FOIS*. CEUR-WS.org, 2016.

[MVC+10] L. Masud, F. Valsecchi, P. Ciuccarelli, D. Ricci, and G. Caviglia. From Data to Knowledge - Visualizations as Transformation Processes within the Data-Information-Knowledge Continuum. In *International Conference on Information Visualisation*, 2010.

[NB03] D. Nardi and R. J. Brachman. The description logic handbook. chapter An Introduction to Description Logics, pages 1–40. Cambridge University Press, New York, NY, USA, 2003.

[NHL13] S. Negru, F. Haag, and S. Lohmann. Towards a Unified Visual Notation for OWL Ontologies: Insights from a Comparative

User Study. In *Proceedings of the 9th International Conference on Semantic Systems*. ACM, 2013.

[OBCF18] A. Oyarzún, G. Braun, L. Cecchi, and P. Fillottrani. A Graphical Web Tool with DL-based Reasoning. In *Congreso Argentino de Ciencias de la Computación (CACIC)*, 2018.

[OCRMGR15] L. Otero-Cerdeira, F. Rodríguez-Martínez, and A. Gómez-Rodríguez. Ontology matching: A literature review. *Expert Syst. Appl.*, 2015.

[OWL09] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[OWL12] W3C OWL Working Group. *OWL 2 Web Ontology Language Profiles (Second Edition)*. W3C Recommendation, 11 December 2012. Available at `http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/`.

[PBvdL05] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., 2005.

[PS18] S. Peroni and D. M. Shotton. The SPAR ontologies. In *The Semantic Web - ISWC 2018*, 2018.

[PSH06] P. Patel-Schneider and I. Horrocks. *OWL 1.1 Web Ontology Language Overview*. W3C Recommendation, Diciembre 2006. Available at `https://www.w3.org/Submission/owl11-overview/#ref-owl-1.1-specification`.

[Qui67] M. Quillian. Word concepts: a theory and simulation of some basic semantic capabilities. *Behavioral Science Journal*, 1967.

[sch12] *OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition)*. W3C Recommendation, 2012. Disponible en `https://www.w3.org/TR/owl-rdf-based-semantics/`.

[SMH08] R. Shearer, B. Motik, and I. Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In *OWLED@ISWC*, 2008.

[SMJ02] P. Spyns, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, pages 12–17, 2002.

[SP07] E. Sirin and B. Parsia. SPARQL-DL: SPARQL Query for OWL-DL. In *OLED*, 2007.

[SPG+07] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics*, 2007.

[Spo16] F. Sportelli. NORMA: A software for intelligent conceptual modeling. In *JOWO@FOIS*. CEUR-WS.org, 2016.

[SSS91] M. Schmidt-Schaubßand G. Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 1991.

[ST05] K. Siau and X. Tan. Improving the Quality of Conceptual Modeling Using Cognitive Mapping Techniques. *Data Knowl. Eng.*, 2005.

[TH06] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *IJCAR*. Springer, 2006.

[TNNM13] T. Tudorache, C. Nyulas, N. Noy, and M. Musen. WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the Web. *Semantic Web Journal*, 2013.

[Tob00] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive Description Logics. *Journal of Artificial Intelligence Research*, 2000.

[Tob01] S. Tobies. Complexity Results and Practical Algorithms for Logics in Knowledge Representation. *CoRR*, cs.LO/0106031, 2001.

[Top11] TopQuadrant. TopQuadrant — Products — TopBraid Composer, 2011. `http://www.topquadrant.com/products/TB_Composer.html` accedido en agosto del 2011.

[VBJS14] M. Vigo, S. Bail, C. Jay, and R. Stevens. Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. *International Journal of Human Computer Studies*, 2014.

[Vra10] D. Vrandecic. *Ontology evaluation*. PhD thesis, Karlsruhe Institute of Technology, 2010.

[War04] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., 2004.

[Wor04] World Wide Web Consortium (W3C). OWL Web Ontology Language Reference, 2004. `http://www.w3.org/TR/owl-ref/`, accedido en Junio de 2013.

[XCK+18] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-Based Data Access: A Survey. In *IJCAI*, 2018.

[ZBF17] M. Zárate, G. Braun, and P. Fillottrani. Adding Biodiversity Datasets from Argentinian Patagonia to the Web of Data. In *S4BioDiv@ISWC*, 2017.