

**PROPUESTA Y CONSTRUCCIÓN DE UNA ONTOLOGÍA PARA LENGUAJES DE
MODELADO GRÁFICO**

Gloria Patricia Carmona Ríos

Trabajo final

Especialización en Sistemas

Directora: Gloria Lucía Giraldo Gómez

Universidad Nacional de Colombia Sede Medellín

Facultad de Minas

Escuela de Sistemas

2008

AGRADECIMIENTOS

A Gloria Lucia Giraldo Gómez, Directora del trabajo de grado y Docente de la Universidad Nacional de Colombia Sede Medellín, por su asesoría, colaboración y guía durante todo el proceso de desarrollo del trabajo de grado.

A mis padres Néstor Carmona y Gloria Inés Ríos, a mis hermanos Elizabeth Carmona Ríos y Julián Andrés Carmona Ríos y a mi mejor amigo Jorge Luís Aristizabal Arias por el apoyo incondicional que me brindaron y por que me ayudaron a conseguir las fuerzas necesarias en los momentos en que más lo necesité.

Igualmente, agradezco a mis compañeros de especialización y maestría, por su compañía, apoyo y colaboración, pues me ayudaron a seguir adelante en los momentos difíciles y de esta manera a obtener las metas que me propuse.

TABLA DE CONTENIDO

1.	INTRODUCCION	5
2.	ANTECEDENTES Y JUSTIFICACIÓN	5
3.	MARCO TEORICO	8
3.1.	UML (Unified Modelling Language) [19] [20] [21] [22]	8
3.1.1.	Elementos	9
3.1.2.	Relaciones	10
3.1.3.	Diagramas	10
3.2.	SysML (System Modelling Language) [24] [25]	11
3.2.1.	Diagrama de definición de bloques	12
3.2.2.	Diagrama de bloques interno	13
3.2.3.	Diagrama de actividad	13
3.2.4.	Diagrama de requisitos	14
3.2.5.	Diagrama paramétrico	15
3.3.	WebML (Web Modelling Language)[27] [28]	16
3.3.1.	Diagrama de estructura	16
3.3.2.	Diagrama de composición	17
3.3.3.	Diagrama de navegación	19
3.4.	Ontologías	20
3.4.1.	Técnicas de modelado de conocimiento	21
3.4.2.	OWL (Ontology Web Language) [18]	23
3.4.3.	Frames y OWL: similitudes y diferencias [34]	26
4.	CONSTRUCCIÓN DE LA ONTOLOGÍA	28
4.1.	Determinación del dominio y el alcance de la ontología	30
4.2.	Obtención de los términos relevantes para la ontología	31
4.3.	Definición de clases y organización en una jerarquía taxonómica	32
4.4.	Definición de propiedades	34
4.5.	Definición de facetas para las propiedades	35
4.6.	Definición de Aserciones para las clases	37
4.7.	Creación de individuos	39
4.8.	Validación de la ontología	39
5.	CONCLUSIONES Y TRABAJOS FUTUROS	46
6.	REFERENCIAS BIBLIOGRÁFICAS	48
	ANEXO 1: PREGUNTAS DE COMPETENCIA EN SPARQL	51
	ANEXO 2: RESUMEN	57

LISTA DE FIGURAS

Imagen 1. Tipos de diagramas de SysML [tomada de [24] (2009)] _____	12
Imagen 2. Diagrama de definición de bloques [tomada de [24] (2009)] _____	12
Imagen 3. Diagrama de bloques interno [tomada de [24] (2009)] _____	13
Imagen 4. Diagrama de actividad [tomada de [25] (2009)] _____	14
Imagen 5. Conector Anidamiento de UML _____	14
Imagen 6. Diagrama de requisitos [tomada de [24] (2009)] _____	15
Imagen 7. Diagrama paramétrico [tomada de [24] (2006)] _____	16
Imagen 8. Diagrama de estructura _____	17
Imagen 9. Notación gráfica de unidades y su posible ejecución en HTML [tomada de [27] (2009)] _____	18
Imagen 10. Notación gráfica de páginas y su posible ejecución en HTML [tomada de [27] (2009)] _____	19
Imagen 11. Diagrama de navegación con enlaces contextuales y su posible ejecución en HTML [tomada de [27] (2009)] _____	20
Imagen 12. Jerarquía de clases _____	33
Imagen 13. Modelo del dominio Lenguajes de Modelado Gráfico _____	34
Imagen 14. Propiedades de la ontología _____	35
Imagen 15. Facetas para la propiedad tieneNotacion _____	36
Imagen 16. Facetas de la propiedad tipoDiagrama _____	36
Imagen 17. Propiedades de la clase Diagrama y facetas de cada propiedad _____	37
Imagen 18. Aserciones para la clase ComponenteLMG _____	38
Imagen 19. Creación de individuos para la clase Diagrama _____	39
Imagen 20. Verificación de consistencia con RacerPorter _____	40
Imagen 21. Verificación de consistencia con RacerPro desde Protége OWL _____	41
Imagen 22. Inferencia de nuevas relaciones de herencia o equivalencia entre clases con RacerPro _____	42
Imagen 23. Inferencia de miembros para clases con Racer _____	43
Imagen 24. Ejemplo de inferencia de miembros para clases con Racer _____	44
Imagen 25. Ejecución de la pregunta ¿Cuáles son los diagramas de UML que son usados en SysML? _____	44
Imagen 26. Ejecución de la pregunta ¿Qué características de un sistema se pueden modelar con WebML? en SPARQL _____	46

LISTA DE TABLAS

Tabla 1. Diagramas de UML 2. _____	11
Tabla 2. Componentes de una ontología según la técnica empleada para modelarla _____	23
Tabla 3. Términos candidatos a ser clase _____	32
Tabla 4. Clases de la ontología _____	32

1. INTRODUCCION

En la ingeniería de software ha sido útil desde hace mucho tiempo, la representación de ideas a través de gráficas, permitiendo no solo comunicar dichas ideas entre desarrolladores sino también servir de apoyo en los procesos de análisis de un problema, para esto son valiosos los lenguajes de modelado gráfico (LMG), que son un conjunto de reglas y notaciones gráficas utilizadas para representar en mayor o menor medida todas las fases de un proyecto informático. La gran cantidad de LMG, de elementos que componen un LMG y los múltiples significados de esos elementos puede ser abrumadora para la mayoría de ingenieros de software a la hora de representar y comunicar las necesidades de los clientes y sus ideas, razón por la cual se crea un abismo riesgoso entre la especificación de los requisitos y la implementación del sistema durante el ciclo de vida de construcción de un producto de software. Por lo tanto, se hace necesario contar con una herramienta que sirva de soporte a los integrantes de un proyecto informático para aproximarlos a conocer, comprender e identificar cuál es la estructura de los LMG y su utilidad, con el propósito de ayudarlos a identificar el LMG y los diagramas que sean más acordes para modelar el sistema informático que se pretende desarrollar. En este trabajo se propone entonces, la construcción de una ontología que cumpla con el anterior propósito, para el desarrollo de la ontología se toma como base la metodología propuesta por Noy y McGuinness [1] para el desarrollo de ontologías basadas en Frames y se modifica ligeramente para ser implementada en OWL (*Ontology Web Language*) utilizando la herramienta Protégé-OWL.

2. ANTECEDENTES Y JUSTIFICACIÓN

Los lenguajes de modelado juegan un papel importante en el proceso de representación del comportamiento y la estructura de sistemas. Para ello, se utilizan diagramas que hacen posible a los desarrolladores el análisis, diseño, implementación, pruebas y mantenimiento de estos, de manera consistente a lo largo de su ciclo de vida. Diversas organizaciones, como el OMG (*Object Management Group*), proponen lenguajes de modelado orientados a cubrir varias situaciones de acuerdo a sus perspectivas, a través de diagramas que permiten tomar la información suministrada por los clientes y representarla. Sin embargo, debido a la multiplicidad de elementos que utilizan los lenguajes de modelado, su interpretación en conjunto es compleja. Por lo tanto, se hace importante la unificación de conceptos, con el fin de aproximar al usuario a los diagramas de estos lenguajes, facilitándoles la comprensión de los mismos y de igual manera lograr el consenso de la comunidad desarrolladora, alrededor de un vocabulario común.

Varias técnicas han sido propuestas para realizar un análisis conceptual de los lenguajes de modelado. Éstas, estudian de manera independiente cada uno de ellos para satisfacer necesidades específicas. Sin embargo, en la revisión de la literatura no se encuentra una técnica que haga posible el estudio y el análisis conceptual unificado de varios lenguajes de modelado, independientemente de su orientación, con el

propósito de encontrar equivalencias y diferencias entre los conceptos de cada uno de ellos, y así, ayudar en la comprensión y selección del lenguaje de modelado que sea más apropiado para modelar las necesidades de los clientes.

Entre estas técnicas, se encuentra el modelo MOF (*Meta-Object Facility*) [2] [3] [4], éste, permite representar mediante clases, relaciones, tipos de datos y paquetes la descripción de un lenguaje de modelado. Los metamodelos instanciados de MOF, se emplean para definir cuáles deben ser los componentes utilizados en cada modelo de un determinado lenguaje y cómo debe ser la forma de relacionarlos. Esto, garantiza la coherencia y compatibilidad para transportar los diagramas resultantes de una herramienta de diagramación a otra. OMG determina una arquitectura compuesta por 4 niveles. El nivel 4 (M3) es el nivel más general para definir metamodelos concretos y es aquí, donde se ubica el modelo MOF. En el nivel 3 (M2), se ubican los lenguajes de modelado, siendo instancias del meta-metamodelo. El nivel 2 (M1), son los modelos de los sistemas concretos, que definen las clasificaciones del nivel M0, los elementos del nivel 1 (M0) son las instancias que componen el sistema informático. MOF es útil para la definición de los metamodelos de los lenguajes de modelado, facilitando la correcta construcción de los diagramas que lo componen. Sin embargo, no facilita la comparación de varios lenguajes de modelado por lo tanto no contribuye en la selección adecuada del lenguaje que mejor se adapte a los requisitos a modelar.

Otra técnica, consiste en describir la estructura de un lenguaje de modelado a través de ontologías, las cuales, representan el conocimiento mediante conceptos, relaciones, funciones, instancias y axiomas.

El filósofo Chisholm propuso una ontología basada en el sentido común crítico [5], que requiere un estándar riguroso de soporte para el conocimiento a adquirir. En base a esta postura, Milton y Kazmierczak [6] señalan que la ontología de Chisholm puede ser utilizada para el análisis ontológico de los lenguajes de modelado de datos. Para ello, estos autores propusieron un modelo que permite evaluar los lenguajes desde el punto de vista ontológico. Milton *et al.* [7] desarrollaron en 2001 un método de análisis conceptual de los lenguajes de modelado de datos basado en la ontología filosófica de Chisholm. Los autores tomaron cinco lenguajes de modelado de datos y los analizaron, contrastándolos con las estructuras del mundo real propuestas por Chisholm. Los lenguajes evaluados fueron: Entidad-Relación (ER) [8], Modelos de Datos Funcionales (FDM) [9], el Modelo de Datos Semántico (SDM) [10][11], NIAM [12] y la Técnica de Modelado de Objetos (OMT) [13]. Los autores concluyen que la ontología de Chisholm tiene la potencia necesaria para ser una teoría unificadora de los modelos de datos.

Por otro lado, Genilloud y Frank [14] proponen usar el modelo de referencia RM-ODP (*Reference Model for Open Distributed Processing*) como una ontología de UML, ya que contiene un amplio marco de trabajo conceptual, planteando un vocabulario y un marco semántico común a todos los usuarios de este lenguaje de modelado.

RM-ODP es un estándar de ISO (*International Organization for Standardization*) y de ITU (*International Telecommunication Union*) que define un marco para la especificación arquitectónica de grandes sistemas distribuidos, integrando toda una serie de estándares [15]. Este define, entre otras cosas, cinco puntos de vista para un sistema y su entorno: empresa, información, computación, ingeniería y tecnología. Los

cinco puntos de vista, no corresponden a etapas de proceso de desarrollo o refinamiento. De los cuatro estándares básicos que componen el modelo, los dos primeros se refieren a la motivación general del mismo y a sus fundamentos conceptuales y analíticos; el tercero (ISO/IEC 10746-3; UTI-T X.903) a la arquitectura, definiendo los puntos de vistas referidos; y el cuarto (ISO/IEC 10746-4; UTI-T X.904) a la formalización de la semántica arquitectónica. El problema que los autores identificaron, se basa en que el estándar de UML (*Unified Modeling Language*) confunde a los usuarios, debido a que utiliza una gran cantidad de conceptos y a su vez, varias definiciones para un mismo concepto, por esto proponen utilizar RM-ODP como una ontología, que permita hacer un análisis conceptual del lenguaje y así disminuir la confusión. Los autores se limitaron al planteamiento del uso de RM-ODP como una ontología de UML, tomando como ejemplo el diagrama de Casos de Uso, sin llegar a la implementación de la ontología.

Con base en la exploración anterior, en el presente trabajo se propone la definición, construcción y representación de una ontología de lenguajes de modelado gráfico, que sirva para realizar un análisis conceptual y comparativo de los elementos que conforman tres diferentes lenguajes de modelado y así ayudar a los ingenieros de software a comprender las diferentes especificaciones de estos lenguajes, y de esta manera apoyarlos en la decisión de cuál es el lenguaje más adecuado para modelar los requisitos de un sistema en particular.

Para la implementación de la ontología propuesta se utiliza Protégé-OWL [16], que es una herramienta de libre distribución. Protégé es un editor para construir ontologías y un marco general para representar el conocimiento. Con la ayuda de este software, se pueden construir aplicaciones basadas en el conocimiento, que representen la información de forma ontológica. La característica de este tipo de herramientas es que emplean un lenguaje muy flexible para ser utilizado en la Web, es decir, que facilitan la tarea, no sólo de ser manejable a través de la Web, sino que hacen una descripción semántica de la información, con lo que todo el sistema es menos rígido y por tanto se hace más flexible y potente. Protégé reconoce Frames, XML (*Extensible Markup Language*) Schema, RDF (*Resource Description Framework*) Schema y OWL (*Ontology Web Language*), que son lenguajes semánticos utilizados en la Web, en contraposición a la rigidez del HTML (*HyperText Markup Language*). Protégé permite realizar programas en OWL brindando facilidades para ello, gracias a su entorno gráfico.

OWL utiliza la conexión proporcionada por RDF, para añadir las siguientes capacidades a las ontologías: capacidad de ser distribuida a través de varios sistemas, escalable a las necesidades de la Web, compatible con los estándares Web de accesibilidad e internacionalización, abierto y extensible. Por las razones anteriores OWL es considerado el lenguaje con más futuro en la representación del conocimiento usando ontologías [17] [18]. Se va a utilizar en consecuencia dicho lenguaje, ya que permite representar ontologías a partir de un vocabulario más amplio y una sintaxis más fuerte que la que permite otros lenguajes.

3. MARCO TEORICO

En el análisis, diseño y desarrollo de software, un soporte para los ingenieros de software es la utilización de herramientas, que les permitan describir de forma estática el objetivo al cual se pretende llegar. Para ello, son utilizados los lenguajes de modelado, que son un conjunto de notaciones y reglas a través de las cuales es posible representar el funcionamiento y la estructura de un sistema. Estos tipos de lenguajes, se clasifican en: lenguajes de modelado textual y lenguajes de modelado gráfico. Estos últimos, son a un conjunto de símbolos gráficos y reglas, utilizados para transmitir la información al usuario a través de diagramas, siendo esta representación gráfica el objeto de estudio y análisis en esta investigación.

Un modelo es una simplificación de la realidad, por lo tanto el objetivo del modelado de un sistema es capturar las partes esenciales del sistema, para facilitar este modelado se realiza una abstracción y se plasma en una notación gráfica, esto se conoce como modelado visual. El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar, cuando se intenta construir un sistema complejo es necesario abstraer la complejidad en modelos que el ser humano pueda entender. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando un LMG se puedan implementar en cualquier lenguaje que soporte las posibilidades del LMG.

La ontología propuesta se construye a partir del análisis de tres lenguajes de modelado gráfico existentes actualmente: UML 2.0, SysML, WebML. UML (*Unified Modelling Language*) [19] [20] [21] [22], es un lenguaje estándar utilizado para modelar la estructura, la arquitectura y el comportamiento de una aplicación, con el objetivo de visualizarlos, especificarlos, construirlos y documentarlos a través de símbolos y reglas, además, permite modelar procesos de negocio y estructuras de datos. También, se encuentra SysML (*System Modelling Language*) [23] [24] [25] [26], que es un lenguaje basado en UML, el cual sirve para especificar, analizar, diseñar y verificar sistemas complejos, asimismo, utiliza fundamentos semánticos, que le permiten modelar los requisitos, comportamiento, estructura y parámetros de un sistema. Sin embargo, se diferencia de UML porque contiene diagramas adicionales, tales como el diagrama paramétrico y el diagrama de requisitos. De igual manera, se analiza el lenguaje WebML (*Web Modelling Language*) [27] [28] [29] que, a través de conceptos, apoya las actividades de diseño de sitios Web, provee gráficos, formalismos, especificaciones, y diseño de procesos, soportados por herramientas gráficas, siendo su principal característica la utilización de estereotipos basados en XML.

3.1. UML (*Unified Modelling Language*) [19] [20] [21] [22]

El lenguaje unificado de modelado es un lenguaje estándar para desarrollo de software y sistemas. Aunque UML está pensado para modelar sistemas complejos con gran cantidad de software, el lenguaje es lo suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo (*workflow*) en una empresa, diseño de la estructura de una organización y por supuesto, el diseño de hardware.

La primera versión de UML le permite a las persona comunicar diseños, expresar la esencia de un diseño y capturar y mapear los requisitos funcionales de un software

particular. El OMG desarrolló la versión 2.0 de UML, al reconocer que el modelado de sistemas, y no solo el modelado de software, podría también beneficiarse de un lenguaje unificado como UML.

Los factores más relevantes que condujeron al desarrollo de UML 2.0 fueron: el desarrollo de software orientado a componentes, la arquitectura dirigida por modelos, UML ejecutable y la necesidad de compartir modelos entre herramientas.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones en:

- Visualizar: permite expresar gráficamente un sistema de forma que otro lo puede entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

La notación de UML está compuesta por tres tipos de bloques de construcción: elementos, relaciones y diagramas.

3.1.1. Elementos

Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.). En UML hay cuatro tipos de elementos:

Elementos estructurales: Son las principales partes estáticas de un modelo, son los sustantivos de los modelos de UML.

Elementos de comportamiento: Son la parte dinámica de los modelos de UML, son los verbos de un modelo, representan comportamiento del sistema sobre el tiempo y el espacio.

Elementos de agrupación: Son la parte organizacional de los modelos de UML, son las cajas dentro de las cuales se pueden descomponer los modelos.

Elementos de anotación: Son la parte explicativa de los modelos de UML, son los comentarios o notas que se hacen sobre algún elemento del modelo para describirlo o explicarlo.

3.1.2. Relaciones

Las relaciones son abstracciones que actúan como unión entre los diferentes elementos. Hay cuatro tipos de relación: Dependencia, asociación, generalización y realización.

Relación de dependencia: Es una relación semántica entre dos elementos de un modelo, en donde al hacer un cambio a uno de los elementos (elemento independiente) debe afectar la semántica del otro elemento (elemento dependiente).

Relación de asociación: Es una relación estructural entre clases, que describe un conjunto de vínculos, siendo un vínculo una conexión entre objetos que son instancias de una clase.

Relación de generalización: Es una relación de especialización o generalización en la cual un elemento especializado (elemento hijo) se construye sobre la especificación de un elemento generalizado (elemento padre), los hijos comparte la estructura y el comportamiento de los padres.

Relación de realización: Es una relación semántica entre elementos estructurales, en donde uno de los elementos estructurales especifica un contrato que el otro elemento estructural deberá cumplir.

3.1.3. Diagramas

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas.

En Tabla 1 se listan y se describen los diagramas que componen UML 2, y se muestra la versión de UML en la que originalmente se introdujeron.

Diagrama	Descripción	Versión
Casos de Uso	Muestra la interacción entre los actores y el sistema. También es útil para mapear los requisitos del sistema.	1.x
Actividades	Muestra las actividades secuenciales o paralelas dentro del sistema. Estos diagramas tienen un amplio número de usos, desde definir un flujo de programa básico, hasta capturar los puntos de decisión y acciones dentro de cualquier proceso generalizado.	1.x
Clases	Muestra una colección de elementos estructurales, tales como clases, tipos, interfaces y relaciones entre ellos.	1.x
Objetos	Muestra como las instancias de elementos estructurales	1.x

	se relacionan y usan en tiempo de ejecución.	
Secuencias	Muestra las interacciones entre objetos, donde el orden de las interacciones es importante.	1.x
Comunicaciones (anteriormente colaboraciones)	Muestra las formas en las cuales los objetos interactúan y las conexiones que son necesarias para soportar las interacciones.	1.x
Tiempos	Muestra la interacción entre objetos donde el tiempo es importante.	2.0
Descripción de interacción	Une los diagramas de comunicación, secuencias y tiempos para capturar una interacción importante que ocurre dentro del sistema.	2.0
Estructura compuesta	Muestra la estructura interna de un elemento estructural y describe las relaciones dentro de un contexto dado.	2.0
Componentes	Muestra los componentes importantes dentro del sistema y las interfaces que usan para interactuar unos con otros.	2.0
Paquetes	Muestra la organización jerárquica de grupos de clases y componentes.	2.0
Maquina de estados	Muestra los estados de un objeto durante todo su ciclo de vida y los eventos que pueden cambiar esos estados.	1.x
Despliegue	Muestra como el sistema debe ser finalmente desplegado en una situación dada del mundo real.	1.x

Tabla 1. Diagramas de UML 2.

Por razones de espacio no se explican detalladamente todos los diagramas, para ellos el lector puede remitirse a [20][22].

3.2. SysML (*System Modelling Language*) [24] [25]

SysML es un lenguaje de modelado gráfico de propósito general, utilizado para especificar, analizar, diseñar y verificar sistemas complejos. En particular, el lenguaje provee representaciones gráficas con fundamentos semánticos para modelar los requisitos, comportamiento, estructura y parámetros de un sistema. SysML reutiliza un subconjunto de UML 2, adicionando nuevos diagramas y modificando otros.

SysML es un lenguaje de modelado para la ingeniería de sistemas y tiene como objetivo proporcionar técnicas de modelado para una gran variedad de sistemas, entre los que se incluyen equipos físicos, software, datos, personas, procedimientos e instalaciones.

Los tipos de diagramas de SysML son identificados en Imagen 1.

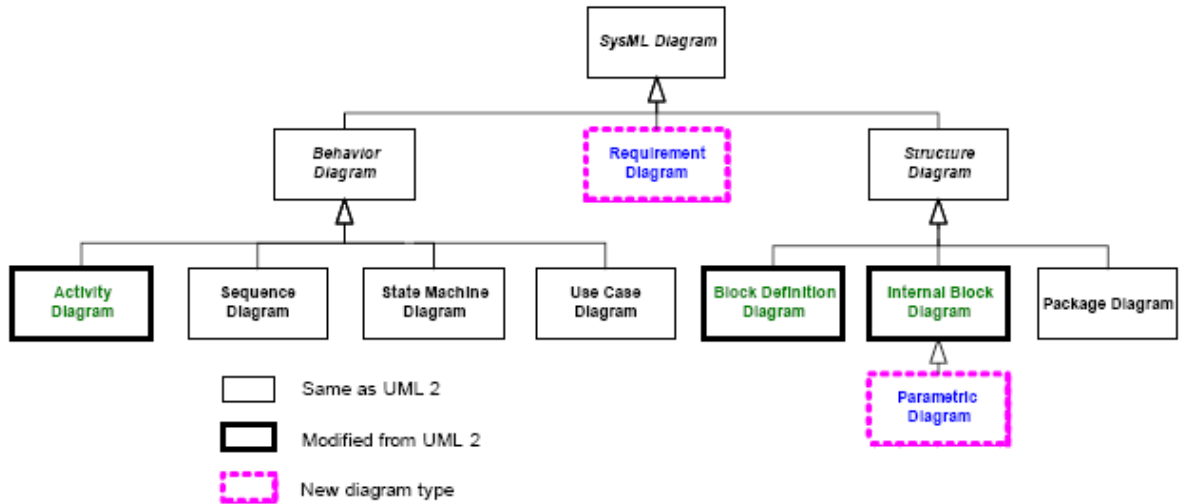


Imagen 1. Tipos de diagramas de SysML [tomada de [24] (2009)]

3.2.1. Diagrama de definición de bloques

El diagrama de definición de bloques (bdd), similar a un diagrama de clases de UML, describe la jerarquía y la clasificación de los componentes del sistema. Es usado para describir relaciones que existen entre bloques. El bloque es la unidad básica de la estructura de SysML y describe un sistema como una colección de partes y conexiones entre ellas, permitiendo comunicación y otros tipos de interacción. Los puertos proveen acceso a la estructura interna de un bloque cuando el objeto es usado dentro del contexto de una estructura. SysML provee puertos estándares los cuales soportan comunicación cliente-servidor y *FlowPorts* que definen flujos desde y hacia un bloque.

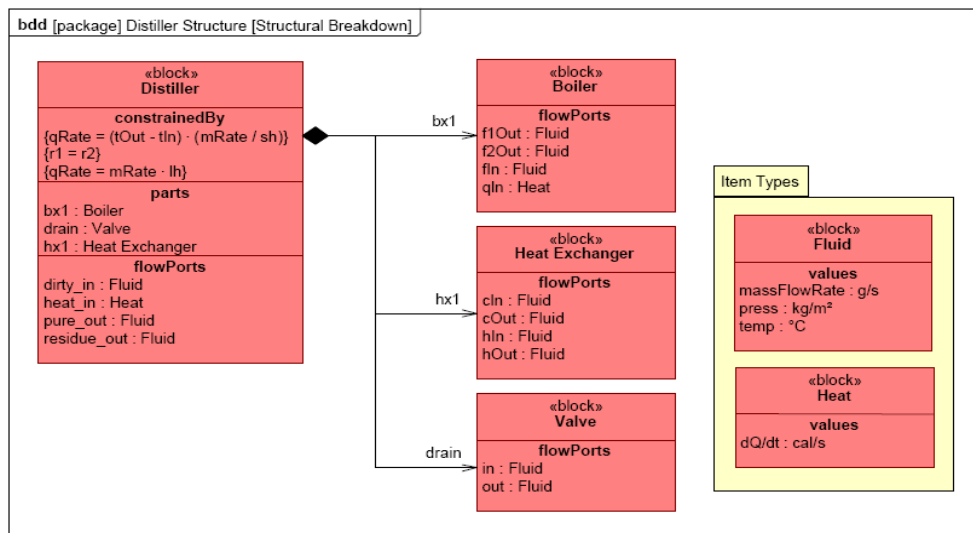


Imagen 2. Diagrama de definición de bloques [tomada de [24] (2009)]

Un ejemplo de un diagrama de definición de bloques es mostrado en Imagen 2, en esta imagen se puede ver un destilador como un bloque compuesto por otros bloques, los roles de los bloques que componen el destilador son colocados al final de cada asociación y corresponden a las partes en el diagrama de bloques interno, en el bloque del destilador se muestran sus restricciones, sus partes y sus *FlowPorts*.

3.2.2. Diagrama de bloques interno

El diagrama de bloques interno (ibd) es usado para describir internamente los bloques. Describe la estructura interna de un sistema en términos de sus partes, puertos y conectores.

Un ejemplo simple del diagrama de bloques interno es mostrado en Imagen 3, en esta imagen se puede ver la estructura interna del bloque del destilador, en ésta solo se muestran dos partes del destilador, el intercambiador de calor (hx1) y la caldera (bx1), que son interconectadas para permitir el flujo de material y energía entre ellas y hacia el exterior.

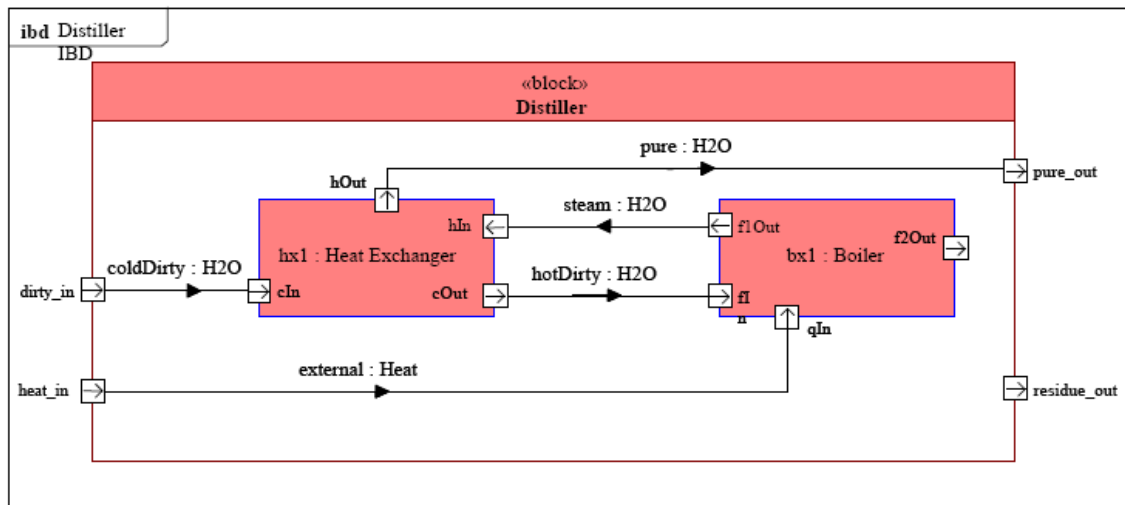


Imagen 3. Diagrama de bloques interno [tomada de [24] (2009)]

3.2.3. Diagrama de actividad

El diagrama de actividad es una de las extensiones de UML 2.0, la actividad representa la unidad básica de comportamiento que es usada en el diagrama de actividad, secuencia y máquina de estados. El diagrama de actividad es usado para representar los flujos de entrada y salida de datos, los cuales pueden ser creados o requeridos por el flujo. Los flujos pueden ejecutarse en paralelo, sincronizados o separados por una condición. El diagrama de actividades de UML 2.0 es extendido por varias propiedades en SysML:

- Flujos de control extendidos con información adicional para parar acciones o controlar el flujo a través de operadores de control.
- Soporte para modelado de sistemas continuos.
- Probabilidades de flujos.
- Reglas para modelar actividades en un diagrama de definición de bloques.

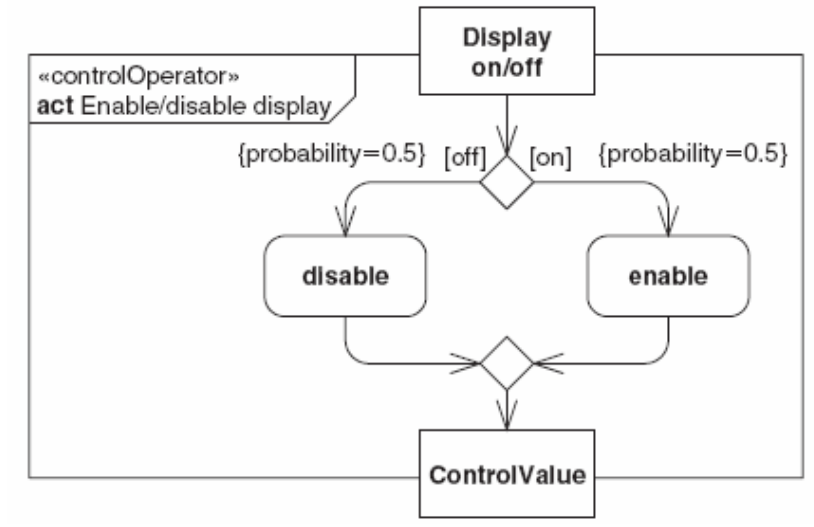


Imagen 4. Diagrama de actividad [tomada de [25] (2009)]

3.2.4. Diagrama de requisitos

El diagrama de requisitos es usado para integrar los modelos de sistemas con los requisitos que son capturados con herramientas típicas de administración de requisitos. SysML incluye un constructor gráfico para representar requisitos y relacionarlos con otros elementos del modelo. El conector Anidamiento de UML (Imagen 5) es usado para indicar los requisitos que componen otro requisito. Un requisito está relacionado con otros artefactos modelados a través de un conjunto de estereotipos del conector Dependencia, por ejemplo: El conector Dependencia *<<deriveReq>>* y el conector Dependencia *<<satisfy>>* describen la derivación de requisitos de otros requisitos y la satisfacción de requisitos por diseño, respectivamente. El conector Dependencia *<<verify>>* muestra la conexión de un caso de prueba al requisito o requisitos verificados por éste.



Imagen 5. Conector Anidamiento de UML

Un ejemplo del diagrama de requisitos es mostrado en Imagen 6.

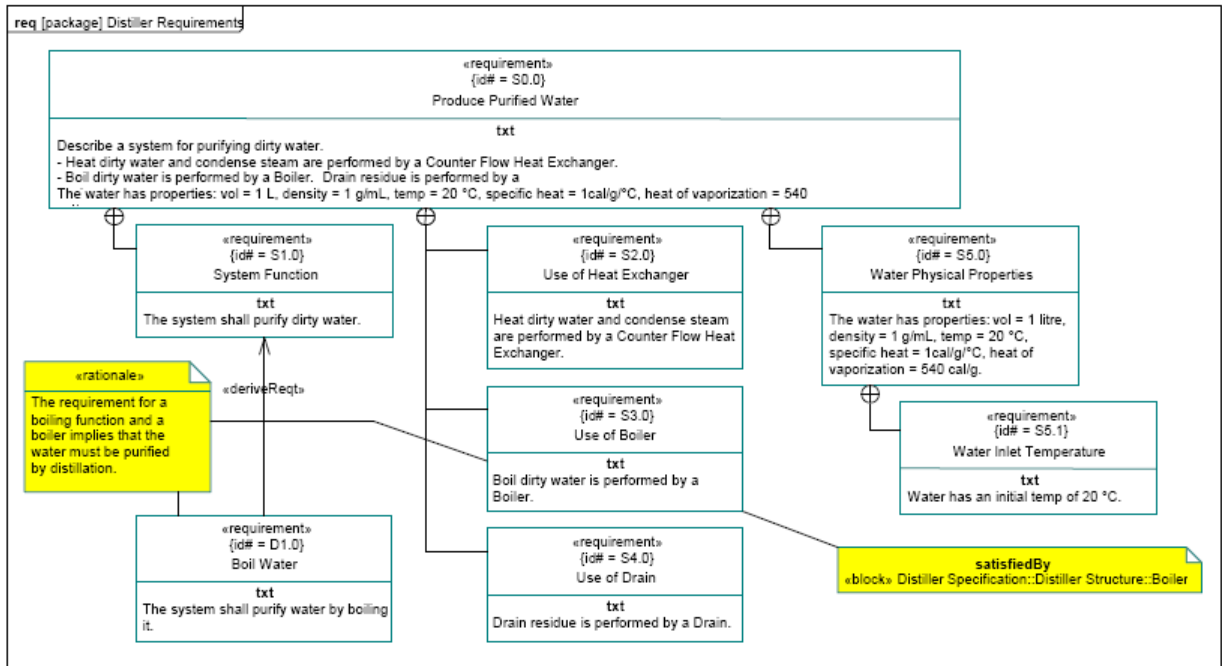


Imagen 6. Diagrama de requisitos [tomada de [24] (2009)]

3.2.5. Diagrama paramétrico

El diagrama paramétrico representa restricciones para los valores de las propiedades del sistema, para este tipo de diagrama SysML introduce el *ConstraintBlock*, éste define un conjunto de parámetros y una o más restricciones para esos parámetros, igualmente puede ser usado para expresar ecuaciones matemáticas, valores estadísticos y funciones.

Un ejemplo del diagrama paramétrico es mostrado en Imagen 7, en esta imagen se describen las restricciones en los flujos del bloque del destilador. Los cuadros ubicados al lado izquierdo y derecho representan los flujos y las cinco restricciones (rectángulos amarillos) muestran como los parámetros de las ecuaciones se unen a las propiedades de los flujos indicando qué restricciones deben cumplir.

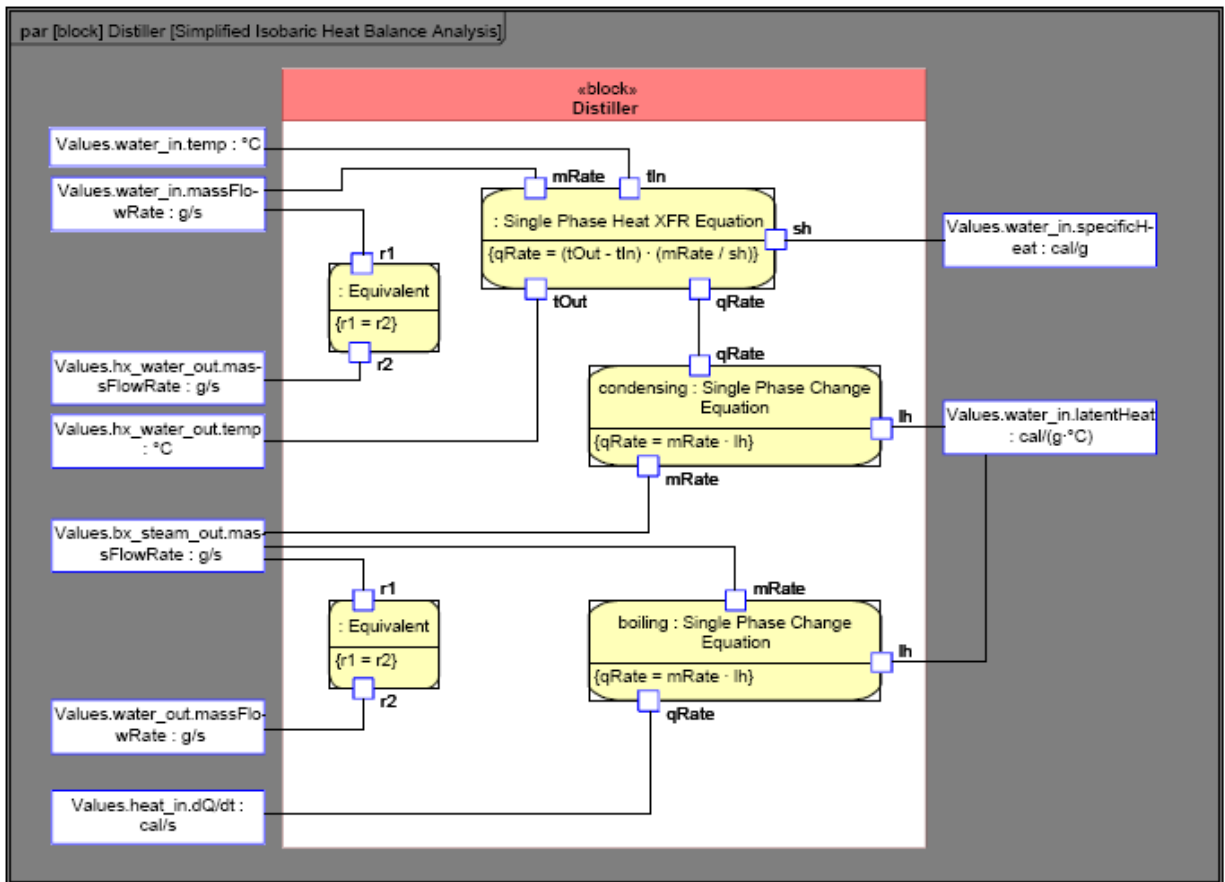


Imagen 7. Diagrama paramétrico [tomada de [24] (2006)]

3.3. WebML (Web Modelling Language)[27] [28]

WebML es un lenguaje de modelado gráfico utilizado para apoyar las actividades del diseño de sitios Web. Provee gráficos, formalismos, especificaciones y diseño de procesos apoyados por herramientas gráficas. Define varios tipos de diagramas: de estructura, composición y navegación.

3.3.1. Diagrama de estructura

En el diagrama de estructura se definen las entidades o contenedores de datos y sus relaciones, este diagrama expresa el contenido de un sitio Web en términos de entidades y relaciones relevantes. El elemento fundamental del modelo de estructura son las entidades (contenedores de datos) y las relaciones (conectores de entidades), las entidades deben tener atributos con un tipo asociado y las relaciones deben tener una cardinalidad y un rol asociado. En Imagen 8 se muestra un ejemplo de un diagrama de estructura, el cual consiste de cuatro entidades (*Artist*, *Album*, *Review*, *Track*) y tres relaciones (*Artist2Album*, *Artist2Review*, *Album2track*).

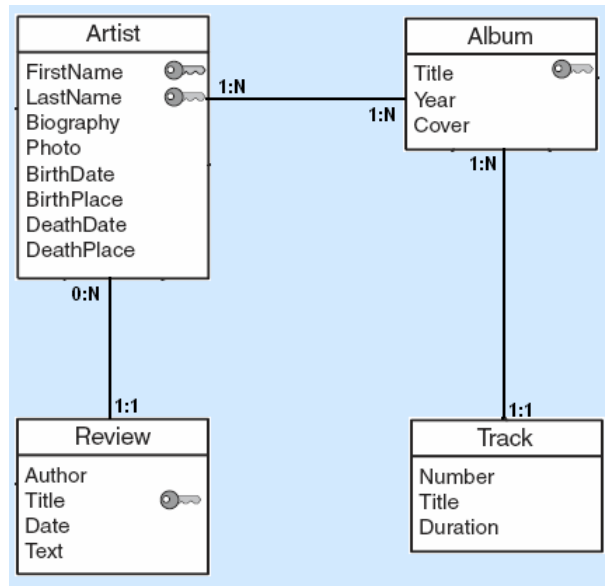


Imagen 8. Diagrama de estructura

3.3.2. Diagrama de composición

El propósito del diagrama de composición es definir los nodos que forman parte del hipertexto contenido en el sitio Web, es decir, se especifican las páginas y las unidades (elementos atómicos de información que deben aparecer en el sitio Web) que componen el sitio Web.

WebML soporta seis tipos de unidades que pueden ser usadas para componer hipertexto:

Unidades de Datos: Muestran información sobre un solo objeto, son definidas para seleccionar una mezcla de información. Para definir una unidad de datos se requiere la indicación del concepto al cual se refiere la unidad y la selección de los atributos de la unidad.

Unidad Multidatos: Muestra información sobre un conjunto de objetos, presenta múltiples instancias de una entidad o componente. Una unidad multidatos tiene dos partes: el contenedor que incluye las instancias que se desean mostrar y la unidad de datos usada para la presentación de cada instancia.

Unidad Índice: presenta múltiples instancias de una unidad o componente como una lista, esta unidad tiene dos partes principales: el contenedor que incluye las instancias que se desean mostrar (las instancias deben ser una entidad, una relación o un componente) y los atributos usados como clave del índice.

Unidad Scroller: provee comandos para desplazarse a través de los objetos en un contenedor. Esta unidad es normalmente usada junto con una unidad de datos, la cual representa el elemento actual visualizado del contenedor.

Unidad Filtro: provee campos de entrada para buscar los objetos en un contenedor, esta unidad es normalmente usada junto con una unidad índice o multidatos, la cual muestra los objetos que coinciden con las condiciones de búsqueda.

Unidad Directa: Expresa un tipo particular de índice, el cual contiene un solo objeto asociado a otro objeto por una relación uno a uno.

En Imagen 9 se muestra la notación gráfica de las unidades y una posible ejecución de cada unidad en una implementación basada en HTML.

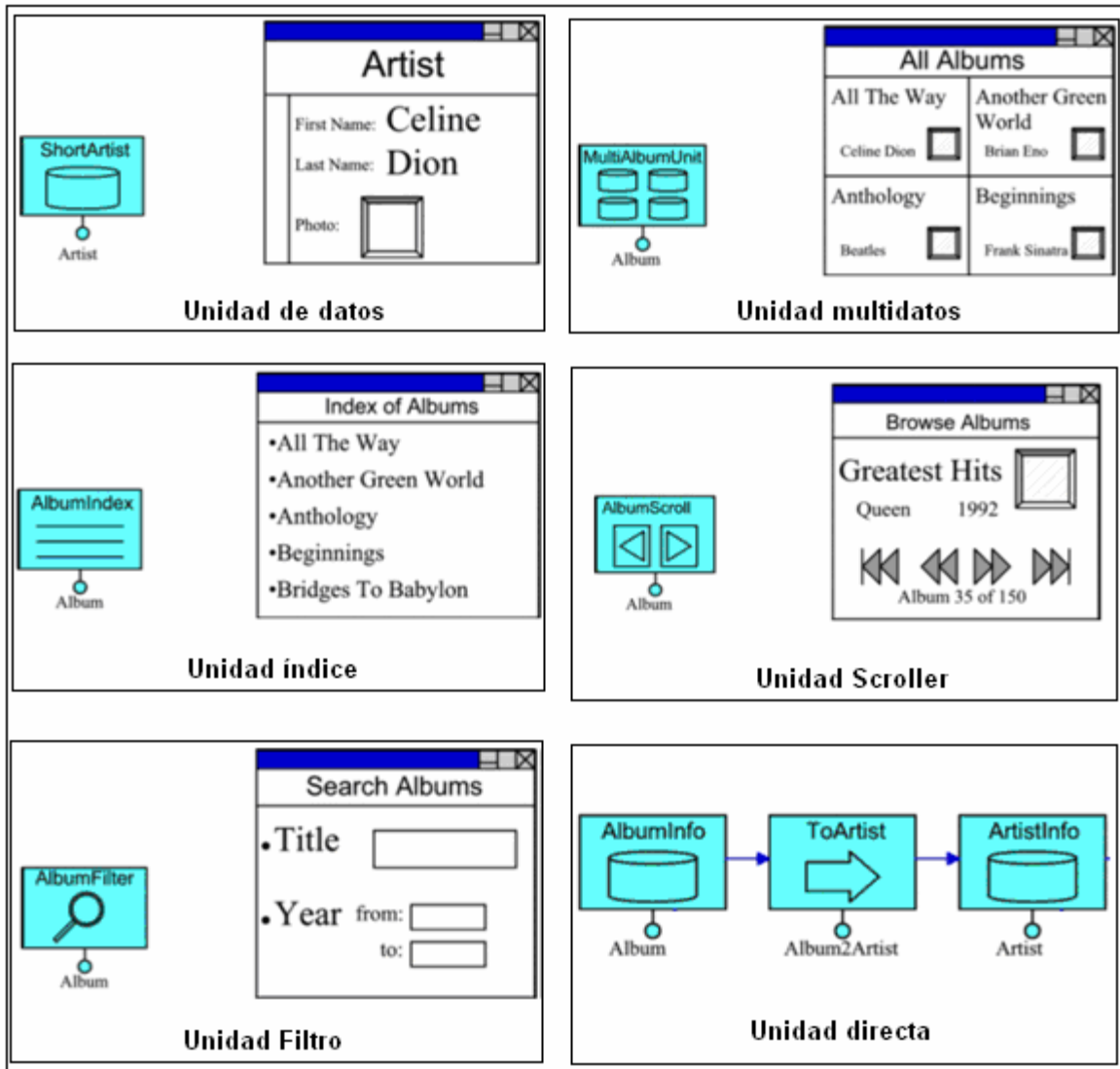


Imagen 9. Notación gráfica de unidades y su posible ejecución en HTML [tomada de [27] (2009)]

Una página es una abstracción de una región independiente de la pantalla, la cual es tratada como un bloque de interfaz independiente. Las páginas deben ser internamente

organizadas en unidades y/o recursivamente en páginas, para este último las subpáginas de la página contenedora son tratadas como bloques de presentación independientes, hay dos formas de visualizar las subpáginas: conjuntas (se muestran las subpáginas al mismo tiempo) o disjuntas (unas páginas se muestran como alternativas de otras). En Imagen 10 se muestran varias subpáginas y su posible ejecución en una implementación basada en HTML, las páginas del lado izquierdo se muestran conjuntas y las del lado derecho se muestran disjuntas.

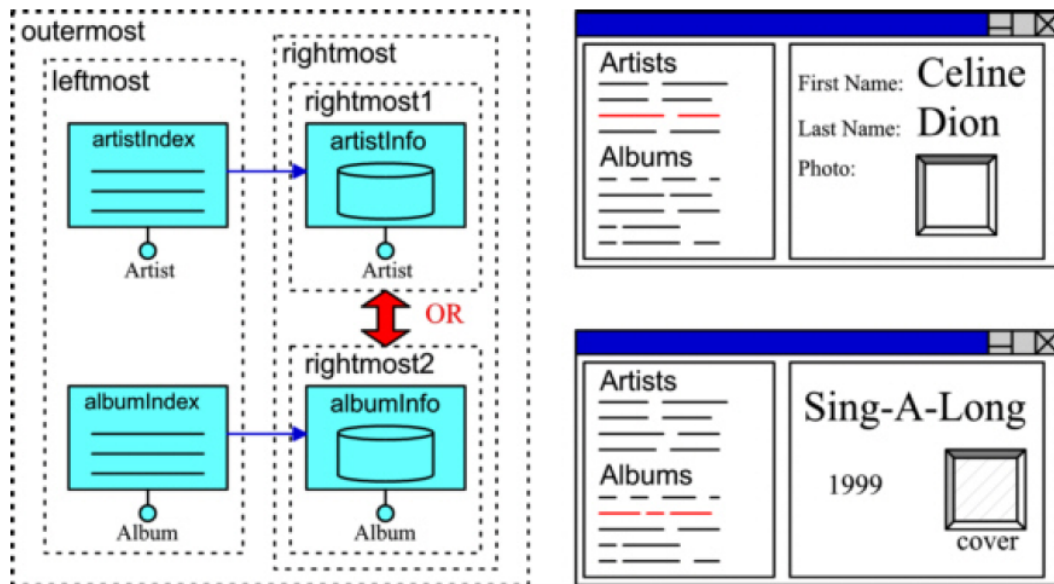


Imagen 10. Notación gráfica de páginas y su posible ejecución en HTML [tomada de [27] (2009)]

3.3.3. Diagrama de navegación

El propósito del diagrama de navegación es especificar la forma en la cual las unidades y las páginas son conectadas para formar un hipertexto, para esto WebML provee la noción de enlaces, de los cuales hay dos tipos:

Enlaces contextuales: conectan unidades de una forma coherente a la semántica expresada por el diagrama de estructura de la aplicación. Un enlace contextual lleva información (de contexto) de la unidad de origen a la unidad destino, esta información es usada para determinar el objeto o conjunto de objetos a ser mostrados en la unidad destino.

Enlaces no contextuales: conectan páginas libremente, independientemente del contexto.

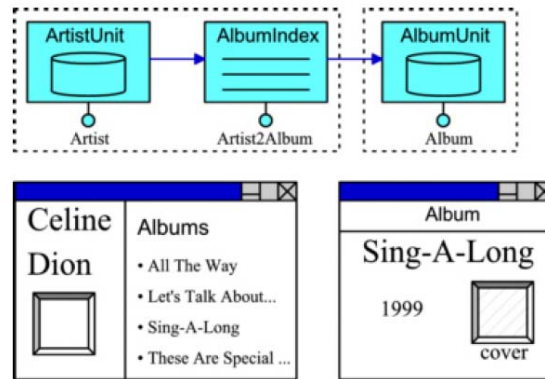


Imagen 11. Diagrama de navegación con enlaces contextuales y su posible ejecución en HTML [tomada de [27] (2009)]

En Imagen 11 se muestra un diagrama de navegación con enlaces contextuales, en donde dos unidades pertenecen a una misma página (página origen) y la otra unidad pertenece a una página diferente (página destino). Todas las unidades son conectadas con enlaces contextuales.

3.4. Ontologías

Aunque existen varias definiciones del concepto de ontología, una de las más ampliamente aceptadas es la de Thomas Gruber [30]: “Una Ontología es una especificación formal y explícita de una conceptualización compartida”. El término “conceptualización” implica que toda ontología desarrolla un modelo abstracto del dominio o fenómeno del mundo que representa, dicho modelo abstracto se basa esencialmente en el empleo de conceptos, atributos, valores y relaciones. Con “especificación explícita” se quiere expresar que una ontología supone la descripción y representación de un dominio concreto mediante conceptos, atributos, valores, relaciones, funciones, etc., definidas explícitamente, las máquinas no pueden dar nada por supuesto o por obvio, y todo conocimiento (por básico que parezca, mientras sea necesario) debe ser explícitamente representado. El término “formal” hace referencia al hecho de que cualquier representación (concepto, atributo, valor, etc.) ha de ser expresada en una ontología mediante un formalismo siempre idéntico, de manera que pueda ser reutilizada y leída por cualquier máquina independientemente del lugar, de la plataforma o idioma del sistema que lo emplee. Por último una ontología es “compartida” cuando dicha conceptualización y su representación formal y explícita ha sido favorablemente acogida por todos los usuarios de la misma, esto permite por una parte distinguir claramente las ontologías de las bases de datos (en las que también puede hablarse de conceptualización y especificación formal y explícita mediante conceptos, atributos y valores, pero en la que el creador no tiene que lograr el consenso de nadie).

3.4.1. Técnicas de modelado de conocimiento

Las ontologías pueden ser modeladas con diferentes técnicas de modelado de conocimiento, a lo largo de los años se han diseñado diversos formalismos y lenguajes que permiten modelar de un modo formal el conocimiento, entre las técnicas diseñadas están [31]:

Lógica proposicional: una proposición es una sentencia que puede decirse que es falsa o verdadera, en la lógica proposicional se asignan símbolos a cada sentencia y se utilizan operadores lógicos sobre ellos para crear proposiciones más complejas. Los símbolos utilizados son *AND*, *OR*, *NOT*, *IMPLIES* y *EQUIVALENCE*. Partiendo de los símbolos y utilizando los diferentes operadores, se construyen las proposiciones complejas, de las cuales es posible obtener si son ciertas o falsas, operando a partir de los valores de verdad de cada uno de los símbolos iniciales, utilizando el cálculo proposicional.

Lógica de primer orden: Es una ampliación de la lógica proposicional, que utiliza dos operadores más, el cuantificador universal y el existencial. Utiliza también símbolos para representar conocimiento y operadores lógicos para construir sentencias más complejas, pero a diferencia de la lógica proposicional, los símbolos pueden representar constantes, variables, predicados y funciones. Las constantes son símbolos que comienzan por minúsculas y las variables símbolos que empiezan por mayúsculas. Los predicados representan afirmaciones sobre objetos y las funciones permiten asociar elementos de un conjunto a un elemento de otro conjunto.

Lógica descriptiva: La lógica descriptiva se basa en representar el conocimiento utilizando una terminología o vocabulario del dominio (*TBOX*) y un conjunto de afirmaciones (*ABOX*). El vocabulario consiste en conceptos y roles. Los conceptos corresponden a conjuntos de elementos y los roles a relaciones binarias entre elementos. Existen conceptos y roles atómicos y complejos. Se pueden construir y existen razonadores que permiten razonar sobre las *TBOX* y *ABOX*, pudiendo determinar, por ejemplo, si el contenido de la *TBOX* es factible, o qué relaciones están incluidas en otras. Al trabajar sobre el *ABOX*, un razonador puede indicar que a partir de las afirmaciones existentes, un determinado elemento es una instancia de un concepto y si las afirmaciones son consistentes con el modelo.

Lenguajes basados en frames: Estos lenguajes son similares a los lenguajes de programación orientados a objetos, en el sentido de que modelan el conocimiento utilizando clases (frames), atributos, objetos y relaciones, y utilizan relaciones de generalización y especialización para representar la organización jerárquica de los conceptos.

Técnicas de ingeniería de software: Se enfoca en emplear UML como técnica para modelado de conocimiento, basándose en que UML es de fácil uso y entendimiento, que es una representación gráfica estándar y que muchas herramientas CASE (*Computer Aided Software Engineering*) lo soportan. Se usa el diagrama de clases para representar conceptos y relaciones entre conceptos y axiomas en OCL (*Object Constraint Language*). El diagrama de clases se usa para representar instancias.

Tecnología de bases de datos: La notación Entidad-Relación permite modelar clases, atributos y relaciones *Ad hoc*, además existen muchas otras extensiones que pueden ser usadas para modelar conocimiento, por ejemplo restricciones de claves, dependencias funcionales, roles, relación de generalización y especialización y restricciones de cardinalidad para relaciones. Los axiomas pueden ser representados usando restricciones de integridad o usando notaciones complementarias.

Dependiendo de la técnica de modelado de conocimiento utilizada para modelar una ontología, los términos manejados para referirse a los componentes de la ontología varían, en Tabla 2 se listan los diferentes términos que se manejan según la técnica empleada [32]:

Frames y lógica de primer orden	Lógica descriptiva	Técnicas de ingeniería de software	Tecnología de base de datos	Definición
Clase	Concepto	Clase	Entidad	Descripción formal de una entidad del universo o dominio que se quiere representar. Constituye la pieza básica de estructuración del conocimiento.
Relación	Rol	Asociación	Relación	Interacción o enlace entre los conceptos/clases/entidades del dominio que se modela. Algunas relaciones semánticas básicas son: subclase de, parte de, parte exhaustiva de, conectado a, es un, etc. Suelen configurar la taxonomía del dominio.
Atributos	Propiedades	Atributos	Atributo	Característica que permite describir más detalladamente la clase/concepto/entidad. Este componente se concreta mediante un valor. Los valores pueden ser tipos básicos como cadenas de caracteres o números, pero también pueden ser otras clase/concepto/entidad.
Axiomas	Axiomas	Axioma (OCL)	Restricción de integridad	Regla que se añade a la ontología y que permite describir el comportamiento de los conceptos/clases/entidades. Se establecen a partir de valores específicos de las propiedades/atributos. Permiten dejar constancia de que ciertos valores de propiedades/atributos introducidos

				son coherentes con las restricciones de la ontología, o bien inferir posteriormente valores de propiedades/atributos que no se han introducido explícitamente. De esta forma, a través de este componente es posible inferir conocimiento no codificado explícitamente en la ontología.
Instancias	Individuos	Objetos	Instancias (sentencia insert de SQL)	Representan objetos concretos del dominio. La colección de este componente constituye la base de hechos (también denominada base de conocimiento) del modelo.

Tabla 2. Componentes de una ontología según la técnica empleada para modelarla

En [33] los autores proponen una ontología basada en Frames, en la cual se describen los componentes de tres lenguajes de modelado (UML, SysML y WebML) y se organizan jerárquicamente, permitiendo a través de la jerarquía de clases y de las restricciones sobre los atributos de esas clases, obtener los componentes que pertenecen a un determinado lenguaje de modelado o a un diagrama. En el presente trabajo se adopta OWL DL como paradigma de representación de la ontología, para obtener una mayor expresividad, y de esta manera, no solo organizar los componentes de los tres lenguajes de modelado jerárquicamente, sino dar una definición más completa de cada uno de los conceptos de la ontología y de las relaciones entre esos conceptos, a través de la definición de aserciones sobre los conceptos y las propiedades, dándole un mayor sentido a la ontología y ofreciendo una mayor capacidad de razonamiento e inferencia.

3.4.2. OWL (*Ontology Web Language*) [18]

OWL forma parte de un conjunto creciente de recomendaciones del W3C (*World Wide Web Consortium*) relacionadas con la Web semántica:

XML proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.

XML Schema es un lenguaje que se utiliza para restringir la estructura de los documentos XML, además de para ampliar XML con tipos de datos.

RDF es un modelo de datos para objetos ("recursos") y relaciones entre ellos, proporcionando una semántica simple para éste. Este tipo de modelo de datos puede ser representado en una sintaxis XML.

RDF *Schema* es un vocabulario utilizado para describir propiedades y clases de recursos RDF, con una semántica para la generalización y jerarquización tanto de propiedades como de clases.

OWL es el acrónimo de *Ontology Web Language* o Lenguajes de ontologías Web, es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la WWW (*World Wide Web*). OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML. Se espera que, junto al entorno RDF y otros componentes, estas herramientas hagan posible el proyecto de la Web semántica.

La Web semántica se basará en la capacidad de XML para definir esquemas de etiquetas a medida y en la aproximación flexible de RDF, para representar datos. El primer requisito por encima de RDF para la Web semántica es un lenguaje de ontologías que pueda describir formalmente el significado de la terminología usada en los documentos Web. Si se espera que las máquinas hagan tareas útiles de razonamiento sobre estos documentos, el lenguaje debe ir más allá de las semánticas básicas del RDF Schema, OWL ha sido diseñado para cubrir esta necesidad de un lenguaje de ontologías Web, por lo que puede usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL es una extensión del lenguaje RDF, por lo que emplea las tripletas de éste, aunque es un lenguaje con más poder expresivo. OWL posee más funcionalidades para expresar el significado y semántica que XML, RDF y RDFS (*RDF Schema*), por lo que va más allá que estos lenguajes pues ofrece la posibilidad de representar contenido de la Web interpretable por máquinas.

OWL proporciona un lenguaje para la definición de Ontologías estructuradas, basadas en la Web, que ofrece una integración e interoperabilidad de datos más rica entre comunidades descriptivas. Los lenguajes anteriores se utilizaron para desarrollar herramientas y ontologías para comunidades de usuarios específicas (particularmente en las ciencias y en aplicaciones de comercio electrónico de compañías específicas), pero no fueron definidos para ser compatibles con la arquitectura de la World Wide Web en general, y de la Web Semántica en particular. OWL utiliza URIs (*Uniform Resource Identifiers*) para fijar nombres y la infraestructura para descripciones en la Web proporcionada por RDF para que las ontologías puedan ser distribuidas por muchos sistemas, sean escalables a las necesidades de la Web y sean compatible con estándares Web para la accesibilidad y la internacionalización.

El lenguaje RDF es muy útil en situaciones en las que la información necesita ser procesada por aplicaciones que intercambian información legible por máquina, más que por humanos. RDF provee un marco común de trabajo para expresar esta información y para intercambiarla entre aplicaciones distintas mediante una serie de "*parsers*" o analizadores RDF y otras herramientas de procesamiento automatizado.

RDF está basado en la idea de identificar los recursos en la Web usando URIs, y describiendo los recursos en términos de propiedades simples y valores. Una descripción RDF es un conjunto de proposiciones simples (también llamadas sentencias o declaraciones) y una proposición se conoce también como una tripleta, porque está compuesta de 3 cosas: un sujeto, un predicado y un objeto. Estas sentencias se pueden representar formalmente usando la tripleta (sujeto, predicado, objeto), pero existe otra forma de notación que es mostrar una sentencia mediante

grafos dirigidos. Así, en RDF es posible representar declaraciones simples sobre los recursos como un grafo (graph) de nodos y arcos que representan los recursos, y sus propiedades y valores. Los sujetos y objetos son nodos, mientras que los predicados son arcos. Así pues, una tripleta se representa mediante nodos conectados por líneas con etiquetas. Los nodos representan recursos y las líneas con etiquetas las propiedades de esos recursos. Los 3 elementos de una tripleta se representan mediante URIs.

RDF también provee una sintaxis basada en XML (llamada RDF/XML) para guardar e intercambiar estos grafos. Como HTML, este RDF/XML es procesable por máquina y, usando URIs se pueden enlazar las piezas de información a lo largo de la Web. Sin embargo, al contrario que el hipertexto convencional, los URIs de RDF pueden hacer referencia a cualquier cosa identificable, incluyendo cosas que pueden no ser directamente recuperables en la Web. El resultado es que, en adición para describir tales cosas como páginas web, RDF puede también describir carros, negocios, gente, noticias, eventos etc. E incluso, las propiedades RDF que tienen los URIs, para precisamente identificar las relaciones que existen entre los términos enlazados. En RDF tanto los sujetos, como las propiedades y los objetos, son recursos.

Mientras el español o el inglés son buenos para la comunicación entre humanos, RDF está concebido para ser procesado por máquina. Para hacer declaraciones adaptables al procesamiento por máquina, se necesitan 2 cosas:

- Un sistema de identificadores procesable por máquina para identificar un sujeto, predicado u objeto en una declaración, sin posibilidad de confusión con un identificador similar que debe usarse por alguien en la Web.
- Un lenguaje procesable por máquina para representar estas declaraciones e intercambiar la información entre máquinas.

Afortunadamente, la existencia de la arquitectura Web ofrece ambas posibilidades. La Web ya provee una forma de identificación, el *Uniform Resource Locator* (URL). Un URL es una cadena de caracteres que identifica los recursos de la Web para representar un mecanismo de localización, sin embargo, es preciso recordar que muchas cosas en la Web no tienen localización o URLs. Con el tiempo, el localizador URL ha sido reemplazado, en RDF, por el localizador URI que es mucho más preciso y que combina y engloba las posibilidades de identificación y localización tanto de URL como de URN (*Uniform Resource Name*).

Conceptos básicos del modelo RDF son la idea de usar referencias URI para identificar las cosas referidas a las declaraciones en RDF, y el uso de RDF/XML como una forma procesable por máquina para representar las declaraciones RDF.

OWL añade más vocabulario para describir propiedades y clases, relaciones entre clases (por ejemplo, desunión), cardinalidad (por ejemplo, "uno exacto"), igualdad, más tipos de propiedades, características de propiedades (por ejemplo, simetría) y clases enumeradas.

El lenguaje OWL tiene tres sub-lenguajes que incrementan su expresión, cada uno con nivel de expresividad mayor que el anterior: OWL Lite, OWL DL, y OWL Full.

OWL Lite está diseñado para aquellos usuarios que necesitan principalmente una clasificación jerárquica y restricciones simples. Por ejemplo, a la vez que admite restricciones de cardinalidad, sólo permite establecer valores cardinales de 0 ó 1. OWL Lite proporciona una ruta rápida de migración para tesauros y otras taxonomías.

OWL DL está diseñado para aquellos usuarios que quieren la máxima expresividad conservando completitud computacional (se garantiza que todas las conclusiones sean computables), y resolubilidad (todos los cálculos se resolverán en un tiempo finito). OWL DL incluye todas las construcciones del lenguaje de OWL, pero sólo pueden ser usados bajo ciertas restricciones (por ejemplo, mientras una clase puede ser una subclase de otras muchas clases, una clase no puede ser una instancia de otra). OWL DL es denominado de esta forma debido a su correspondencia con la lógica descriptiva.

OWL Full está dirigido a aquellos usuarios que necesitan la máxima expresividad y la libertad sintáctica de RDF pero sin garantías computacionales. Permite, por ejemplo, aumentar el significado de vocabulario predefinido (en RDF o en OWL), por lo que es muy improbable que ningún software de razonamiento sea capaz de soportar razonamiento completo para cualquier característica de OWL Full.

OWL Full se puede ver como una extensión de RDFS, mientras que OWL Lite y OWL DL se pueden ver como extensiones de una vista restringida de RDF. Cualquier documento OWL (Lite, DL, Full) es un documento RDF, y cualquier documento RDF es un documento OWL Full pero sólo algunos documentos RDF serán documentos OWL Lite o OWL DL legales. Por esta razón, hay que tener cuidado cuando se desea migrar un documento RDF a OWL.

Los desarrolladores de ontologías que adoptan OWL deben considerar cuál es el sublenguaje que mejor se adapta a sus necesidades. La elección entre OWL Lite y OWL DL depende de las necesidades de los usuarios sobre la expresividad de las construcciones, siendo OWL DL el que proporciona construcciones más expresivas. La elección entre OWL DL y OWL Full depende principalmente de las necesidades de los usuarios sobre los recursos de metamodelado del esquema RDF (por ejemplo, definir clases de clases, o definir propiedades de clases). Cuando se usa OWL Full en comparación con OWL DL, el soporte en el razonamiento es menos predecible, ya que no existen en este momento implementaciones completas de OWL Full.

3.4.3. Frames y OWL: similitudes y diferencias [34]

Actualmente OWL y Frames son considerados como los paradigmas de modelado de ontologías más usados. Ambos paradigmas tienen constructores similares: ambas son construidas sobre la noción de clases representando conceptos de un dominio, las clases tienen instancias, propiedades para describir los atributos de esas clases, relaciones entre las clases y restricciones que expresan limitaciones sobre los valores de las propiedades. Sin embargo, se presentan diferencias en la semántica de esos constructores y en la manera como esos constructores son usados para inferir nuevos

hechos en la ontología o para determinar si la ontología es consistente. A continuación se presentan algunas diferencias semánticas entre Frames y OWL DL.

3.4.3.1. Diferencias Semánticas

Las siguientes representan algunas de las diferencias más relevantes entre Frames y OWL DL.

Suposición de nombres únicos: En Frames, si dos objetos tienen nombres diferentes se supone que son diferentes, en OWL DL esta suposición no se hace.

Suposición de mundos cerrados *versus* Suposición de mundos abiertos: En Frames todo es prohibido hasta que sea permitido, en OWL DL todo es permitido hasta que sea prohibido.

Un solo modelo *versus* múltiples modelos: Una ontología en Frames solo tiene un modelo, este modelo satisface cada una de las afirmaciones de la ontología, es decir que los modelos para ontologías en Frames solo pueden contener instancias que son especificadas explícitamente. Una ontología en OWL DL tendrá muchos modelos compuestos por todas las posibles interpretaciones que satisfagan cada una de las afirmaciones de la ontología.

Estas diferencias tienen implicación directa en el funcionamiento de la inferencia, y por lo tanto tiene implicaciones directas para el uso de los constructores de modelado de ambas técnicas.

Afirmaciones *versus* clasificación: En Frames, se hacen afirmaciones sobre todas las instancias de una clase cuando se describen las condiciones *necesarias* para las instancias de esa clase, esto se hace al definir facetas para una propiedad de una clase o restricciones para una propiedad a un nivel superior. En OWL DL hay dos tipos de afirmaciones sobre clases: aquellas que, como en Frames, deben ser verdaderas para todas las instancias de una clase y aquellas que son *necesarias y suficientes* para reconocer miembros de una clase. Un clasificador de OWL DL puede utilizar las condiciones suficientes para inferir cuáles clases son subclases de una clase definida, Frames no tiene una característica equivalente para esto.

Verificación de restricciones *versus* verificación de consistencia: El mismo razonador que valida la clasificación, también valida que una base de conocimiento de OWL DL sea consistente. El clasificador trata de construir un modelo que satisfaga todos los axiomas en la ontología, en caso de que el modelo no pueda ser construido la ontología no es consistente, en caso contrario al construir un modelo que satisfaga todas las afirmaciones, el clasificador debe asignar nuevos tipos a las instancias, adicionales a los tipos establecidos explícitamente por el modelador. Un razonador de Frames solo valida si los valores de las instancias para las propiedades cumplen con las restricciones, si no cumplen con las restricciones la instancia no es conformada, en Frames no se pueden asignar nuevos tipos a una instancia por medio de la inferencia.

Otra diferencia importante incluye la asociación de propiedades a clases e individuos: En OWL DL las propiedades pueden ser usadas con una clase o individuo a menos

que la propiedad no cumpla con las restricciones explícitamente especificadas. En Frames, las propiedades deben ser explícitamente asignadas a las clases antes de crearles facetas o de que sean usadas para asignarle valores para una instancia de clase. El trato de múltiples dominios y rango para las propiedades también es diferente. En OWL DL los múltiples dominios y rangos son tratados como una intersección y en Frames como una unión.

Todas estas diferencias traen como consecuencia que el estilo de modelado para ambas técnicas sea diferente. Un desarrollador de una ontología en OWL DL debe pensar en términos de condiciones necesarias y suficientes para definir clases, y un desarrollador de una ontología en Frames se debe enfocar en definir para una instancia cuales son las implicaciones de ser miembro de una clase particular.

3.4.3.2. Ventajas de expresividad de OWL DL

Definición de clases: Como se mencionó anteriormente OWL DL permite definir las condiciones *necesarias y suficientes* para una clase, permitiendo inferir subclases de la clase, en Frames no hay una característica que sea equivalente a ésta.

Definición de conceptos anónimos: OWL permite construir expresiones complejas, sin tener que nombrar a cada concepto intermediario, antes de usarlo. Esto reduce el número de hechos de la ontología, que se deben mantener explícitamente.

Conjunto de combinaciones para clases: OWL DL le permite a los usuarios aplicar el conjunto de operadores estándar (unión, intersección y complemento) en la descripción de clases. Con la combinación de los axiomas de clases, como clases equivalentes, se pueden modelar enunciados como cubrimiento y disyunciones.

Características de las propiedades: En Frames los usuarios pueden decir que una propiedad es funcional o simétrica. En OWL DL a parte de funcional y simétrica, las propiedades pueden ser transitivas.

4. CONSTRUCCIÓN DE LA ONTOLOGÍA

Como se mencionó anteriormente la ontología propuesta cubre tres lenguajes de modelado gráfico: UML, SysML y WebML. Las razones por las que se escogieron estos tres lenguajes para determinar el alcance de la ontología fueron:

- UML es el lenguaje de modelado más conocido y utilizado en la actualidad, se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh, y a que incorpora las principales ventajas de cada uno de los métodos particulares.
- SysML es un subconjunto ampliado de UML, y surgió por la necesidad de un lenguaje que permita especificar sistemas, sean estos informáticos o no, aunque su mayor fortaleza como lenguaje es para aquellos sistemas que combinen elementos del mundo físico (Hardware) con elementos del mundo lógico (Software).

- UML y SysML están respaldados por el OMG, convirtiéndose en lenguajes estándar para el desarrollo de software orientado a objetos y para modelar sistemas, respectivamente, pues el OMG busca el desarrollo de especificaciones para la industria, que sean técnicamente “excelentes”, comercialmente viables e independientes del vendedor.
- Con la introducción del Internet y de la Web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso y uso de información desde cualquier parte del mundo. Con los avances en tecnología, cada vez se demandan aplicaciones más rápidas, ligeras y robustas que permitan ser usadas sin importar ni el lugar, ni el horario. Las aplicaciones Web tienen varias ventajas sobre los programas de software tradicionales, como la compatibilidad multiplataforma y el acceso concurrente de múltiples usuarios. WebML es un lenguaje de especificación de alto nivel, para diseñar aplicaciones Web que usan datos intensivos, que además cubre aspectos avanzados de modelado de sitios Web, incluyendo presentación, modelado de usuarios y personalización.

Los pasos a seguir para el desarrollo y construcción de la ontología son los siguientes:

- Determinación del dominio y el alcance de la ontología: se determina el dominio y el alcance a través de una serie de preguntas de competencia que la ontología debería ser capaz de contestar.
- Obtención de los términos relevantes para la ontología: se listan los términos que tienen relación con el dominio. Algunos de estos términos se transformarán en clases, otros en propiedades y otros en instancias.
- Definición de clases y organización en una jerarquía taxonómica: partiendo de la lista del paso anterior se seleccionan los términos que describen objetos con existencia independiente para constituir los conceptos y organizarlos jerárquicamente.
- Definición de propiedades: partiendo de la lista de términos, se seleccionan aquellos que describen cómo son los objetos con existencia independiente y se asocian a la clase correspondiente. En este paso se describen las características y la estructura de cada clase.
- Definición de facetas para las propiedades: se definen las facetas que describen o caracterizan el tipo de valor que posee cada propiedad.
- Definición de aserciones para las clases: a través de condiciones de aserciones, se define de forma más precisa cada una de las clases.
- Creación de individuos: se crean instancias para cada clase y se llenan los valores de sus propiedades.
- Validación de la ontología: se valida la ontología haciendo uso de las preguntas de competencia y verificando la consistencia con del razonador Pellet.

4.1. Determinación del dominio y el alcance de la ontología

Se definieron las siguientes preguntas de competencia para determinar el alcance, el dominio y el propósito de la ontología.

- ¿Qué características de un sistema se pueden modelar con un lenguaje de modelado gráfico?
- ¿Que sistemas pueden ser modelados con UML?
- ¿Que sistemas pueden ser modelados con WebML?
- ¿Que sistemas pueden ser modelados con SysML?
- ¿Cuáles son los diagramas que permiten modelar características estructurales?
- ¿Cuáles son los diagramas que permiten modelar características de comportamiento?
- ¿Qué características de un sistema se pueden modelar con los diagramas de estructura?
- ¿Qué características de un sistema se pueden modelar con los diagramas de comportamiento?
- ¿Qué características de un sistema se pueden modelar con UML?
- ¿Qué características de un sistema se pueden modelar con SysML?
- ¿Qué características de un sistema se pueden modelar con WebML?
- ¿Qué características de un sistema se pueden modelar con un diagrama determinado?
- ¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de UML?
- ¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de SysML?
- ¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de WebML?
- ¿Qué diagramas se pueden emplear para modelar las interacciones de un sistema?
- ¿Qué diagramas se pueden emplear para modelar estados en un sistema?

- ¿Cuáles son los diagramas que conforman a UML?
- ¿Cuáles son los diagramas que conforman a SysML?
- ¿Cuáles son los diagramas que conforman a WebML?
- ¿Cuáles son las notaciones empleadas en UML y el propósito de cada una de ellas?
- ¿Cuáles son las notaciones de SysML y el propósito de cada una de ellas?
- ¿Cuáles son las notaciones empleadas en WebML y el propósito de cada una de ellas?
- ¿Cuáles son las notaciones empleadas en un diagrama determinado?
- ¿Cuál es el objetivo de un conector determinado?
- ¿Cuál es el objetivo de un elemento determinado?
- ¿Cuáles son las reglas empleadas para la construcción de un diagrama determinado?
- ¿Cuáles son los diagramas de UML que son usados en SysML?
- ¿Cuáles son los componentes de un determinado lenguaje de modelado?

Considerando las anteriores preguntas de competencia, el dominio que se determina para la ontología es la representación de los componentes de lenguajes de modelado gráfico (LMG) y el objetivo de cada uno de ellos, el alcance cubre los lenguajes de modelado UML, SysML y WebML y el propósito es dar soporte a los integrantes de un proyecto informático para aproximarlos a conocer, comprender e identificar cuál es la estructura de los LMG y su utilidad, con el propósito de ayudarlos a identificar el LMG y los diagramas que sean más acordes al proyecto informático que pretenden desarrollar.

4.2. Obtención de los términos relevantes para la ontología

Al analizar el dominio, se identifican una serie de términos relevantes que en pasos posteriores ayudarán a definir clases, propiedades e instancias de la ontología. Después de identificar los términos más importantes del dominio, se organizan en tres grupos según el rol que podrían jugar en la ontología, en el primer grupo se colocan aquellos términos que describen objetos con existencia independiente, en el segundo grupo aquellos términos que describen cómo son esos objetos y en el tercer grupo los términos restantes. Los términos del primer grupo corresponden a posibles clases y los del segundo corresponden a posibles propiedades de clase. Del tercer grupo se deben seleccionar los términos que pasarán a ser posibles Individuos de clase. En el primer grupo se identifican 13 términos, en el segundo 20 términos y en el tercero 130 términos.

4.3. Definición de clases y organización en una jerarquía taxonómica

A partir de la lista de términos definida en el paso anterior y de su organización, se seleccionan los términos candidatos a ser clases (Tabla 3), de esos términos seleccionados se analizan las relaciones existentes entre ellos y finalmente se determinan las clases que forman parte de la ontología. En Tabla 4 se listan las clases definidas para la ontología propuesta.

	Término
1	Lenguaje de modelado gráfico
2	Diagrama
3	Notación
4	Símbolo
5	Elemento
6	Relación
7	Conector
8	Regla
9	Modelo
10	Componente
11	Sistema
12	Organización
13	Característica del sistema

Tabla 3. Términos candidatos a ser clase

Clases	Definición
Lenguaje de Modelado (LMG)	Conjunto de notaciones y reglas a través de las cuales es posible representar el funcionamiento y la estructura de un sistema
Componente LMG	Elementos que constituyen un lenguaje de modela gráfico
Diagrama	Representación gráfica de un conjunto de elementos con sus relaciones
Notación	Conjunto de símbolos gráficos
Elemento	Notación que corresponde a abstracciones de cosas reales o ficticias (objetos, acciones, etc.)
Conector	Notación que corresponde a abstracciones que actúan como unión entre los diferentes elementos
Regla	Pautas que especifican la sintaxis y la semántica de las notaciones
Sistema	Reunión o conjunto de elementos relacionados que interactúan entre sí para lograr un fin determinado
Característica Sistema	Características que identifican un sistema

Tabla 4. Clases de la ontología

Existen diversas aproximaciones a la hora de definir la jerarquía de clases [1]:

- Top-down: consiste en comenzar con la definición de los conceptos o clases más generales del dominio y posteriormente llevar a cabo su especialización,

esta aproximación fue la que se tuvo en cuenta a la hora de definir la jerarquía de clases para la ontología.

- Botom-up: contraria a la aproximación anterior, ésta consiste en comenzar con la definición de los conceptos o clases más específicas, reagrupando posteriormente estas clases en conceptos más generales.
- Combinación de las anteriores: consiste en comenzar con la enumeración de los conceptos o clases más destacadas y posteriormente se generalizan y especializan apropiadamente.

Teniendo la lista de clases, se construyó en la herramienta Protégé-OWL, la jerarquía utilizada para estructurar las clases que componen la ontología (Imagen 12).

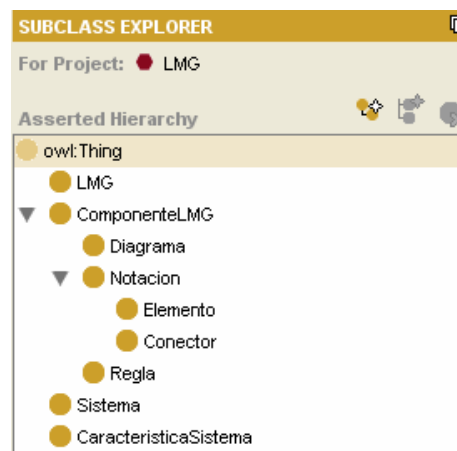


Imagen 12. Jerarquía de clases

4.4. Definición de propiedades

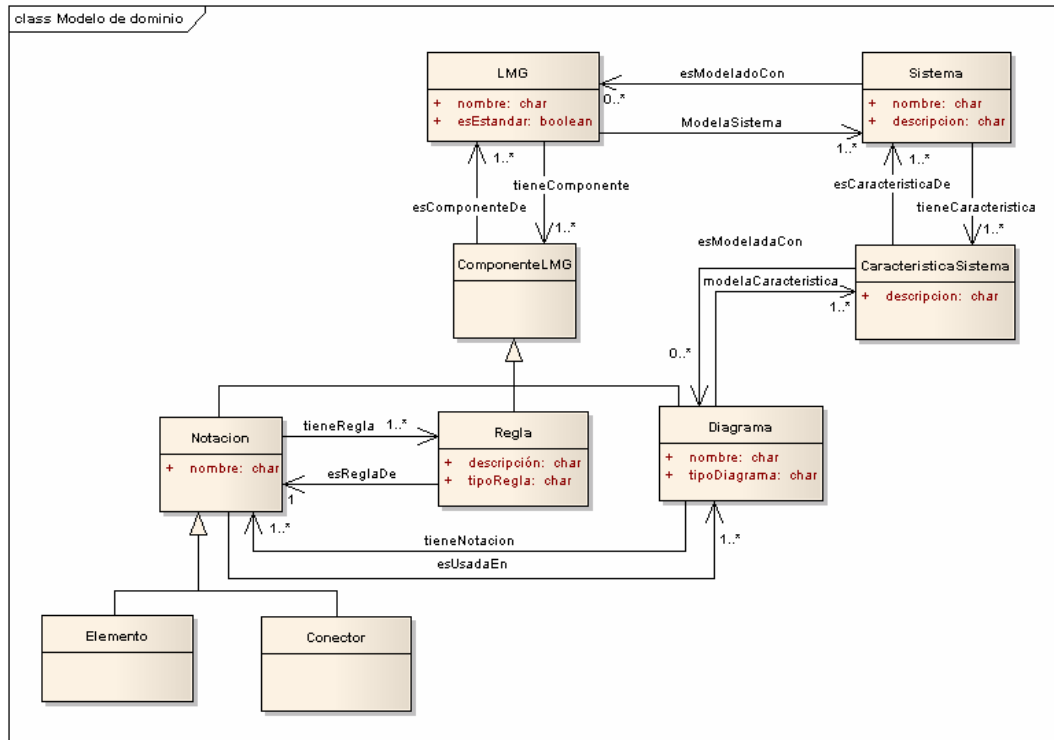


Imagen 13. Modelo del dominio Lenguajes de Modelado Gráfico

A partir de la lista de términos definida y de su organización, se seleccionan los términos candidatos a ser propiedades, de esos términos seleccionados se analizan las relaciones existentes entre ellos y las clases definidas, para finalmente determinar las propiedades que forman parte de la ontología, para asociar dichas propiedades a la clase correspondiente, en este trabajo se define, tomando como referencia a [20] [25] [28], un modelo de dominio que permite asegurar la inclusión de todos los términos relevantes del dominio y todas las relaciones existentes entre ellos. En Imagen 13 [Elaboración propia] se muestra el modelo de dominio resultante.

Teniendo en cuenta el modelo del dominio, en Protégé-OWL, se crean las propiedades que forman parte de la ontología (Imagen 14). Dentro de las propiedades que se crean hay propiedades cuyo valor corresponde a una o varias instancias de una clase y hay propiedades cuyo valor corresponde a un tipo de dato.

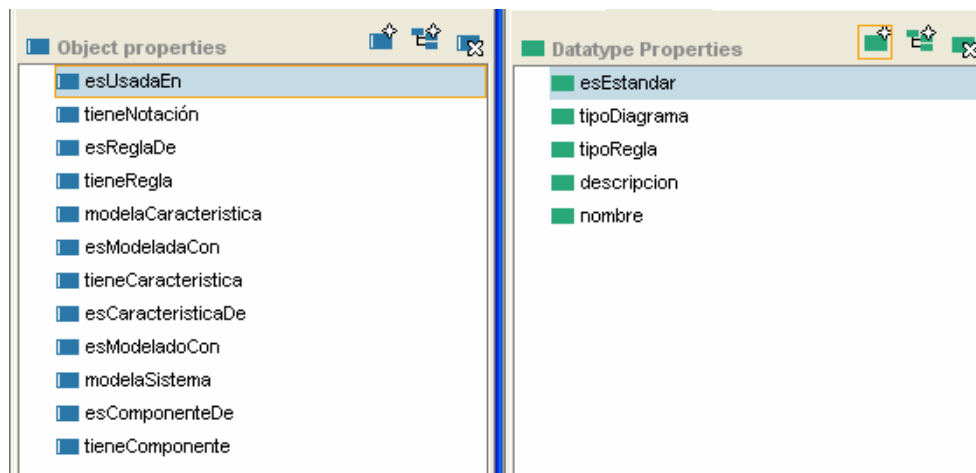


Imagen 14. Propiedades de la ontología

4.5. Definición de facetas para las propiedades

Para cada propiedad creada en el paso anterior, se definen las facetas que describen o caracterizan su tipo de valor. Para cada propiedad es posible definir la cardinalidad, el tipo de valor, el dominio, el rango, valores permitidos y el tipo de propiedad.

Cardinalidad: establece cuantos valores puede tener una propiedad.

Tipo de valor: describe el tipo de valor que posee una propiedad. Los tipos de valores más comunes son *String*, *Int*, *Boolean* e Instancia.

Dominio: conjunto de clases que describe o caracteriza la propiedad.

Rango: clases permitidas para una propiedad de tipo instancia. Para el rango es posible definir aserciones utilizando expresiones OWL.

Valores permitidos: lista de valores permitidos para una propiedad.

Tipo de propiedad: características de la propiedad. Las más comunes son: Funcional (propiedades con un valor único), Simétrica (si una propiedad es simétrica, y el par (x, y) es una instancia de esa propiedad simétrica P , entonces el par (y, x) es también una instancia de la propiedad simétrica P), Transitiva (una propiedad es transitiva, si el par (x, y) es una instancia de la propiedad transitiva P , y el par (y, z) es otra instancia de la propiedad transitiva P , entonces el par (x, z) también es una instancia de P) e Inversa (propiedad inversa de otra propiedad, si se estableciera la propiedad $P1$ como inversa de la propiedad $P2$, y relacionáramos X con Y mediante la propiedad $P2$, entonces Y estaría relacionado con X mediante la propiedad $P1$).

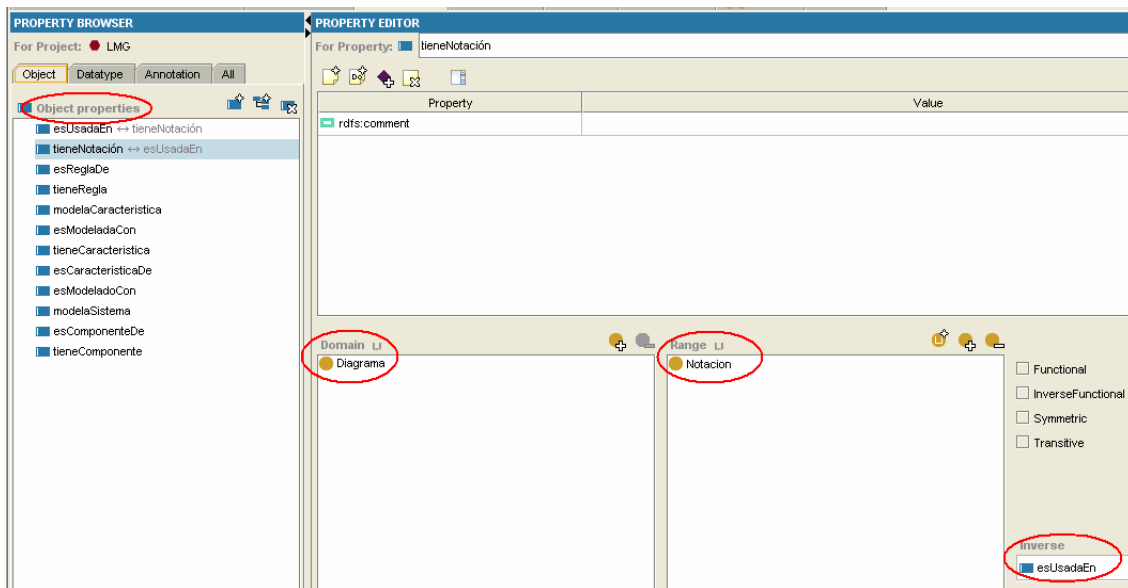


Imagen 15. Facetas para la propiedad *tieneNotación*

En Imagen 15 se muestra la definición en Protégé-OWL de las facetas para la propiedad *tieneNotación*, perteneciente a la clase *Diagrama*. El tipo de Valor de la propiedad es *Instancia*, el dominio esta compuesto por la clase *Diagrama* y el rango por la clase *Notación*, es una propiedad inversa a la propiedad *esUsadaEn*.

En Imagen 16 se muestra la definición de las facetas para la propiedad *Funcional tipoDiagrama*, perteneciente a la clase *Diagrama*. El rango de la propiedad esta compuesto por valores de tipo *String*, el dominio de la propiedad esta compuesto por la clase *Diagrama*, los valores permitidos son “*Estructura*” y “*Comportamiento*”.

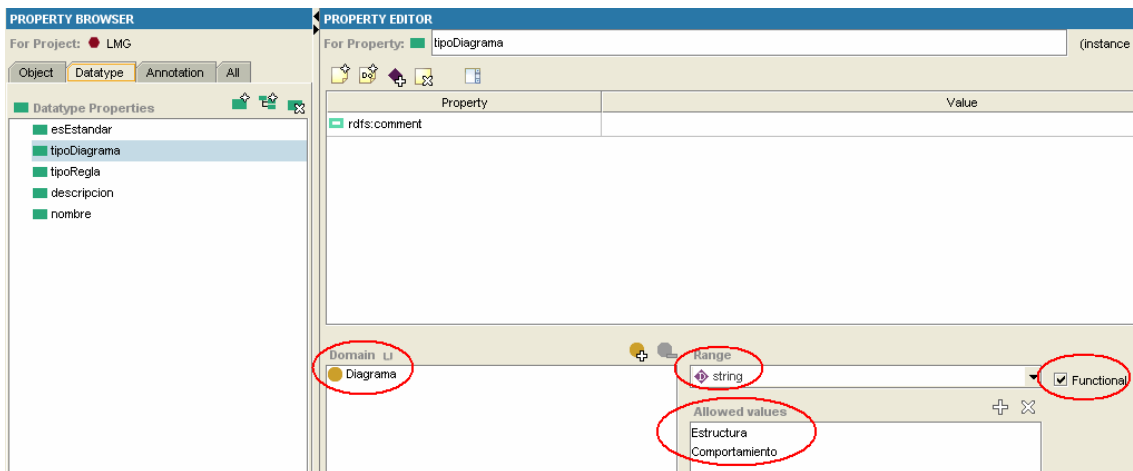


Imagen 16. Facetas de la propiedad *tipoDiagrama*

En Imagen 17 se muestran todas las propiedades de la clase *Diagrama* y las facetas para cada propiedad. En la imagen se indica que un diagrama es una clase que debe modelar al menos una característica de un sistema, debe tener un solo nombre, debe estar compuesto por al menos dos notaciones, debe contener al menos un elemento y un conector, es un componente de al menos un lenguaje de modelado gráfico y debe ser un diagrama de estructura o de comportamiento.

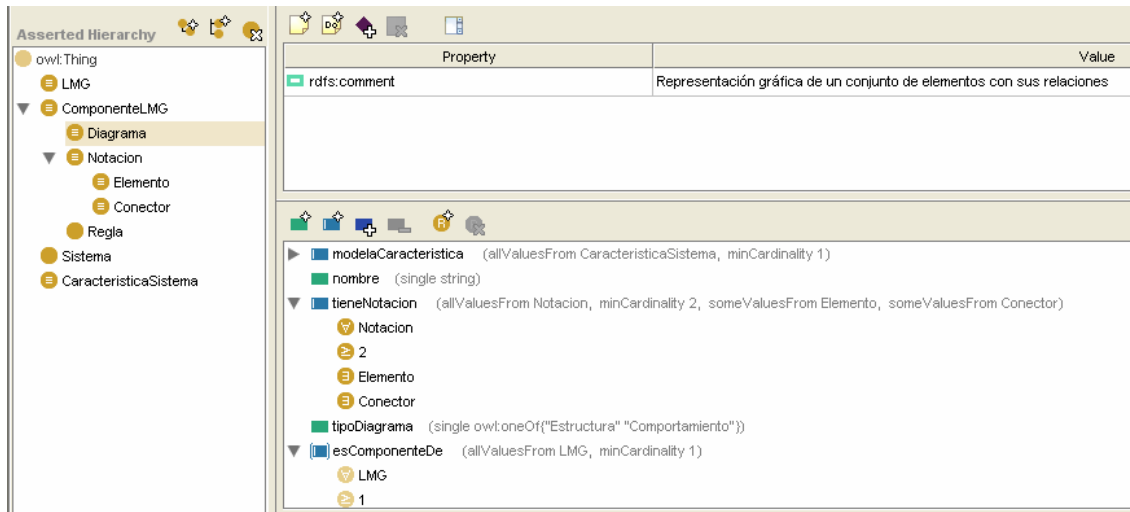


Imagen 17. Propiedades de la clase Diagrama y facetas de cada propiedad

4.6. Definición de Aserciones para las clases

Se da una descripción más precisa de cada clase, a través de la definición de condiciones necesarias y suficientes y de expresiones OWL.

Una condición necesaria indica que un elemento o hecho es condición necesaria de otro cuando, se debe dar el 1º para que se dé el 2º, pero que se dé el 1º no es una condición suficiente para que se dé el 2º. Lo que si puede decirse es que dado el 2º, se puede afirmar con seguridad la existencia del 1º.

Una condición necesaria y suficiente indica que, un elemento o hecho es condición necesaria y suficiente de otro, si al darse el 1º, se da necesariamente el 2º, y a la inversa.

Las expresiones OWL más comunes son:

allValuesFrom: La restricción *allValuesFrom* se establece sobre una propiedad con respecto a una clase. Una clase particular puede tener una restricción sobre una propiedad que haga que todos los valores para esa propiedad sean de un tipo concreto..

someValuesFrom: La restricción *someValuesFrom* se establece sobre una propiedad con respecto a una clase. Una clase particular puede tener una restricción sobre una propiedad que haga que al menos un valor para esa propiedad sea de un tipo concreto.

oneOf. (Clases enumeradas): Las clases se pueden describir mediante la enumeración de los individuos que la componen. Los miembros de la clase son exactamente el grupo de los individuos enumerados: ni más, ni menos.

hasValue: (Valores de la propiedad): Para una propiedad puede ser necesario que tenga un determinado individuo como un valor.

disjointWith: Es posible establecer que las clases sean disjuntas unas de otras.

unionOf, complementOf, intersectionOf (Combinaciones booleanas): OWL DL y OWL Full permiten combinaciones booleanas arbitrarias de clases y de restricciones.

minCardinality, maxCardinality, cardinality (Cardinalidad completa): Mientras que en OWL Lite, las cardinalidades se ciñen a como mínimo, como máximo o exactamente 1 ó 0, OWL Full permite realizar declaraciones de cardinalidad para números enteros no negativos arbitrarios.

En Imagen 18 se muestran las condiciones de aserción creadas para la clase *ComponenteLMG*. Para que un individuo sea miembro de la clase *ComponenteLMG* basta con que cumpla con las siguientes condiciones necesarias y suficientes: el individuo debe estar relacionado con mínimo un individuo de la clase *LMG* a través de la propiedad *esComponenteDe* y todos los individuos con los que está relacionado a través de la propiedad *esComponenteDe* deben ser individuos de la clase *LMG*. Igualmente es necesario que el individuo pertenezca a la clase *Diagrama* o a la clase *Notacion* o a la clase *Regla*. Para la definición de las condiciones se hace uso de las expresiones *allValuesFrom*, *minCardinality* y *unionOf*.

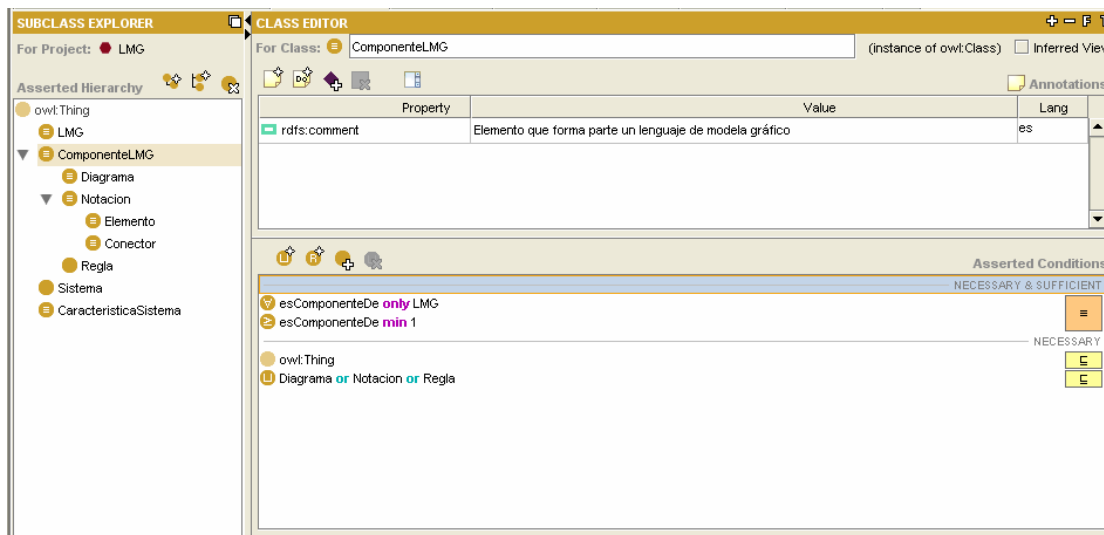


Imagen 18. Aserciones para la clase *ComponenteLMG*

4.7. Creación de individuos

Después de tener la definición de todas las clases que componen la ontología y de cada una de las propiedades de esas clases, se crean los individuos, la creación de los individuos exige, elegir una clase, crear un individuo para esa clase y llenar los valores de sus propiedades.

En Imagen 19 se muestran los individuos creados para la clase Diagrama, y se muestran los valores introducidos en las propiedades para el individuo Diagrama_de_Casos_de_uso.

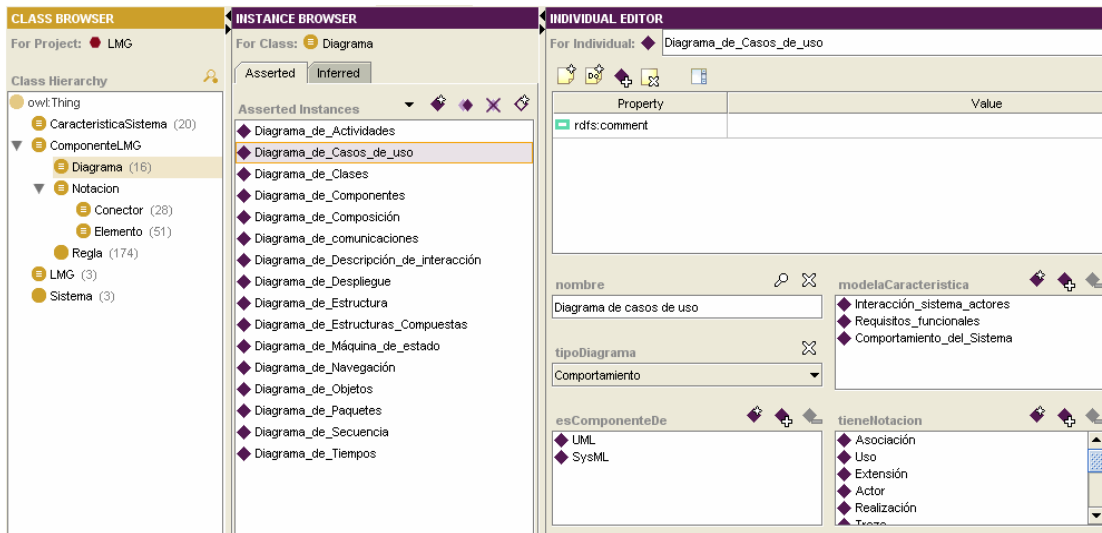


Imagen 19. Creación de individuos para la clase Diagrama

4.8. Validación de la ontología

Validar la ontología significa asegurar que no existen contradicciones, ver las relaciones de herencia que se pueden inferir de las aserciones y verificar que la ontología sea capaz de responder a las preguntas de competencias planteadas.

Para la validación de la ontología se utiliza el razonador *Racer*, con éste se realizan los siguientes pasos:

- Validación de consistencia: Verificar que no existan clases con una definición que no permita tener nunca instancias.
- Clasificación de las clases de la ontología: Verificar si el razonador puede inferir más relaciones de herencia o equivalencia entre clases.

Racer es un razonador comercial basado en lógica de descripciones, éste implementa mecanismos de razonamiento sobre la terminología del dominio (*TBOX*) y sobre el conjunto de Afirmaciones (*ABOX*). Para la validación de la consistencia de la ontología

se hace uso de la interfaz de usuario gráfica de *RacerPro*, llamada *RacerPorter* y de la conexión que proporciona Protégé OWL por medio de *DIG/HTTP (DL Implementation Group / HyperText Transfer Protocol)* para activar el razonador como servidor *HTTP* en el puerto 8080.

En Imagen 20 se muestran los resultados positivos de verificar la coherencia de la terminología del dominio y de verificar la consistencia de las afirmaciones, a través de *RacerPorter*. Al seleccionar las opciones para verificar la consistencia y la coherencia, *RacerPorter* muestra en el *log* (parte inferior) la respuesta, muestra una T cuando es coherente o consistente según la pregunta o NIL en caso contrario.

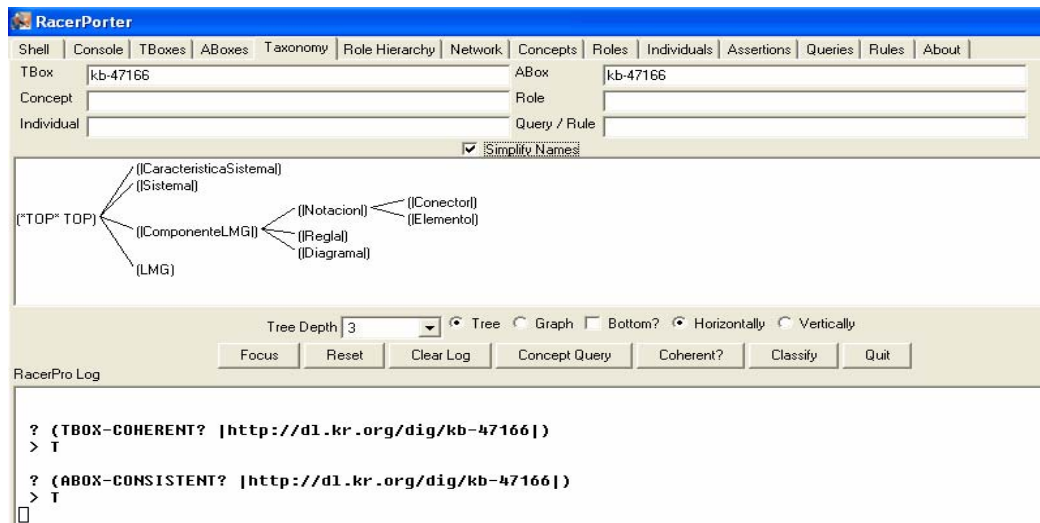


Imagen 20. Verificación de consistencia con *RacerPorter*

En Imagen 21 se muestra la validación de la consistencia de conceptos desde Protégé OWL. Cuando Protégé OWL encuentra una inconsistencia muestra en el *log* del razonador las clases involucradas en las inconsistencias encontradas, y en el explorador de clases las muestra resaltadas en rojo. En caso de no encontrar ninguna inconsistencia solo muestra el tiempo demorado en realizar cada operación de la validación.

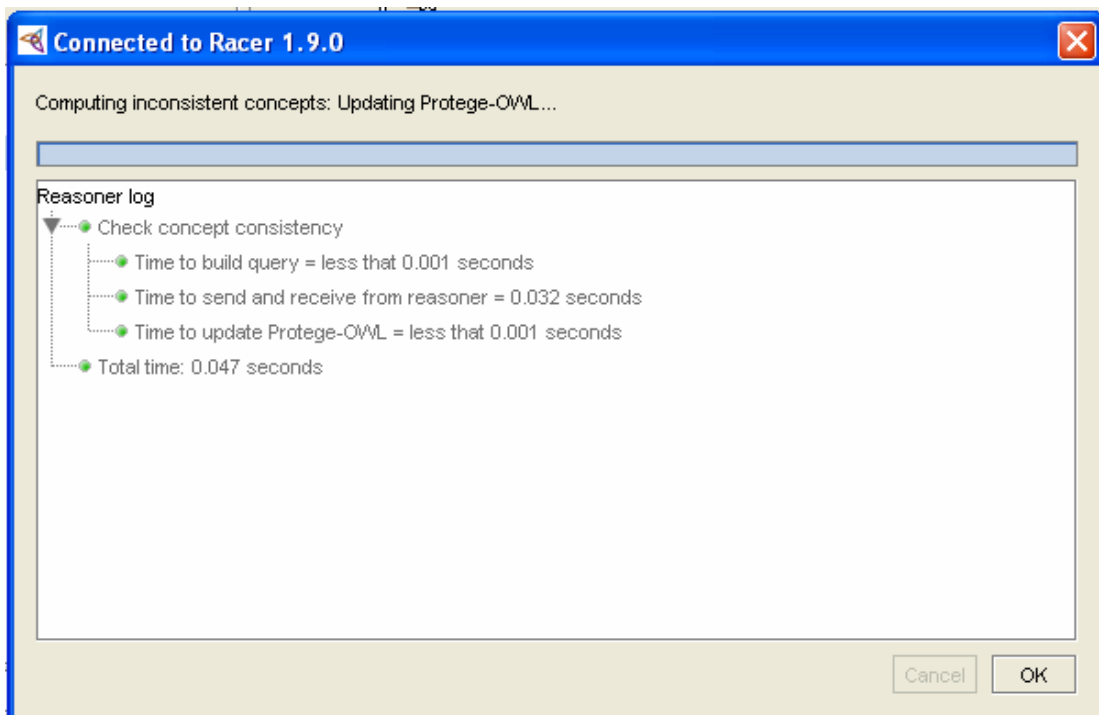


Imagen 21. Verificación de consistencia con *RacerPro* desde Protége OWL

Después de verificar que todas las clases incluidas en la ontología son consistentes, se solicita al razonador la inferencia de nuevas relaciones de herencia o equivalencia entre clases. En Imagen 22 se muestran los resultados del proceso de inferencia que realiza el razonador, la imagen indica que éste verifica la consistencia, realiza el proceso de inferencia de nuevas relaciones de herencia y el proceso de inferencia de equivalencias entre clases, pero para ninguno de los dos procesos encuentra resultados.

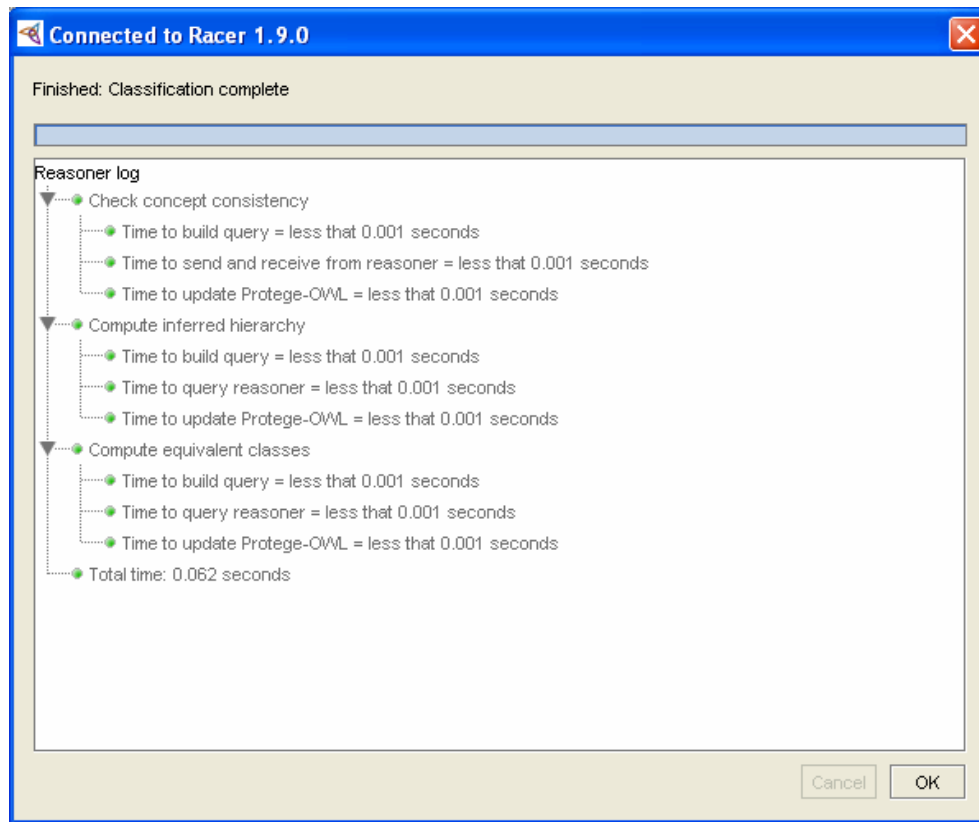


Imagen 22. Inferencia de nuevas relaciones de herencia o equivalencia entre clases con *RacerPro*

Otra de las opciones que ofrece el razonador, es verificar si puede inferir para cada clase, según sus condiciones de aserción, los individuos que son miembros, a parte de los que le fueron asignados explícitamente. En Imagen 23 se muestran los resultados de solicitarle al razonador que realice esta inferencia, se puede observar que el razonador para cada clase infiere el mismo número de miembros asignados explícitamente.

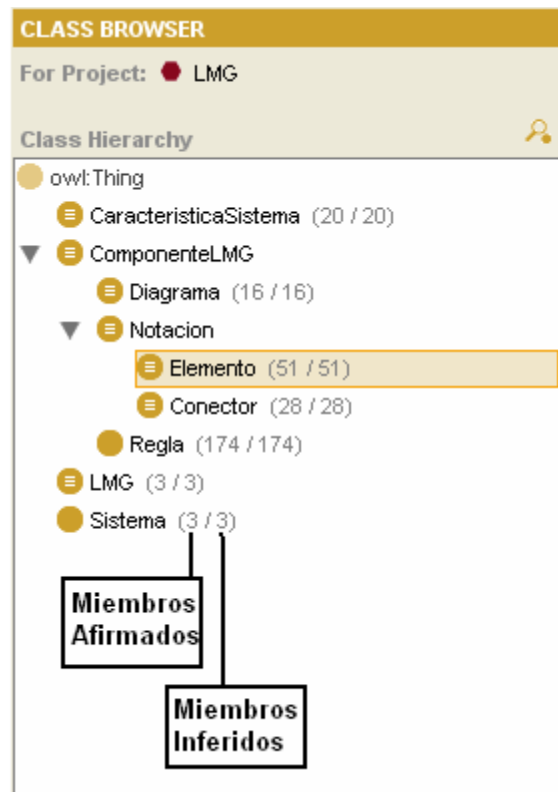


Imagen 23. Inferencia de miembros para clases con Racer

Para mirar un ejemplo de inferencia de miembros para las clases, se modifican las condiciones de aserción necesarias y suficientes de las clases Elemento y Conector, y la clase Elemento se coloca como subclase de ComponenteLMG y no como subclase de Notación. Se le solicita al razonador la inferencia de nuevos miembros para clases, encontrando como resultado 51 nuevos miembros para la clase Notación, esto se debe a que esta clase tiene como condiciones necesarias y suficientes que el miembro sea un Elemento o un Conector y que tenga mínimo dos reglas, estas condiciones las cumplen los 51 miembros de la clase Elemento, que son los nuevos miembros inferidos para la clase Notación. En Imagen 24 se pueden ver estos resultados.

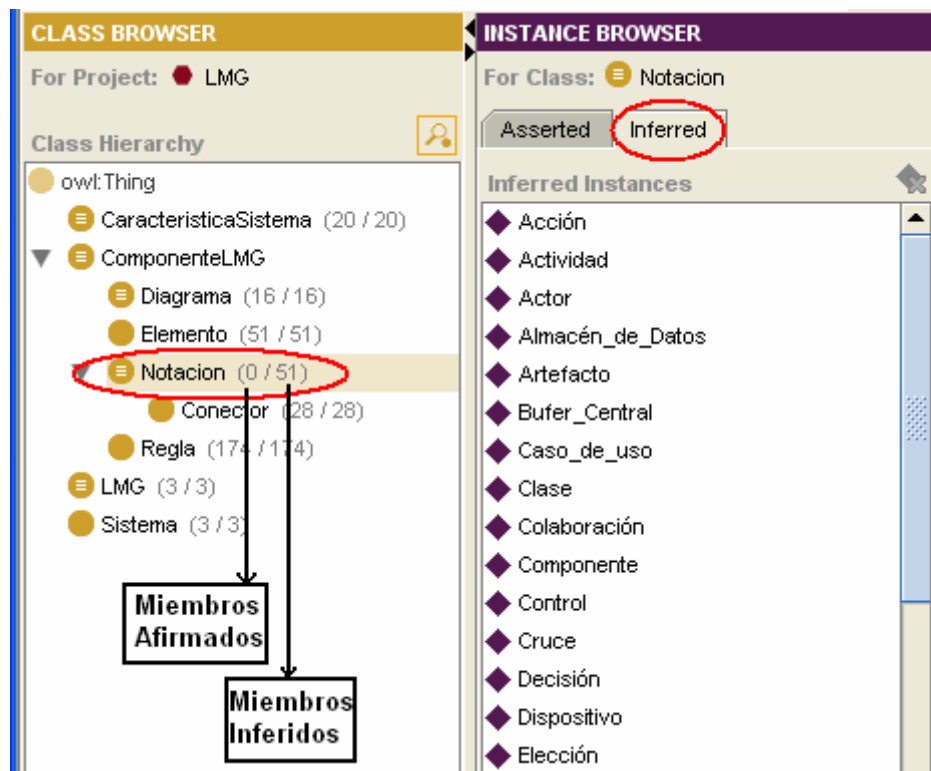


Imagen 24. Ejemplo de inferencia de miembros para clases con Racer

Después de validar la consistencia de la ontología y de solicitarle al razonador los procesos de inferencia, se le realizan a la ontología las preguntas de competencia, para completar el proceso de validación. En Imagen 25 se muestran los resultados de realizar una de las preguntas en la ventana *Queries* de Protégé OWL.

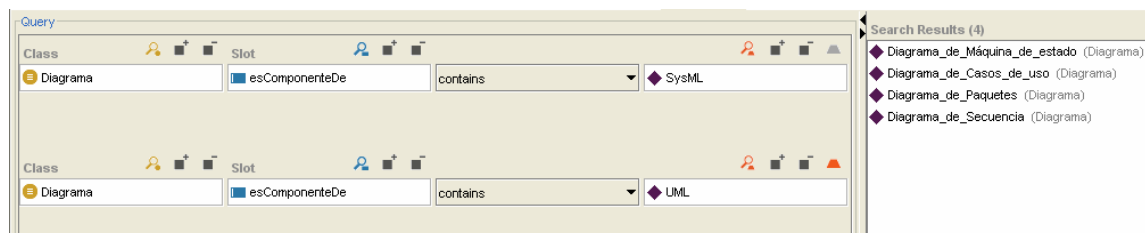


Imagen 25. Ejecución de la pregunta ¿Cuáles son los diagramas de UML que son usados en SysML?

Todas las preguntas pueden ser resueltas por la ontología, pero a través de la ventana *Queries* de Protégé OWL solo se pueden realizar las que se listan a continuación, debido a que a través de esa ventana no se pueden seleccionar las clases que deberían salir como resultado y a que no se pueden realizar consultas con condiciones anidadas con el operador *AND* que involucren varias clases. El resto de preguntas son

realizadas con el lenguaje de recuperación Web SPARQL (*Protocol and RDF Query Language*), que permite realizar búsquedas más complejas. En el Anexo 1 se muestran todas las preguntas de competencia realizadas en SPARQL.

- ¿Que sistemas pueden ser modelados con UML?
- ¿Que sistemas pueden ser modelados con WebML?
- ¿Que sistemas pueden ser modelados con SysML?
- ¿Cuáles son los diagramas que permiten modelar características estructurales?
- ¿Cuáles son los diagramas que permiten modelar características de comportamiento?
- ¿Qué características de un sistema se pueden modelar con un diagrama determinado?
- ¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de UML?
- ¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de SysML?
- ¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de WebML?
- ¿Qué diagramas se pueden emplear para modelar las interacciones de un sistema?
- ¿Qué diagramas se pueden emplear para modelar estados en un sistema?
- ¿Cuáles son los diagramas que conforman a UML?
- ¿Cuáles son los diagramas que conforman a SysML?
- ¿Cuáles son los diagramas que conforman a WebML?
- ¿Cuáles son las notaciones empleadas en UML y el propósito de cada una de ellas?
- ¿Cuáles son las notaciones de SysML y el propósito de cada una de ellas?
- ¿Cuáles son las notaciones empleadas en WebML y el propósito de cada una de ellas?
- ¿Cuáles son las notaciones empleadas en un diagrama determinado?

- ¿Cuál es el objetivo de un conector determinado?
- ¿Cuál es el objetivo de un elemento determinado?
- ¿Cuáles son los diagramas de UML que son usados en SysML?
- ¿Cuáles son los componentes de un determinado lenguaje de modelado?

En Imagen 26 se muestra un ejemplo de una de las preguntas realizadas en SPARQL. La pregunta esta compuesta por dos condiciones, las cuales involucran dos clases, Diagrama y LMG, permitiendo hacer una intersección entre ellas, igualmente está compuesta por la selección de los resultados que deberá retornar.

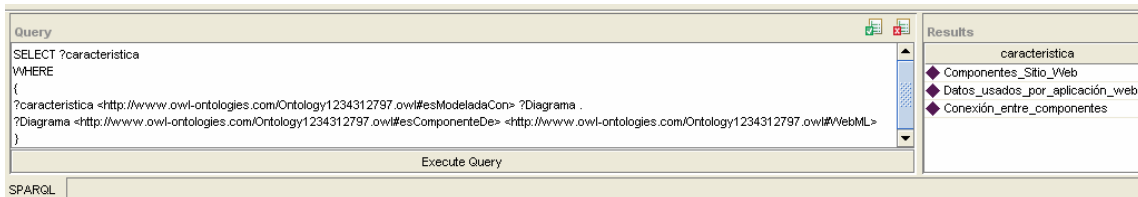


Imagen 26. Ejecución de la pregunta ¿Qué características de un sistema se pueden modelar con WebML? en SPARQL

5. CONCLUSIONES Y TRABAJOS FUTUROS

La ontología que se construyó, sirve como análisis conceptual, para ayudar a los ingenieros de software a seleccionar el lenguaje de modelado más adecuado para modelar y comunicar las características de un sistema particular, pues la ontología estructura y define la información relacionada con tres lenguajes de modelado importantes, permitiendo realizar comparaciones y consultas que ayuden a entender sus especificaciones.

Después de analizar los tres lenguajes de modelado gráfico que forman parte del dominio y el alcance de la ontología, se concluye que UML es el lenguaje que permite modelar más características y mas sistemas, pues SysML, aunque es una extensión de UML, es un lenguaje para modelar principalmente sistemas que involucran hardware y software al tiempo, y WebML solo permite modelar el diseño de sitios Web, más específicamente permite modelar los elementos que componen un sitio Web.

El paradigma de representación del conocimiento seleccionado para estructurar la información sobre los tres lenguajes de modelado fue OWL DL, pues éste optimiza la relación entre la riqueza descriptiva y la posibilidad de razonar e inferir, es decir, permite describir con mayor precisión las clases y las propiedades de esas clases, y con esta descripción más precisa se pueden emplear razonadores sobre la ontología para verificar consistencia de clases y para realizar procesos de inferencia. Además OWL permite añadir a las ontologías la capacidad de ser distribuida a través de varios sistemas, ser escalable a las necesidades de la Web y ser compatible con los estándares Web de accesibilidad e internacionalización.

Se propone realizar en un futuro un estudio más detallado sobre las reglas utilizadas en cada uno de los lenguajes de modelado gráfico, con el propósito de hacer una representación más especializada que permita tener más estructurada la información para aprovechar mejor las ventajas de los razonadores. Otra propuesta es ampliar el alcance de la ontología para que cubra no solo los lenguajes de modelado gráfico, sino también los textuales.

6. REFERENCIAS BIBLIOGRÁFICAS

- [1] Noy, N., McGuinness, D. *Ontology Development 101: A Guide to Creating Your First Ontology*. Knowledge Systems Laboratory Stanford University.
- [2] Fuentes L., Troya, J., and Vallecillo, A. *Using UML Profiles for Documenting Web-Based Application Frameworks*, *Analyse of Software Engineering* 13, pp 249 - 264, Madrid, 2002.
- [3] Giandini, R., Pons, C. *Un lenguaje para Transformación de Modelos basado en MOF y OCL* In: *32th Latin-American Conference on Informatics - CLEI 2006*, Santiago, 2006.
- [4] Fuentes, L., Vallecillo, A. *Una introducción a los perfiles UML, UML y la Ingeniería del Modelado*. *Novática*, vol. 168, 2004
- [5] Chisholm, R. *A Realistic Theory of Categories – An Essay on Ontology*. Cambridge University Press, 1996.
- [6] Milton, S., Kazmierczak E. *An ontology of data modelling languages: a study using a common-sense realistic ontology*. *J. database management*, vol 15(2) pp. 19-38, 2004
- [7] Milton, S., Kazmierczak, E. and Keen, C. *An Ontological Study of Data Modelling Languages using Chisholm's Ontology*. *Proc. 11th European-Japanese Conference Information Modelling and Knowledge bases*, 21-32. Maribor, 2001.
- [8] Chen, P. *The Entity-Relationship Model—Toward a Unified View of Data*. *ACM Transactions on Database Systems*, pp. 9–36, 1976.
- [9] Kerschberg, L. and Pacheco, J. E. S. *A functional database model*, Pontificia Univ. Catholica do Rio de Janeiro, Rio de Janeiro, Brazil, 1976.
- [10] Hammer, M. and Mcleod, D. *Database Description with SDM: A Semantic Database Model*. *ACM Transactions on Database Systems*, pp. 351–386, 1981.
- [11] Hammer, M. and Mcleod, D. In *Research Foundations in Object-oriented and Semantic Database Systems* (Eds, Cárdenas, A. F. and McLeod, D.) Prentice Hall, Englewood Cliffs, pp. 341, 1990.

- [12] Nijssen, G. M. and Halpin, T. A. Conceptual Schema and Relational Database Design: A Fact Oriented Approach, Prentice-Hall, New York, 1989.
- [13] Blaha, M. and Premerlani, W. Object-Oriented Modeling and Design for Database Applications, Prentice Hall, Upper Saddle River, 1998.
- [14] Genilloud, G., Frank, W. Use Case Concepts using a Clear, Consistent, Concise Ontology, Journal of Object technology, Vol. 4, No. 6, Special Issue: Use Case Modeling at UML- Madrid , 2004, 2005.
- [15] Vallecillo, A. RM-ODP: The ISO Reference Model for Open Distributed Processing. DINTEL Edition on Software Engineering. No. 3, pp. 69-99. 2001
- [16] Horridge M., Knublauch H., Rector A., Stevens R. and Wroe C. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and COODE Tools Edition 1.0, The University Of Manchester Stanford University. 2004
- [17] Antoniou, G. and van Harmelen, F. Web ontology language: Owl. In Handbook on Ontologies in Information Systems, pp. 67--92, 2003
- [18] McGuinness D. and van Harmelen F. (eds), 10 de Febrero, 2004, Lenguaje de Ontologías Web (OWL) Vista General, Recomendación W3C, <http://www.w3.org/2007/09/OWL-Overview-es.html>
- [19] Hernández, E. El lenguaje Unificado de Modelado (UML), Manual Formativo Vol. 26 Universidad Politécnica de Valencia, 2002
- [20] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide. Addison Wesley. Edición Dos. 2005
- [21] Schumler , J. Aprendiendo UML en 24 horas. Prentice Hall
- [22] Hamilton K., Miles R., Learning UML 2.0. O'Reilly. 2006
- [23] Balmelli, L. An Overview of the Systems Modeling Language for Products and Systems Development, IBM Technical Report 2006, Journal of Object Technology (JOT) at www.jot.fm, Tokyo, 2006
- [24] Hause, M. The SysML Modelling Language, Fifth European Systems Engineering Conference, Cheltenham, Glos. UK, 2006

- [25] Weilkiens T., Systems engineering with SysML/UML: modeling, analysis, design, Morgan Kaufmann, 2008
- [26] Hause, M and Thom F. Building Bridges Between Systems and Software with SysML and UML, 2008
- [27] Ceri, S., Fraternali, P. and Bongio A. Web modeling language (WebML): a modeling language for designing Web sites, Computer networks - the international journal of computer and telecommunications networking, pp. 137–157, 2000.
- [28] Ceri S., Fraternali P., Bongio A., Brambilla M., Comai S., Matera M., Designing Data-Intensive Web Applications, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2002
- [29] Moreno N., Fraternali P. and Vallecillo A. A UML 2.0 Profile for WebML Modeling, In: Proc. International Workshop on Model-Driven Web Engineering (MDWE2006), 2006.
- [30] Contreras J. and Martínez J. Tutorial ontologías, Madrid, 2007
- [31] Gasevic D., Djuric D., Devedic V. and Selic B., Model Driven Architecture and Ontology Development, Springer, 2006.
- [32] Gomez-Perez A., Corcho O. and Fernandez-Lopez M., Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition, Springer, 2006
- [33] Zapata C. M., Giraldo G. L., Portilla B. E., Gómez D. A., Naranjo M., Carmona P., Una aproximación a una ontología para lenguajes de modelado gráfico. Por aparecer en Revista de Ingeniería de la Universidad de los Andes.
- [34] Wang H., et al., Frames and OWL Side by Side, 9th International Protégé Conference, Stanford, California, 2006

ANEXO 1: PREGUNTAS DE COMPETENCIA EN SPARQL

¿Qué características de un sistema se pueden modelar con un lenguaje de modelado gráfico?

```
SELECT DISTINCT ?Caracteristica
WHERE { ?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esCaracteristicaDe> ?Sistema .
?Sistema <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon> ?Img .
?Img rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#LMG> . }
```

¿Qué sistemas pueden ser modelados con UML?

```
SELECT ?sistemas
WHERE {?sistemas <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML> }
```

¿Qué sistemas pueden ser modelados con WebML?

```
SELECT ?sistemas
WHERE {?sistemas <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#WebML> }
```

¿Qué sistemas pueden ser modelados con SysML?

```
SELECT ?sistemas
WHERE {?sistemas <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> }
```

¿Cuáles son los diagramas que permiten modelar características estructurales?

```
SELECT ?diagramas
WHERE {?diagramas <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoDiagrama>
"Estructura" }
```

¿Cuáles son los diagramas que permiten modelar características de comportamiento?

```
SELECT ?diagramas
WHERE {?diagramas <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoDiagrama> "Comportamiento" }
```

¿Qué características de un sistema se pueden modelar con los diagramas de estructura?

```
SELECT ?caracteristicas
WHERE {?caracteristicas <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon> ?Diagrama .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoDiagrama> "Estructura" }
```

¿Qué características de un sistema se pueden modelar con los diagramas de comportamiento?

```
SELECT ?caracteristicas
WHERE {?caracteristicas <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon> ?Diagrama .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoDiagrama> "Comportamiento" }
```

¿Qué características de un sistema se pueden modelar con UML?

```
SELECT DISTINCT ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esCaracteristicaDe> ?Sistema .
?Sistema <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML> }
```

¿Qué características de un sistema se pueden modelar con SysML?

```
SELECT DISTINCT ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esCaracteristicaDe> ?Sistema .
?Sistema <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> }
```

¿Qué características de un sistema se pueden modelar con WebML?

```
SELECT DISTINCT ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esCaracteristicaDe> ?Sistema .
?Sistema <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladoCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#WebML> }
```

¿Qué características de un sistema se pueden modelar con el diagrama de casos de uso?

```
SELECT DISTINCT ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama_de_Casos_de_uso> }
```

¿Qué características de un sistema se pueden modelar con el diagrama Paramétrico?

```
SELECT DISTINCT ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon>
<http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama_Paramétrico> }
```

¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de UML?

```

SELECT ?Diagrama ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon> ?Diagrama .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML>}
ORDER BY ?Diagrama

```

¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de SysML?

```

SELECT ?Diagrama ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon> ?Diagrama .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML>}
ORDER BY ?Diagrama

```

¿Qué características de un sistema se pueden modelar con cada uno de los diagramas de WebML?

```

SELECT ?Diagrama ?Caracteristica
WHERE {?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#esModeladaCon> ?Diagrama .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#WebML>}
ORDER BY ?Diagrama

```

¿Qué diagramas se pueden emplear para modelar las interacciones de un sistema?

```

SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#modelaCaracteristica>
?Caracteristica .
?Caracteristica rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#CaracteristicaSistema> .
?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
FILTER regex(str(?Descripcion), "interac", "I")}

```

¿Qué diagramas se pueden emplear para modelar estados en un sistema?

```

SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#modelaCaracteristica>
?Caracteristica .
?Caracteristica rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#CaracteristicaSistema> .
?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
FILTER regex(str(?Descripcion), "estado", "I")}

```

¿Qué diagramas de UML se pueden emplear para modelar estados en un sistema?

```

SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#modelaCaracteristica> ?Caracteristica .
?Caracteristica rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#CaracteristicaSistema> .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML> .
?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
FILTER regex(str(?Descripcion), "estado", "I")}

```

¿Qué diagramas de SysML se pueden emplear para modelar estados en un sistema?

```
SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#modelaCaracteristica> ?Caracteristica .
?Caracteristica rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#CaracteristicaSistema> .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> .
?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
FILTER regex(str(?Descripcion), "estado", "I")}
```

```
SELECT ?Diagrama ?Caracteristica
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#modelaCaracteristica> ?Caracteristica .
?Caracteristica rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#CaracteristicaSistema> .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> .
?Caracteristica <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
FILTER regex(str(?Descripcion), "estado", "I")}
```

¿Cuáles son los diagramas que conforman a UML?

```
SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML> .
?Diagrama rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama> }
```

¿Cuáles son los diagramas que conforman a SysML?

```
SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> .
?Diagrama rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama> }
```

¿Cuáles son los diagramas que conforman a WebML?

```
SELECT ?Diagrama
WHERE {?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#WebML> .
?Diagrama rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama> }
```

¿Cuáles son las notaciones empleadas en UML y el propósito de cada una de ellas?

```
SELECT ?Notacion ?Descripcion
WHERE {?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML> .
{?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector>} UNION
{?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento>} .
?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#tieneRegla> ?Regla .
?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoRegla> "Semantica"
}
```

¿Cuáles son las notaciones de SysML y el propósito de cada una de ellas?

```
SELECT ?Notacion ?Descripcion
WHERE {?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> .
 {?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector>} UNION
 {?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento>} .
 ?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#tieneRegla> ?Regla .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoRegla> "Semantica"
}
```

¿Cuáles son las notaciones empleadas en WebML y el propósito de cada una de ellas?

```
SELECT ?Notacion ?Descripcion
WHERE {?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#WebML> .
 {?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector>} UNION
 {?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento>} .
 ?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#tieneRegla> ?Regla .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoRegla> "Semantica" }
```

¿Cuáles son las notaciones empleadas en un diagrama determinado?

```
SELECT ?Diagrama ?Notacion
WHERE {
 {?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector>} UNION
 {?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento>} .
 ?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#esUsadaEn> ?Diagrama .
 ?Diagrama rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama> .}
ORDER BY ?Diagrama
```

¿Cuál es el objetivo de un conector determinado?

```
SELECT ?Conector ?Descripcion
WHERE {
 ?Conector rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector> .
 ?Conector <http://www.owl-ontologies.com/Ontology1234312797.owl#tieneRegla> ?Regla .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoRegla> "Semantica" }
ORDER BY ?Conector
```

¿Cuál es el objetivo de un elemento determinado?

```
SELECT ?Elemento ?Descripcion
WHERE {
 ?Elemento rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento> .
 ?Elemento <http://www.owl-ontologies.com/Ontology1234312797.owl#tieneRegla> ?Regla .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .
 ?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#tipoRegla> "Semantica" }
ORDER BY ?Elemento
```

¿Cuáles son las reglas empleadas para la construcción de un diagrama determinado?

```
SELECT ?Diagrama ?Notacion ?Regla ?Descripcion
WHERE {
?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#esUsadaEn> ?Diagrama .
?Diagrama rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama> .
{?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector>} UNION
{?Notacion rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento>} .
?Notacion <http://www.owl-ontologies.com/Ontology1234312797.owl#tieneRegla> ?Regla .
?Regla <http://www.owl-ontologies.com/Ontology1234312797.owl#descripcion> ?Descripcion .}
ORDER BY ?Diagrama
```

¿Cuáles son los diagramas de UML que son usados en SysML?

```
SELECT ?Diagrama
WHERE {
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#UML> .
?Diagrama <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe>
<http://www.owl-ontologies.com/Ontology1234312797.owl#SysML> }
```

¿Cuáles son los componentes de un determinado lenguaje de modelado?

```
SELECT ?LMG ?Componente
WHERE {
?LMG rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#LMG> .
{?Componente rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Conector>} UNION
{?Componente rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Elemento>} UNION
{?Componente rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Diagrama>} UNION
{?Componente rdf:type <http://www.owl-ontologies.com/Ontology1234312797.owl#Regla>} .
?Componente <http://www.owl-ontologies.com/Ontology1234312797.owl#esComponenteDe> ?LMG}
ORDER BY ?LMG
```

ANEXO 2: RESUMEN
PROPUESTA Y CONSTRUCCIÓN DE UNA ONTOLOGÍA PARA LENGUAJES DE
MODELADO GRÁFICO

Autor: Gloria Patricia Carmona Ríos
Director: Gloria Lucia Giraldo Gómez
Trabajo Final Especialización en Sistemas
Facultad de Minas
Universidad Nacional de Colombia
Sede Medellín
2008-II

RESUMEN

En el presente trabajo se propone el desarrollo y construcción de una ontología, ésta tiene como dominio y alcance tres lenguajes de modelado gráfico existentes actualmente: UML 2.0, SysML, WebML. UML es un lenguaje estándar utilizado para modelar la estructura, la arquitectura y el comportamiento de una aplicación. SysML es un lenguaje basado en UML, el cual permite modelar los requisitos, comportamiento, estructura y parámetros de sistemas complejos. WebML apoya las actividades de diseño de sitios Web.

La ontología propuesta se desarrolla en OWL DL y se implementa en Protégé-OWL, se toma este lenguaje como paradigma de representación del conocimiento pues ofrece mayor poder expresivo comparado con otros paradigmas, permitiendo una mayor capacidad de razonamiento e inferencia al definir un conjunto de aserciones, restricciones y condiciones sobre clases y propiedades, utilizando para estas definiciones expresiones OWL. Otra de las razones por las que se utiliza OWL es porque utiliza la conexión proporcionada por RDF, para añadir a las ontologías la capacidad de ser distribuida a través de varios sistemas, ser escalable a las necesidades de la Web y ser compatible con los estándares Web de accesibilidad e internacionalización.

El propósito de la ontología es realizar un análisis conceptual de los elementos que conforman los tres lenguajes de modelado y así ayudar a los ingenieros de software a comprender las diferentes especificaciones de estos lenguajes, y de esta manera apoyarlos en la decisión de cuál es el lenguaje más adecuado para modelar los requisitos de un sistema en particular.

Palabras clave: Ontología, OWL, lenguaje de modelado gráfico, diagrama, UML, WebML y SysML.

ABSTRACT

This work proposes development and construction of an ontology, which has three currently existing graphic modeling languages as domain and scope: UML 2.0, SysML and WebML. UML is a standard language used for modeling application's structure, architecture and behavior. SysML is a UML-based language that allows modeling requirements, behavior and parameters of complex systems. WebML supports Website design activities.

Proposed ontology is developed on OWL DL and implemented on Protégé OWL, this language is taken as representation paradigm of knowledge because it offers bigger expression power compared with other paradigms, allowing bigger reasoning and inference capacity by defining a set of assertions, constraints and conditions on classes and properties, using OWL expressions for these definitions. Another of the reasons for using OWL is because it uses RDF provided connection, in order to add ability for being distributed across several systems to ontologies, for being scalable to Web needs and for being compatible with accessibility and internationalization Web standards.

Purpose of the ontology is to do conceptual analysis of elements that make up three modeling languages and, so, to assist software engineers to understand these different languages specifications, and in this way support them on their decision about which of the languages is more suitable for modeling particular system requirements.

Key Words: Ontology, OWL, graphic modeling languages, diagram, UML, WebML and SysML.